KTH Royal Institute of Technology

DD2424 Deep Learning in Data Science

Assignment 3

Name: Polyxeni Ioannidou
E-mail: ppio@kth.se

December 19, 2019

Introduction

In this assignment it has been built a multi-layer network to classify images from the CIFAR-10 dataset. The model has been trained with the Mini-Batch Gradient Descent with cyclical learning rates. We used the CIFAR-10 dataset, which consists of 60000 32x32 colour images in 10 classes. The dataset is divided into five training batches and one test batch, each with 10000 images. Our model has been trained by using data from data_batch_1 as our training set and data from data_batch_2 as validation set. Moreover, a cost function that computes the weighted sum of the cross-entropy loss is implemented and L2 regularization of the weight matrices using mini-batch gradient descent is also being applied. For computing and testing the gradients the relative error between the numerical and analytical gradients' values has been calculated. Last thing that should be mentioned is that all functions are implemented in Python 3.7, as well as there has been no use of libraries with already build-in networks. Google Colab is also the environment that has been used for testing the model. You can find the code here: `https://github.com/Xenia-Io/DD2424-Deep_Learning/blob/master/Assignment_3/multi_layer.ipynb`

Exercise 1: Build the Model and Check the Gradients

The model is a multi-layer neural network. We built a 3-layer and a 9-layer network. The purpose of this assignment is to use Batch Normalization, especially when we have many layers - 3 or more - so as to increase the speed of training the network and to achieve higher learning rates. Moreover, Batch normalization helps us to initialize the weights easily, which is an important advantage, as the way we initialize the parameters of the network affects our training procedure and our convergence rate.

The algorithm of Mini-Batch Gradient Descent with cyclical learning rates has been implemented to train the model. In mini-batch with GD, we computed and checked gradients analytically and numerically. For computing and testing my gradients the relative error between the numerical and analytical gradients' values has been calculated. More specifically, we got the following results for a 3-layer and a 9-layer network:

The 3-layers network
Network has the following structure:
Layer 0 = 'shape': (50, 30), 'activation': 'relu'
Layer 1 = 'shape': (50, 50), 'activation': 'relu'
Layer 2 = 'shape': (10, 50), 'activation': 'softmax'

Checking gradients for 3-layers network
The relative error for layer 1 W: 1.98168
The relative error for layer 1 b: 7.10538e-06
The relative error for layer 1 gamma: 1.32576
The relative error for layer 1 beta: 1.18741
The relative error for layer 2 W: 1.9871
The relative error for layer 2 b: 9.10375e-06
The relative error for layer 2 gamma: 0.000107843
The relative error for layer 2 beta: 9.47161e-07
The relative error for layer 3 W: 2.68847e-05
The relative error for layer 3 b: 3.2725e-08
The relative error for layer 3 gamma: 0

The relative error for layer 3 beta: 0

The 9-layers network

Network has the following structure:
Layer 0 = 'shape': (50, 30), 'activation': 'relu'
Layer 1 = 'shape': (30, 50), 'activation': 'relu'
Layer 2 = 'shape': (20, 30), 'activation': 'relu'
Layer 3 = 'shape': (20, 20), 'activation': 'relu'
Layer 4 = 'shape': (10, 20), 'activation': 'relu'
Layer 5 = 'shape': (10, 10), 'activation': 'relu'
Layer 6 = 'shape': (10, 10), 'activation': 'relu'
Layer 7 = 'shape': (10, 10), 'activation': 'relu'
Layer 8 = 'shape': (10, 10), 'activation': 'softmax'

