DD2424 Deep Learning in Data Science

Assignment 3

Option 1: Batch Normalization

Ramona Häuselmann

May 6, 2021

1 Gradient Computation

I managed to successfully write the functions to correctly compute the gradient analytically. I tested my implementation by computing the gradients using ComputeGradsNumSlow.m. Then I compared my results with the numerical approaches by calculating the absolute difference of each gradient element as in equations below. Then I checked those against a threshold (1e-5). When using the reduced set with `n=2` on a 3 layer network with [50, 50] nodes in the hidden layers and using ComputeGradsNumSlow.m I get a maximum error of

- `diff_W1_max = 2.2204e-11`
- `diff_b1_max = 2.2204e-11`
- `diff_gamma1_max = 2.2356e-11`
- `diff_beta1_max = 2.2356e-11`
- `diff_W2_max = 4.1234e-13`
- `diff_b2_max = 2.2204e-16`
- `diff_gamma2_max = 3.3339e-11`
- `diff_beta2_max = 3.3339e-11`
- `diff_W3_max = 2.3122e-11`
- `diff_b3_max = 2.4201e-11`

These errors are small enough to conclude that my implementation works.

$$diff\_W = abs(ngrad\_W - grad\_W) \tag{1}$$

$$diff\_b = abs(ngrad\_b - grad\_b) \tag{2}$$

$$diff\_gamma = abs(ngrad\_gamma - grad\_gamma) \tag{3}$$

$$diff\_beta = abs(ngrad\_beta - grad\_beta) \tag{4}$$

2 3-layer Network

Training a 3-layer network with [50, 50] hidden nodes, lambda=0.005, eta_min=1e-5, eta_max=1e-1, cycles=2, n_batch=100, n_s=5*45000/n_batch.

2.1 Without batch normalization

accuracy_validation = 54.1%
accuracy_test = 53.05%

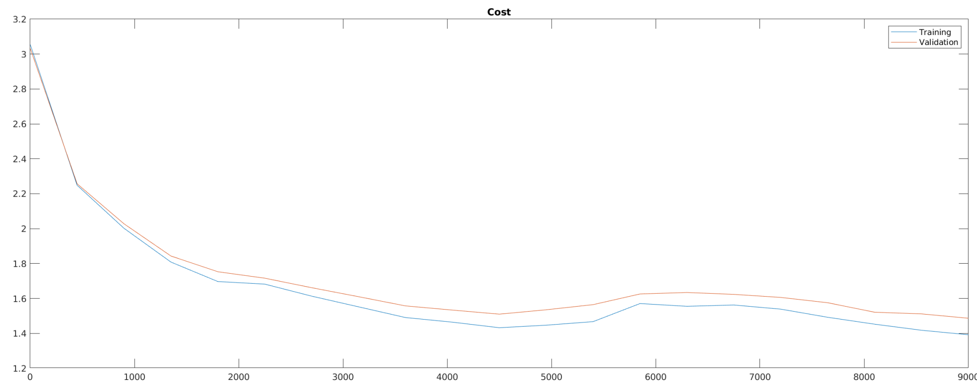

Figure 1: Loss, 3-layer without batch normalization

2.2 With batch normalization

accuracy_validation = 55.1%
accuracy_test = 53.95%

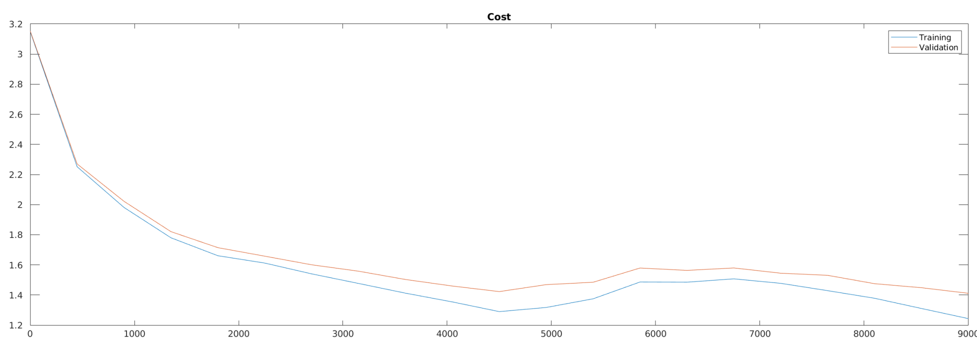

Figure 2: Loss, 3-layer with batch normalization

3 9-layer Network

Training a 9-layer network with [50; 30; 20; 20; 10; 10; 10; 10] hidden nodes, lambda=0.005, eta_min=1e-5, eta_max=1e-1, cycles=2, n_batch=100, n_s=5*45000/n_batch.

