Assignment 2: Two layer network - Bonus

DD2424 - Deep Learning
Svenja Räther

March 2020

1 Introduction

The bonus part of assignment 2 consists of two parts. Part 1 tries to optimize the performance of the network. To achieve this the following measures were implemented in the following order: 1b (The effect of more hidden nodes), 1e (Data augmentation), and 1c (Dropout). Part 2 refers to the paper [1] *Cyclical learning rates for training neural networks*. The task is to find good values for eta_min and eta_max for the network with a different number of hidden nodes and regularization.

2 Bonus 1: Optimize the performance

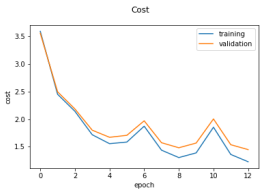
2.1 1b) Exploring the impact of having more hidden layers

To explore the effect of an increasing number of hidden nodes, the best performing setup from the basic assignment is chosen. Those has the following parameters: *lambda = 0.003389296561437l645, eta_min = 1e-5, eta_max = 1e-1, n_batches = 100, epochs = 12.*

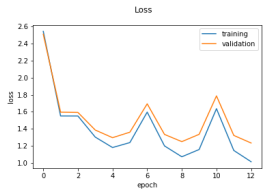
Table 1 shows the accuracy values for the increased number of hidden nodes. More nodes seem to increase accuracy. However, at some point, the increase becomes very small compared to the computational effort needed. This can be seen by comparing the values for m=2000 which results in a Validation accuracy of *57.2%* and Test accuracy of *56.4%* and m=300 which has a Validation accuracy of *57%* and Test the accuracy of *55.66%*.
The lecture notes state that even though the network has a lot more features than training examples, it most often does not overfit when increasing the number of nodes. Figure 2 even shows very close curves for the cost values of Train and Validation.

The theory in the lecture also states that when no regularization is applied, the outcome is highly dependent on the initialization. The network size should

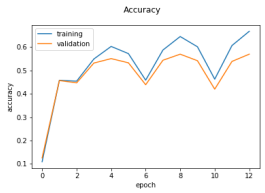
never be used for regularization. Therefore, table 2 examines the effect of different lambdas on those sets. Increasing lambda leads to Validation and Test values that are closer together as seen in figure 3 and table 2. However, the accuracy decreases for this setting. Decreasing lambda for a higher number of hidden nodes, as shown in figure 4, improves the values but also leads to a greater gap between Train and Test values which might indicate insufficient generalization.



(a) Cost for m = 300

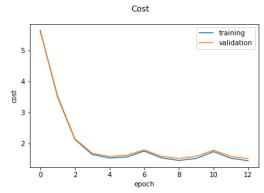


(b) Loss for m = 300

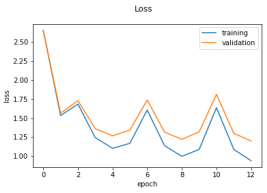


(c) Accuracy for m = 300

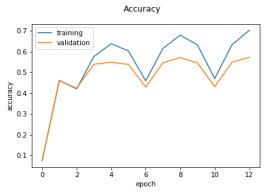
Figure 1: Plots for m = 300, lambda = 0.0033892965614371645



(a) Cost for m = 2000

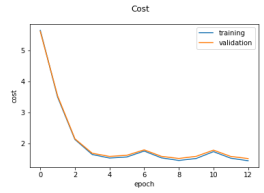


(b) Loss for m = 2000

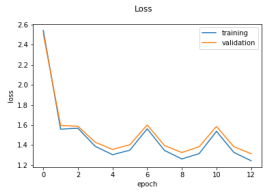


(c) Accuracy for m = 2000

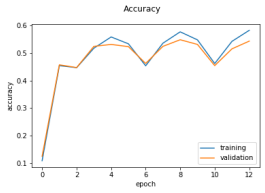
Figure 2: Plots for m = 2000, lambda = 0.0033892965614371645



(a) Cost for m = 300, lambda = 0.01

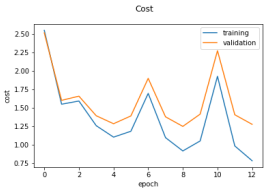


(b) Loss for m = 300, lambda = 0.01

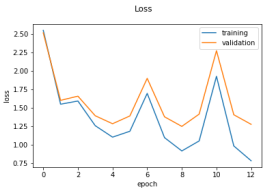


(c) Accuracy for m = 300, lambda = 0.01

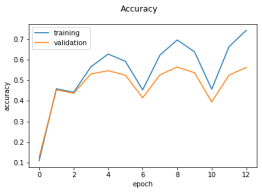
Figure 3: Plots for m = 300, lambda = 0.01