Checking gradients for 9-layers network

The relative error for layer 1 W: 1
The relative error for layer 1 b: 1
The relative error for layer 1 gamma: 1
The relative error for layer 1 beta: 1
The relative error for layer 2 W: 1
The relative error for layer 2 b: 1
The relative error for layer 2 gamma: 1
The relative error for layer 2 beta: 1
The relative error for layer 3 W: 1
The relative error for layer 3 b: 1
The relative error for layer 3 gamma: 1
The relative error for layer 3 beta: 1
The relative error for layer 4 W: 1
The relative error for layer 4 b: 1
The relative error for layer 4 gamma: 1
The relative error for layer 4 beta: 1
The relative error for layer 5 W: 1
The relative error for layer 5 b: 1
The relative error for layer 5 gamma: 1
The relative error for layer 5 beta: 1
The relative error for layer 6 W: 1
The relative error for layer 6 b: 1
The relative error for layer 6 gamma: 1
The relative error for layer 6 beta: 1
The relative error for layer 7 W: 1
The relative error for layer 7 b: 1
The relative error for layer 7 gamma: 0.0309776
The relative error for layer 7 beta: 1
The relative error for layer 8 W: 0.107812
The relative error for layer 8 b: 0.788373
The relative error for layer 8 gamma: 6.98896e-14
The relative error for layer 8 beta: 1.19962
The relative error for layer 9 W: 5.63434e-13
The relative error for layer 9 b: 1.6127e-08
The relative error for layer 9 gamma: 0

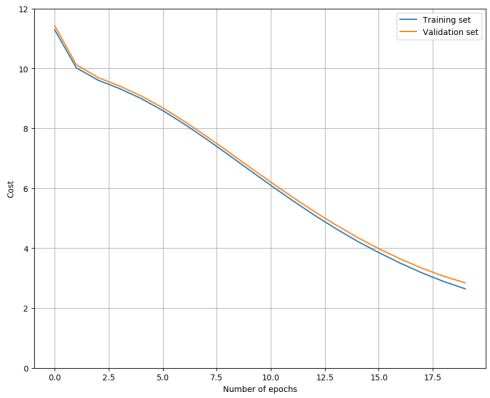
The relative error for layer 9 beta: 0
From these results we can observe that the relative errors are becoming smaller and smaller from one layer to another one.

- In case of 3-layer network:
  - W decreases from 1.98168 in layer 1 to 2.68847e-05 in layer 3
  - b values decreases from 7.10538e-06 in layer 1 to 3.2725e-08 in layer 3
  - gamma and beta values are becoming zero
- In case of 9-layer network:
  - W decreases from 1 in layer 1 to 5.63434e-13 in layer 9
  - b values decreases from 1 in layer 1 to 1.6127e-08 in layer 9
  - gamma and beta values are becoming zero

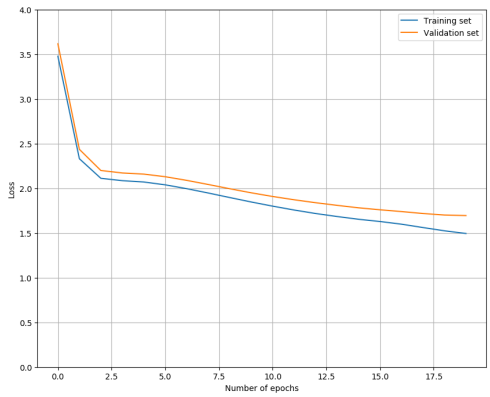
Exercise 2: Evolution of the loss for the 3-layer network

In this part, the loss function in the training and validation set is being presented. We used Batch normalization in a 3-layer network, as well as: learning_rate_min = 1e-5, learning_rate_max=1e-1, batch_size=100, stepsize=2000, labda=0.005, batch_size=100. Also, we ran this test for 20 epochs. As we can observe, the two plots are very similar and in both sets the loss and cost functions are gradually being decreased. Since validation set has almost the same losses as training set, batch normalization does not improve that much the performance of the network in this case.

With Batch Normalization



Costs



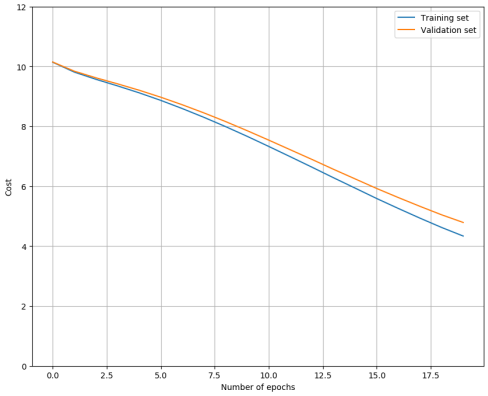
Loss

Below are the results -mean and standard deviation values- for the accuracy, after 10 iterations:

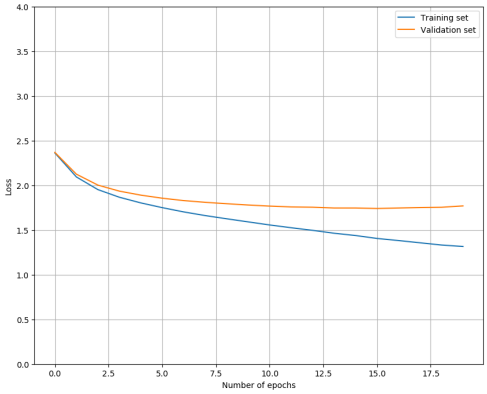
Train mean accuracy:0.42307
Val mean accuracy:0.37433
Test mean accuracy:0.38261
Train stdev accuracy:0.005728108084020608
Val stdev accuracy:0.004724181292974355

Test stdev accuracy:0.006099990892525069

Without Batch Normalization



Costs



Loss

Below are the results -mean and standard deviation values- for the accuracy, after 10 iterations:

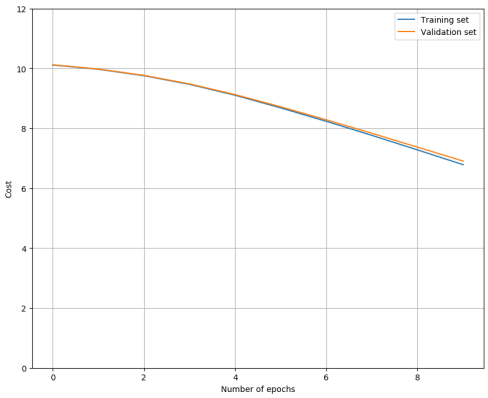
Train mean accuracy:0.42805
Val mean accuracy:0.36952
Test mean accuracy:0.37764
Train stdev accuracy:0.0038204857166479573
Val stdev accuracy:0.0035464536276868554
Test stdev accuracy:0.003487182880842995

Turning now to the comparison of the performance of the network with and without batch normalization, there are two important conclusions. The first thing, that i noticed, is that when BN is used, the loss function has a steeper slope, which means that it decreases instantaneously. Reading this paper[1], we can confirm that the plots are correct. As stated in page 6/26: "in a vanilla (non-BatchNorm), deep neural network, the loss function is not only non-convex but also tends to have a large number of flat regions", which is obvious also in the above plots. In other words, when we use BN we observe (more) convex loss function. The second thing that can be observed is that by using BN the test accuracy got increased.

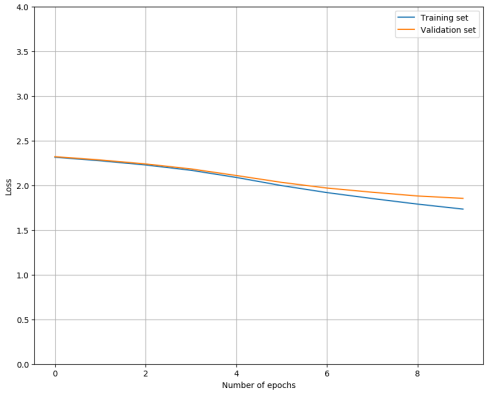
Exercise 3: Evolution of the loss for the 6-layer network

In this part, the loss function in the training and validation set is being presented. We used Batch normalization in a 6-layer network, as well as: learning_rate_min = 1e-5, learning_rate_max=1e-1, batch_size=100, stepsize=2250, labda=0.005, batch_size=100. We also ran this test for 10 epochs.

With Batch Normalization



Costs



Loss

Below are the results -mean and standard deviation values- for the accuracy, after 5 iterations:

Train mean accuracy:0.35748
Val mean accuracy:0.30956
Test mean accuracy:0.31762
Train stdev accuracy:0.012478661787227018
Val stdev accuracy:0.01297971494294076
Test stdev accuracy:0.01128259721872584