3.1 Without batch normalization

accuracy_validation = 47.1%
accuracy_test = 45.18%

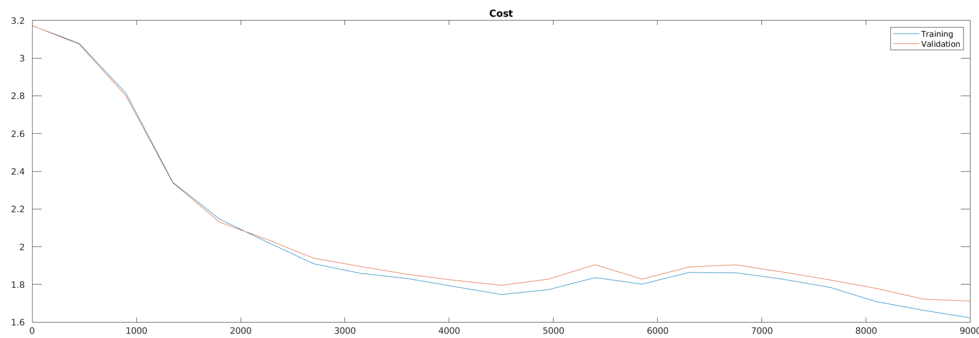

Figure 3: Loss, 9-layer without batch normalization

3.2 With batch normalization

accuracy_validation = 52.3%
accuracy_test = 51.6%

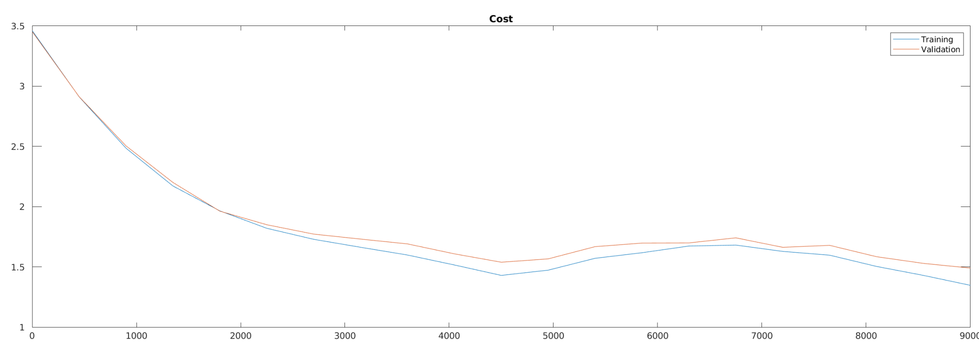

Figure 4: Loss, 9-layer with batch normalization

4 Optimizing 3-layer network

To optimize I first performed a coarse lambda search over a uniform grid between `lambda=1e-5` and `lambda=1e-1`. I sampled 20 lambdas in the interval and trained for 2 cycles. This resulted in a best result (based on validation accuracy) at `lambda=0.00527` and `accuracy=55.5%`. Then I performed a finer, random search between `lambda=0.00001` and `lambda=0.01`. I sampled 20 lambdas randomly and trained for 2 cycles. This resulted in a best result (based on validation accuracy) at `lambda=0.00564` and `accuracy=55.6%`. Then I trained the network with this lambda for 3 cycles, which resulted in a test accuracy of 54.27% and a validation accuracy of 56.1%.

