Deep Learning in Data Science - Assignment 3

Fernando García Sanz

April 21, 2020

Abstract

The scope of this assignment consists in building and training a k-layer network that implements batch normalization and has multiple outputs. This network is used to classify the images contained in the CIFAR-10 [1] dataset. The network is trained using mini-batch gradient descent over a cost function which computes the cross-entropy loss of the classifier applied to the labelled training data and an L_2 regularization term over the weight matrix.

1 Comparison of analytical and numerical gradients

To compare the obtained results with the analytical and numerical gradients, small batches and a reduced dimensionality of the samples have been used to allow computing the results in a reasonable amount of time. The performed tests are the following:

	Analytical - Slow		
Weight Gradient (W_1, W_2, W_3)	1.400e-11	1.371e-11	1.410e-11
Bias Gradient (B_1, B_2, B_3)	2.220e-12	3.106e-18	1.534e-11

	Analytical - Slow	
Gamma Gradient (γ_1, γ_2)	1.552e-11	1.410e-11
Beta Gradient (β_1, β_2)	1.349e-11	1.528e-11

Table 1: Difference between analytical and numerical calculations of gradients initializing the network with seed 42 over the 1000 first samples and employing 100 features.

	Analytical - Slow		
Weight Gradient (W_1, W_2, W_3)	6.901e-08	8.901e-08	1.395e-11
Bias Gradient (B_1, B_2, B_3)	1.332e-12	2.840e-18	1.323e-11

	Analytical - Slow	
Gamma Gradient (γ_1, γ_2)	1.490e-11	1.492e-11
Beta Gradient (β_1, β_2)	1.512e-06	2.113e-07

Table 2: Difference between analytical and numerical calculations of gradients initializing the network with seed 8 over the 1000 first samples and employing 100 features.

	Analytical - Slow		
Weight Gradient (W_1, W_2, W_3)	1.414e-11	1.383e-11	1.380e-11
Bias Gradient (B_1, B_2, B_3)	3.553e-12	4.441e-13	1.466e-11

	Analytical - Slow	
Gamma Gradient (γ_1, γ_2)	1.371e-11	1.359e-11
Beta Gradient (β_1, β_2)	1.664e-11	1.235e-11

Table 3: Difference between analytical and numerical calculations of gradients initializing the network with seed 42 over 1000 random samples and employing 100 features.

	Analytical - Slow		
Weight Gradient (W_1, W_2, W_3)	3.471e-06	2.132e-08	1.374e-11
Bias Gradient (B_1, B_2, B_3)	8.882e-13	3.253e-18	1.631e-11

	Analytical - Slow	
Gamma Gradient (γ_1, γ_2)	1.588e-11	1.352e-11
Beta Gradient (β_1, β_2)	1.023e-06	1.420e-11

Table 4: Difference between analytical and numerical calculations of gradients initializing the network with seed 8 over 1000 random samples and employing 100 features.

The previous experiments prove that the differences between the analytical calculation and the numerical one are quite small, from the order of 10^{-6} as maximum. It is also necessary to remark that the discrepancy between the analytical and the numerical procedure sometimes increases once the calculations approach to the first layers; this is due to the back-propagation of the gradients.

Therefore, due to the small differences between both calculation approaches, it can be assumed that the analytical calculation of the gradients is well performed.

2 Comparison of a 3-layer network

A 3-layer network composed by 50 neurons in the two hidden layers and 10 neurons in the output layer is trained with the following parameters:

- Configuration: $\eta_{min} = 1e\text{-}5$, $\eta_{max} = 1e\text{-}1$, $\lambda = 0.005$, $n_s = 5 * 45000/n_{batch}$

The network, which has been initialized via *He* initialization, has been trained during two cycles and the results obtained, with and without batch normalization, are the following:

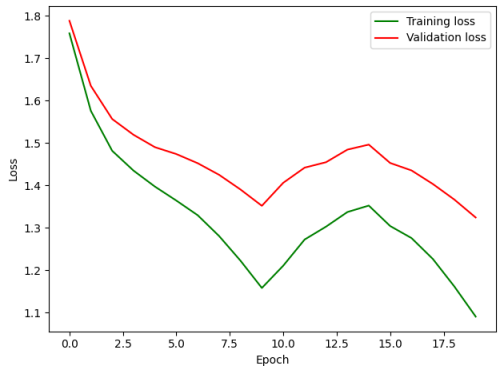

Figure 1: Cost over training epochs with batch normalization.