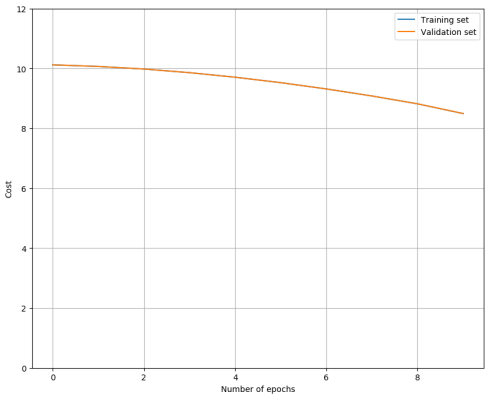
As we can observe, the losses and the costs for both sets are almost the same, they differ slightly. Below listed the original values for the cost and loss functions, in the last iteration, so as to prove that the values are extremely close, but not identical:

Validation cost = [10.122328712012228, 9.982859750841898, 9.770326734911459, 9.485649184451656, 9.128149634559646, 8.719925458498626, 8.28544557168494, 7.836351870310154, 7.370667756595916, 6.9073367972819355]
Training cost = [10.116172650537512, 9.97396910179847, 9.758910963846128, 9.470505211153391, 9.106542091105911, 8.683973013354763, 8.234566238668432, 7.765881184650086, 7.2796251978267605, 6.787290838734975]
Validation loss = [2.3204503241740153, 2.283515226380252, 2.2390939330213517, 2.183444253729028, 2.1097647619540947, 2.032901447405748, 1.9694621962609555, 1.9224496633096362, 1.8806609112643342, 1.8539512865267762]
Training loss = [2.3142942626992995, 2.274624577336824, 2.2276781619560224, 2.168300280430763, 2.088157218500359, 1.9969490022618848, 1.918582863244446, 1.8519789776495672, 1.7896183524951783, 1.7339053279798162]

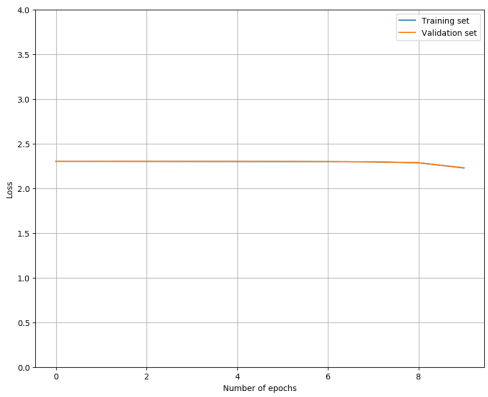
Without Batch Normalization

Below are the results -mean and standard deviation values- for the accuracy, after 5 iterations. It should be mention that by using BN we got more than 50% higher test accuracy: from 0.15345999 (without BN) to 0.31762 (with BN).

Train mean accuracy:0.15776
Val mean accuracy:0.14988
Test mean accuracy:0.15345999999999999
Train stdev accuracy:0.022679351842590206
Val stdev accuracy:0.022349429522920714
Test stdev accuracy:0.02153805933690405



Costs



Loss

Below listed the original values for the cost and loss functions, in the last iteration, so as to prove that the values are extremely close, but not identical.

Validation cost = [10.120870866583058, 10.068232230072837, 9.981058821330883, 9.860587122484093, 9.708512491038443, 9.526704198018969, 9.31727656620934, 9.082058776913902, 8.81986709187131, 8.495925460723848]
Training cost = [10.12096925159041, 10.068279055532063, 9.981015660438107, 9.860427719023706, 9.708234440762139, 9.526322799436514, 9.316802383493208, 9.08148961600517, 8.819167806702058, 8.493868702994801]
Validation loss = [2.302340312724654, 2.3020907080072326, 2.30171804608807, 2.301253166545005, 2.3007433503655474, 2.3000043680466553, 2.2987028522625463, 2.295799463594474, 2.28651760101392, 2.22934323769825]
Training loss = [2.302438697732005, 2.3021375334664578, 2.3016748851952937, 2.301093763084617, 2.3004653000892437, 2.2996229694642, 2.298228669546414, 2.2952303026857415, 2.2858183158446677, 2.227286479969203]

As a result, **using batch normalization in deeper neural networks has a considerable impact in the performance** by increasing the accuracy and decreasing the losses.