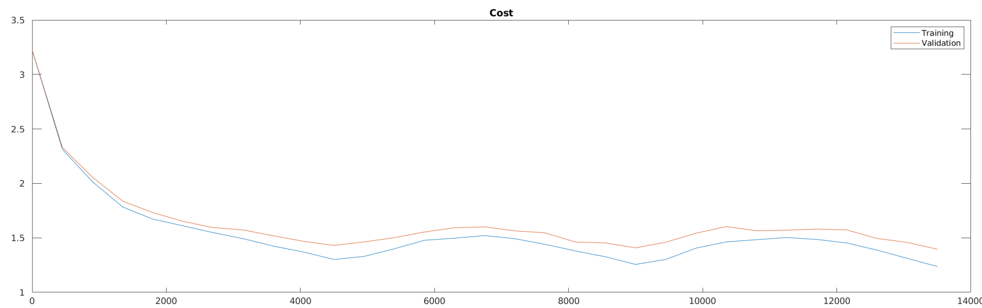

Figure 5: Loss, optimized 3-layer with batch normalization

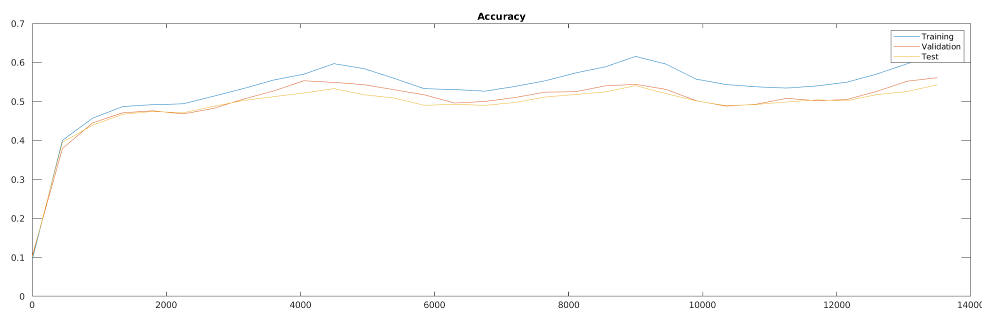

Figure 6: Accuracy, optimized 3-layer with batch normalization

5 Sensitivity to initialization

Instead of He initialization initialize the weights of the network with a normal distribution with the same sigma on each layer. Test this for 3 different sigmas.