(a) Cost for m = 300, lambda = 0



(b) Loss for m = 300, lambda = 0



(c) Accuracy form =300, lambda = 0

Figure 4: Plots for m = 300, lambda = 0

m	Acc. (Train)	Acc. (Validate)	Acc. (Test)
50	0.5857	0.536	0.5192
100	0.6254	0.539	0.5409
200	0.6539	0.5502	0.553
300	**0.6680**	**0.57**	**0.5566**
400	0.6738	0.566	0.5566
500	0.6806	0.572	0.5558
600	0.6842	0.571	0.56
700	0.6868	0.576	0.5609
800	**0.6896**	**0.58**	**0.5577**
1000	0.6919	0.571	0.5593
2000	**0.702**	**0.572**	**0.564**

Table 1: Bonus 1b

m	lambda	Acc. (Train)	Acc. (Validate)	Acc. (Test)
300	0.01	0.5822	0.543	0.5353
300	0.003389	0.6680	0.57	0.5566
300	0.005	0.64048	0.557	0.5507
300	0	0.7434	0.563 .	0.5514

Table 2: Bonus 1b

2.2 1e) Augment your training data by applying a small random geometric and photometric jitter

Augmenting data is an approach to improving generalization error. Jitter is often used in terms of noise that is added to data. Too little might not affect while too much makes it too challenging to learn. [1]

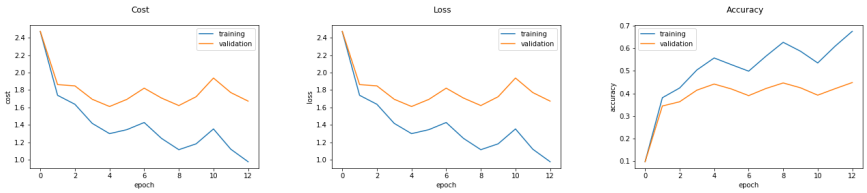
In this example, augmentation is added during the training of the model. Similar to the examples in [2] and [3], some rows and columns of the picture may be randomly eliminated and random rotation of maximum 20 degrees may

be applied.

All tests were run on the same number of hidden nodes, which is the initial standard value of 50 and lambda = 0. In the first try, only one batch of training data was used and the results were compared. The second try was applied to all data except 5000 for Validation. For both setups, the examples with jitter show lower accuracy on Train but better results on Test. Thus, jitter helped to achieve a more general model. The numbers are shown in table 3. Figures 8 and 5 show the graphs for the first try with and without jitter.

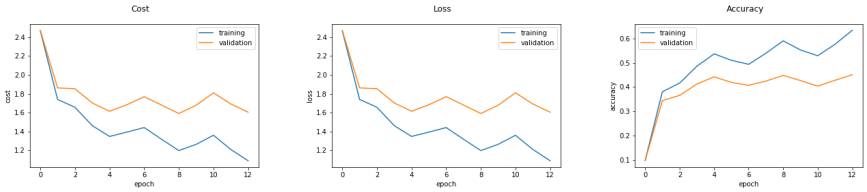
Data Tr.	**Jitter**	**Acc. Tr.**	**Acc. Val.**	**Acc. Te.**	**Cost Tr.**	**Cost Val.**
10000	False	0.6756	0.448	0.4467	0.977	1.673
10000	True	0.6339	0.4509	0.4612	1.089	1.604
45000	False	0.6108	0.5202	0.5104	1.386	1.116
45000	True	0.5938	0.5326	0.518	1.359	1.17

Table 3: Bonus 1e



(a) Cost (b) Loss (c) Accuracy

Figure 5: Plots without jitter for one batch, m=50



(a) Cost (b) Loss (c) Accuracy

Figure 6: Plots jitter for one batch, m=50

2.3 1c) Apply dropout

In Deep Learning the model can overfit even if there are far more examples than features. The drop out methods drops out some nodes during the training of the model. This is done to prevent the neural network from depending too much on any exact activation pathway. The remaining values are debiased and learning is applied. [4,5]

The implementation followed the given examples in [4] and [5] and was applied to one batch of data training for *20 epochs on 300 hidden nodes and lambda = 0*. Dropout is only applied during training. The final values for Test and Validation are acquired on a dropout probability of 0. The resulting values are shown in table 4.

It is visible that the version without dropout (probability = 0) overfits heavier on the available training data with an accuracy *90.72%* compared to *78.84%* on Train. However, the test with applied dropout results in a better accuracy on Test with *48.16%* for dropout = 0.2 compared to *46.65%* for dropout = 0.

Table 5 shows the same setup applied to the entire dataset (45000 examples for Train and 5000 for Validation).

Drop prob.	**Acc. Tr.**	**Acc. Val.**	**Acc. Te.**	**Cost Tr.**	**Cost Val.**
0	0.9072	0.4625	0.4665	0.3822	1.8498
0.2	0.7884	0.4717	0.4816	0.6672	1.7959

Table 4: Bonus 1c on one batch

Drop prob.	**Acc. Tr.**	**Acc. Val.**	**Acc. Te.**	**Cost Tr.**	**Cost Val.**
0	0.8130	0.5516	0.5447	0.5868	1.4066
0.2	0.6971	0.5648	0.5542	0.8913	1.3899

Table 5: Bonus 1c on 45000 examples for training

2.4 Best accuracy

From the knowledge gained form test results in this section, the parameters were set to the following: *m = 300, lam = 0.0033892965614371645, jitter = True, drop_prob = 0.2, epochs = 20, Training examples = 45000.* For this, a Test accuracy of *55.15%* and a Validation accuracy of *55.94%* is achieved.