Exercise 4a: Coarse search for lambda in a 3-layer network

For this part, we sampled 30 values from a uniform distribution in between (1e-5, 1e-1) and then we ran the training algorithm for 2 cycles (20 epochs). Moreover, one batch of the training data was used. The rest of the parameters are similar to the ones in exercise 3. Below are the results:

Lambda = 0.02318054800591446
The accuracy on the training set is: 0.4874
The accuracy on the validation set is: 0.3991
The accuracy on the testing set is: 0.4047

Lambda = 0.05485790522463447
The accuracy on the training set is: 0.4331
The accuracy on the validation set is: 0.3785
The accuracy on the testing set is: 0.3829

Lambda = 0.07466974549549742
The accuracy on the training set is: 0.4023
The accuracy on the validation set is: 0.3609
The accuracy on the testing set is: 0.3667

Lambda = 0.08306798249735785
The accuracy on the training set is: 0.3896
The accuracy on the validation set is: 0.3546
The accuracy on the testing set is: 0.3628

Lambda = 0.0736403289378706
The accuracy on the training set is: 0.4098
The accuracy on the validation set is: 0.3717
The accuracy on the testing set is: 0.3778

Lambda = 0.006429665177647067
The accuracy on the training set is: 0.5845
The accuracy on the validation set is: 0.4059
The accuracy on the testing set is: 0.4144

Lambda = 0.0444296977002321
The accuracy on the training set is: 0.4393
The accuracy on the validation set is: 0.3753
The accuracy on the testing set is: 0.3847

Lambda = 0.0738664107944698
The accuracy on the training set is: 0.3959
The accuracy on the validation set is: 0.3532
The accuracy on the testing set is: 0.3658

Lambda = 0.01097758523512749
The accuracy on the training set is: 0.5206
The accuracy on the validation set is: 0.379

The accuracy on the testing set is: 0.3928

Lambda = 0.05353209148436174
The accuracy on the training set is: 0.4078
The accuracy on the validation set is: 0.3611
The accuracy on the testing set is: 0.3717

Lambda = 0.05285753282532107
The accuracy on the training set is: 0.409
The accuracy on the validation set is: 0.3611
The accuracy on the testing set is: 0.3669

Lambda = 0.0807817910835351
The accuracy on the training set is: 0.3884
The accuracy on the validation set is: 0.3566
The accuracy on the testing set is: 0.3644

Lambda = 0.04901270050636096
The accuracy on the training set is: 0.4262
The accuracy on the validation set is: 0.3819
The accuracy on the testing set is: 0.3791

Lambda = 0.020692016158355954
The accuracy on the training set is: 0.5138
The accuracy on the validation set is: 0.3986
The accuracy on the testing set is: 0.406

Lambda = 0.06014991013300423
The accuracy on the training set is: 0.417
The accuracy on the validation set is: 0.3705
The accuracy on the testing set is: 0.3786

Lambda = 0.0022197340791143545
The accuracy on the training set is: 0.6038
The accuracy on the validation set is: 0.3943
The accuracy on the testing set is: 0.3903

Lambda = 0.08193482429990154
The accuracy on the training set is: 0.3847
The accuracy on the validation set is: 0.3513
The accuracy on the testing set is: 0.3565

Lambda = 0.05183183554652871
The accuracy on the training set is: 0.415
The accuracy on the validation set is: 0.3734
The accuracy on the testing set is: 0.3795

Lambda = 0.09953013065635974
The accuracy on the training set is: 0.3672

The accuracy on the validation set is: 0.3394
The accuracy on the testing set is: 0.3434

Lambda = 0.009136998779990392
The accuracy on the training set is: 0.5435
The accuracy on the validation set is: 0.3918
The accuracy on the testing set is: 0.4012

Lambda = 0.08695084343328083
The accuracy on the training set is: 0.3909
The accuracy on the validation set is: 0.3593
The accuracy on the testing set is: 0.3632

Lambda = 0.045528409565712295
The accuracy on the training set is: 0.4252
The accuracy on the validation set is: 0.3771
The accuracy on the testing set is: 0.3844

Lambda = 0.0943375319707304
The accuracy on the training set is: 0.3609
The accuracy on the validation set is: 0.3292
The accuracy on the testing set is: 0.3328

Lambda = 0.07375014727699207
The accuracy on the training set is: 0.3868
The accuracy on the validation set is: 0.3525
The accuracy on the testing set is: 0.3678

Lambda = 0.06791126607181909
The accuracy on the training set is: 0.3955
The accuracy on the validation set is: 0.3617
The accuracy on the testing set is: 0.3674