5.1 sig=1e-1

5.1.1 Without batch normalization

accuracy_validation = 53.2%
accuracy_test = 53.24%

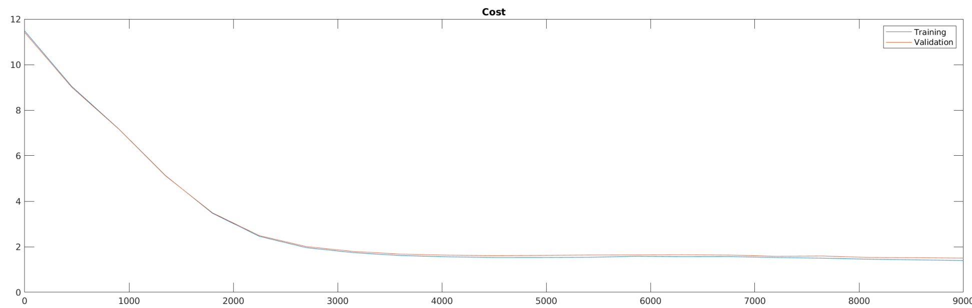

Figure 7: Loss, sig=1e-1, 3-layer without batch normalization

5.1.2 With batch normalization

accuracy_validation = 53.7%
accuracy_test = 53.49%

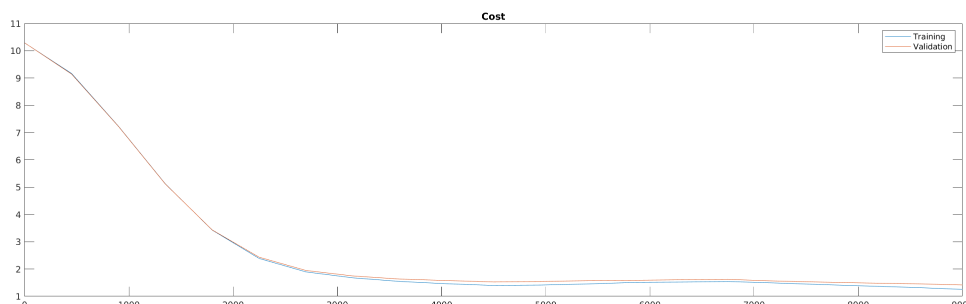

Figure 8: Loss, sig=1e-1, 3-layer with batch normalization

5.2 sig=1e-3

5.2.1 Without batch normalization

accuracy_validation = 50.6%
accuracy_test = 50.29%

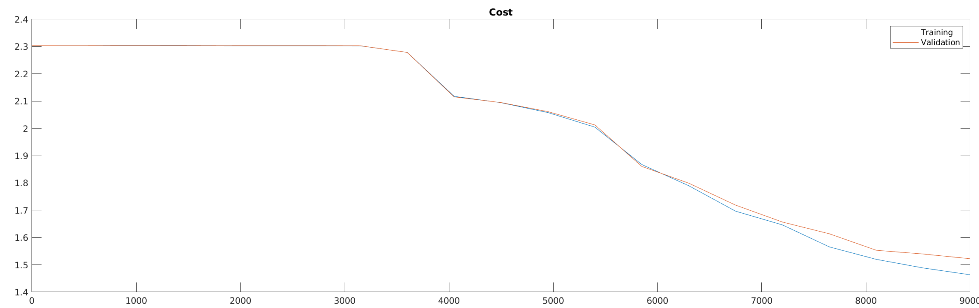

Figure 9: Loss, sig=1e-3, 3-layer without batch normalization

5.2.2 With batch normalization

accuracy_validation = 54.9%
accuracy_test = 53.88%

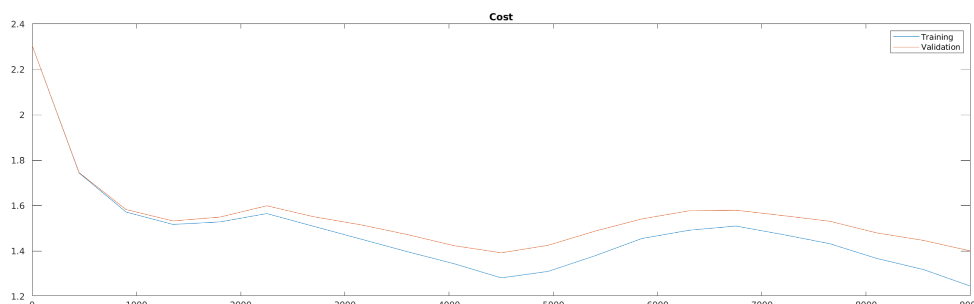

Figure 10: Loss, sig=1e-3, 3-layer with batch normalization

5.3 sig=1e-4

5.3.1 Without batch normalization

accuracy_validation = 7.8%
accuracy_test = 10%

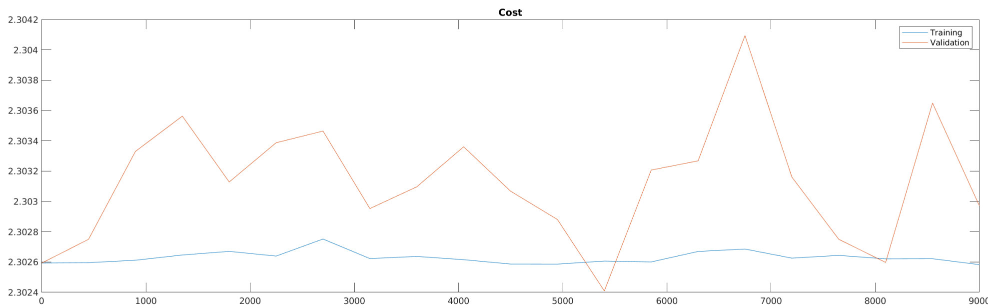

Figure 11: Loss, sig=1e-4, 3-layer without batch normalization

5.3.2 With batch normalization

accuracy_validation = 55.2%
accuracy_test = 53.44%

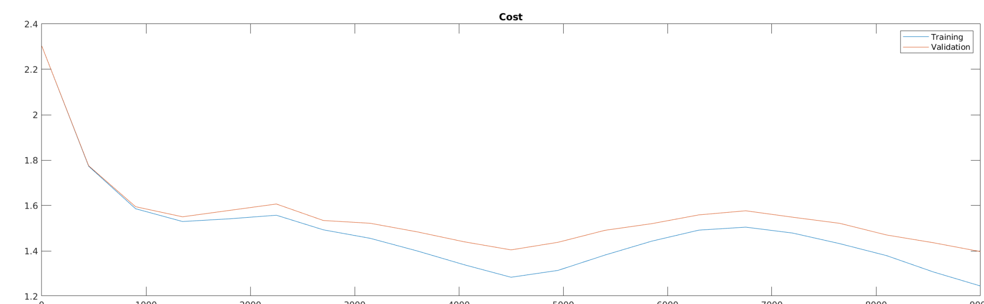

Figure 12: Loss, sig=1e-4, 3-layer with batch normalization

5.4 Conclusion

From the experiment we can see that batch normalization makes the training much more stable. The effect is best visible in the experiment with sig=1e-4. The initialization is very bad for training and without batch normalization the network achieves a very low test accuracy of only 10%. If we use batch normalization we achieve 53.44%, which is (almost) the same test accuracy as the optimized network which used a good initialization (Xavier).