For *m = 300, lam = 0.0033892965614371645, jitter = True, drop_prob = 0, epochs = 12, Training examples = 45000* Test accuracy of ***55.74%*** and a Validation accuracy of ***56.24%*** is achieved.

Regarding those results, it has to be noted that no intensive random search on the optimal setting for the parameter was done as suggested in bonus 1a (not implemented). Getting the right parameters would probably lead to better values.

The biggest jump in accuracy was made by increasing the number of hidden nodes. However, the augmentation only applies an example of what could be done. Different methods could have other effects and maybe also lead to a larger improvement.

Dropout improved the results in 1c when lambda = 0. For the combined test, where lambda = 0.0033892965614371645 the accuracy better when dropout is not applied.

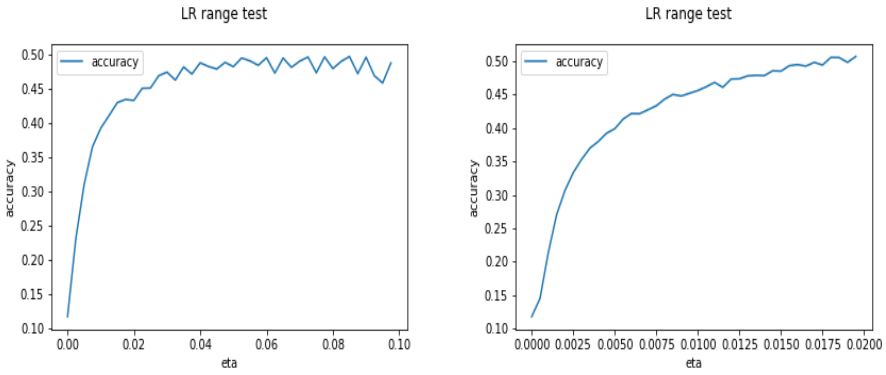
3 Bonus 2

Bonus 2 requires to adapt eta_min and eta_max by applying the method described in the paper *Cyclical Learning Rates for Training Neural Networks* by Leslie N. Smith [6]. An "LR range test" is done to estimate the reasonable minimum and maximum boundary values. Within one training run of the network for a few epochs, the learning rate is linearly increased. To find good boundaries the paper suggests to *"plot the accuracy versus learning rate. Note the learning rate value when the accuracy starts to increase and when the accuracy slows, becomes ragged, or starts to fall. These two learning rates are good choices for bounds".*

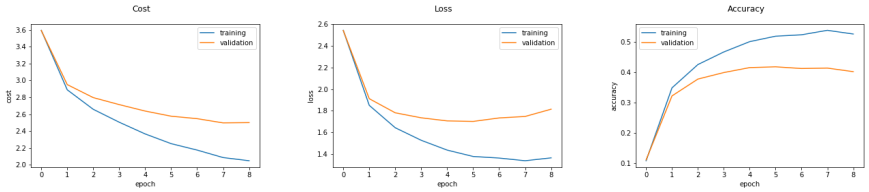
Figure 3 shows those plots for *m = 300, lam = 0.0033892965614371645, jitter = True, drop_prob = 0, epochs = 8, and 45000 Training examples.* Accuracy was plotted 5 times per epoch. Figure 7a shows the plot between *eta_min = 1e-06 to eta_max = 0.1* while figure 7b elaborates the area between *eta_min = 1e-06 to eta_max = 0.02.*

A slow down in accuracy can be seen around 0.016 for figure 7a and 0.0055 for 7b. Trying to run each boundary on the "best performing" setup mentioned

before results in the following values shown in table 6. Other than expected, the accuracy is worse compared to the initial range.



(a) Range test 1e-06 to 0.1 (b) Range test 1e-06 to 0.02



(a) Cost (b) Loss (c) Accuracy

Figure 8: Plots from LR range test 1e-06 to 0.1

Range	Acc. Tr.	Acc. Val.	Acc. Te.	Cost Tr.	Cost Val.
1e-06, 0.016	0.5801	0.5322	0.5195	1.8841	2.0092
1e-06, 0.0055	0.5086	0.485	0.4804	2.3253	2.3882

Table 6: Bonus 2 results on new ranges

4 References

[1] https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/

[2] https://srome.github.io/Jitter,-Convolutional-Neural-Networks,-and-a-Kaggle-Framework/

[3] https://scipy-lectures.org/advanced/image$_p$*rocessing/index.html*

[4] https://gluon.mxnet.io/chapter03$_d$*eep−neural−networks/mlp−dropout−scratch.html*

[5] https://d2l.ai/chapter$_m$*ultilayer − perceptrons/dropout.html*

[6] https://arxiv.org/abs/1506.01186