Lambda = 0.059710903909561915
The accuracy on the training set is: 0.4042
The accuracy on the validation set is: 0.36
The accuracy on the testing set is: 0.3666

Lambda = 0.004426396662766733
The accuracy on the training set is: 0.5893
The accuracy on the validation set is: 0.3997
The accuracy on the testing set is: 0.4021

Lambda = 0.05570809583826468
The accuracy on the training set is: 0.4176
The accuracy on the validation set is: 0.3705
The accuracy on the testing set is: 0.3826

Lambda = 0.06658510827534417

The accuracy on the training set is: 0.3997
The accuracy on the validation set is: 0.3633
The accuracy on the testing set is: 0.3684

Lambda = 0.012615537494586293
The accuracy on the training set is: 0.5387
The accuracy on the validation set is: 0.4054
The accuracy on the testing set is: 0.4121

Exercise 4b: Fine search for lambda in a 3-layer network

The best results based on validation accuracy were for lambda equal to 0.01261 and 0.0064. So based on these two values, now we repeat the search to a more narrower range of (0.01, 0.1). We also sample 20 values this time. From the results below, we choose $\lambda = 0.021$, because it gives us

Lambda	Training accuracy	Validation accuracy	Test accuracy
0.028	0.4514	0.3943	0.4029
0.0147	0.5308	0.4018	0.4129
0.083	0.4439	0.3863	0.3942
0.0588	0.4754	0.3965	0.4006
0.096	0.4393	0.3805	0.3907
0.0629	0.4733	0.3914	0.397
0.09188	0.431	0.3694	0.3749
0.0477	0.5023	0.3932	0.4051
0.0277	0.4549	0.388	0.3976
0.05866	0.4875	0.3952	0.4018
0.021	0.5786	0.4163	0.4233
0.058	0.5047	0.4031	0.4117
0.0249	0.5736	0.4118	0.4205
0.0218	0.5727	0.3984	0.407
0.0778	0.473	0.3857	0.3996
0.01178	0.5825	0.4045	0.4116
0.032	0.5604	0.4017	0.4153
0.03579	0.5306	0.3996	0.4043
0.091	0.4451	0.3869	0.3883
0.023	0.5178	0.3956	0.4038

Table 1: Fine Search for the regularization parameter lambda

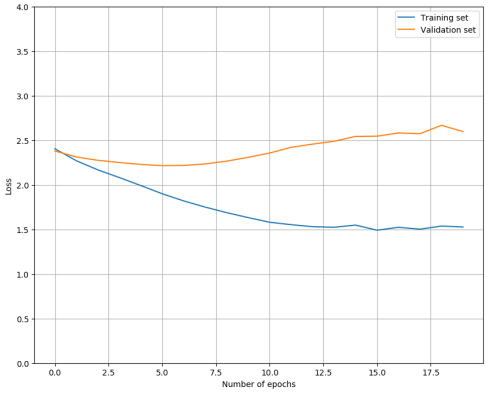
the highest value for the accuracy in the validation set. Moreover, it is interesting that it gives the best accuracy for the test set too. That is why splitting our dataset and using validation set for parameter tuning is an effective technique. Furthermore, it should be mentioned that in a more specific (narrow) area in this range we can observe almost the same performance for our model, as accuracies do not differ a lot in the validation set. Moreover, after using $\lambda = 0.021$ we obtained the following results after training the model for 20 epochs in 10 iterations:

Train mean accuracy:0.44812
Val mean accuracy:0.39065
Test mean accuracy:0.40093
Train stdev accuracy:0.005564730202105241
Val stdev accuracy:0.005501767392797016
Test stdev accuracy:0.0032176768707321275

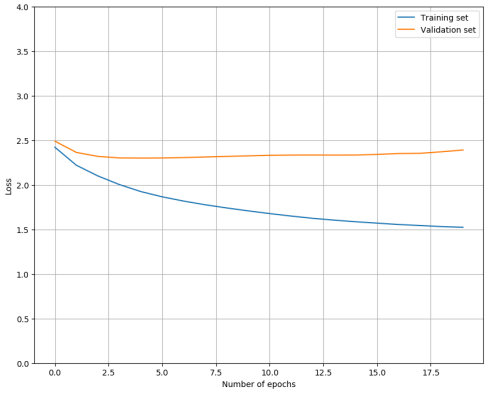
As we can observe, the mean test accuracy and the validation mean accuracy are both much higher than the ones we got in exercise 2 (Val mean accuracy: 0.37433 and Test mean accuracy: 0.38261). This means that we will use this lambda value from now on.

Exercise 5: Sensitivity to initialization in a 3-layer network

Sigma = 1e-1



3-layer with BN, sigma=0.1

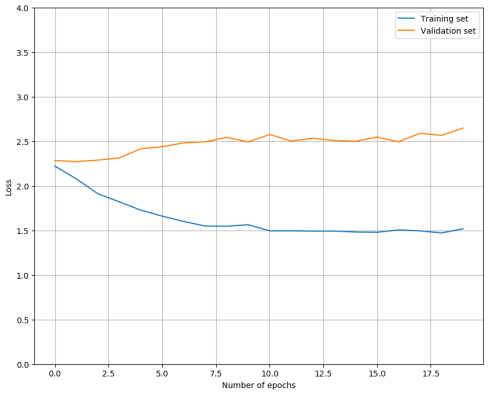


3-layer without BN, sigma=0.1

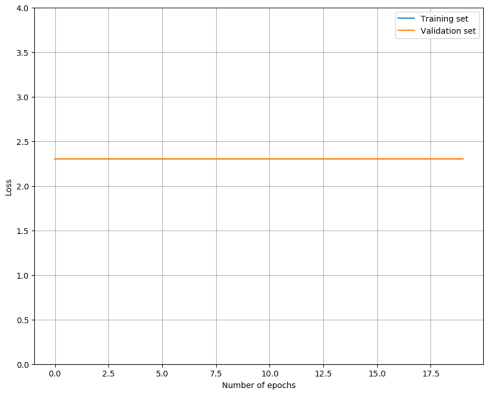
Running Model with Batch normalization:
The accuracy on the training set is: 0.4542
The accuracy on the validation set is: 0.1544
The accuracy on the testing set is: 0.1492

Running Model without Batch normalization:
The accuracy on the training set is: 0.4553
The accuracy on the validation set is: 0.1379
The accuracy on the testing set is: 0.1456

Sigma = 1e-3



3-layer with BN, sigma=0.001

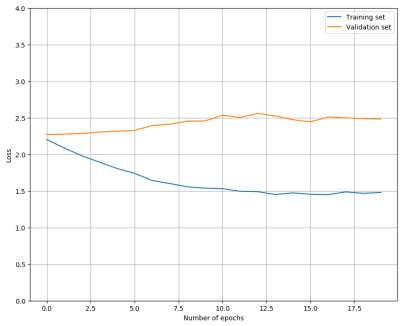


3-layer without BN, sigma=0.001

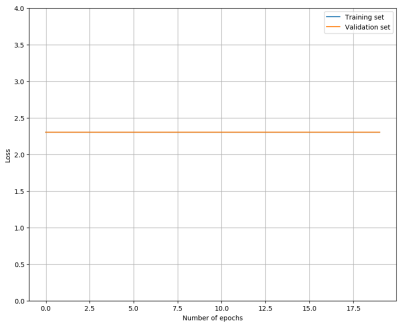
Running Model with Batch normalization:
The accuracy on the training set is: 0.463
The accuracy on the validation set is: 0.1406
The accuracy on the testing set is: 0.1423

Running Model without Batch normalization:
The accuracy on the training set is: 0.1032
The accuracy on the validation set is: 0.101
The accuracy on the testing set is: 0.1

Sigma = 1e-5



3-layer with BN, sigma=0.00001



3-layer without BN, sigma=0.00001

Running Model with Batch normalization:
The accuracy on the training set is: 0.4767
The accuracy on the validation set is: 0.1684
The accuracy on the testing set is: 0.1453

Running Model without Batch normalization:
The accuracy on the training set is: 0.1032
The accuracy on the validation set is: 0.101
The accuracy on the testing set is: 0.1

From these results, we can observe that using BN in deep neural networks makes a great impact in the performance, since we have better accuracy results. Moreover, when sigma is getting lower and lower the network we have a flat loss function which means that the model does not learn anything, whereas by adding BN we can see that the model tries to learn.

References

[1] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.