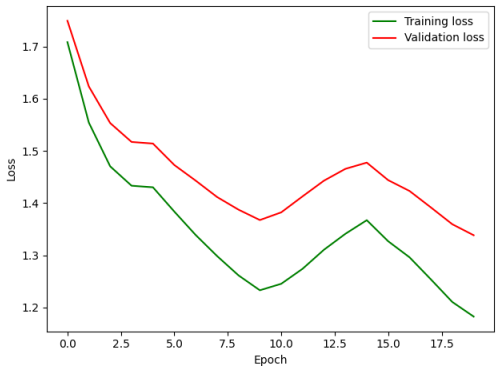

Figure 2: Loss over training epochs without batch normalization.

As can be seen, the use of batch normalization provides smoother shapes in the plot, as well as a slightly lower value of loss in the last epochs. It can be seen the more curvy shape of the first lines compared to the second one. Nevertheless, the results do not show a huge difference in this case due to the network structure, since having three layers is not a big problem to deal with when training the network.

3 Comparison of a 9-layer network

Now a comparison of performance between a 9-layer network is done. The architecture and configuration of the network are the following:

- Architecture: [50, 30, 20, 20, 10, 10, 10, 10, 10]

• Configuration: η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = 0.005$, $n_s = 5 * 45000/n_{batch}$

As in the previous case, the network has been initialized by means of *He* initialization and it has been trained during two cycles, with and without batch normalization, providing the following outcomes:

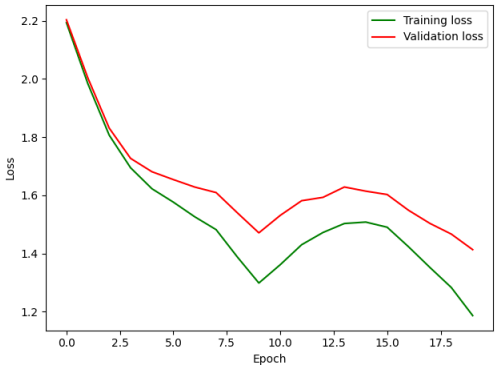

Figure 3: Cost over training epochs with batch normalization.

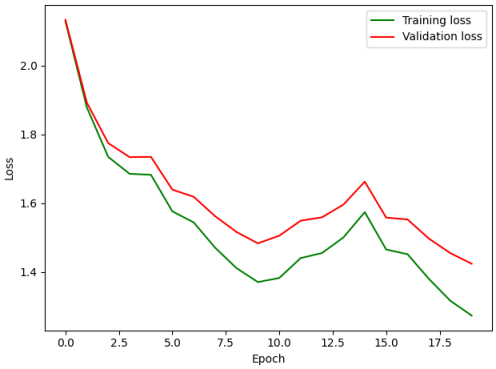

Figure 4: Loss over training epochs without batch normalization.

In this case is much more patent that batch normalization helps reducing the fluctuations in loss, providing a much smoother plot. The first one is quite similar to the batch normalization plot of the 3-layer network shown in the previous section. Nevertheless, it can be seen how the second one is more unsteady, with several peaks. Moreover, the loss value obtained during the last epochs in the one which employs batch normalization is lower than the one obtained in the other. This is the proof of that when using a deeper architecture batch normalization substantially improves the training process.

4 3-layer network with lambda search

It is possible enhancing the performance of the network by performing lambda search. To do so, first, lambda values are randomly selected from an interval defined by [-3, -1], which represents the values of lambda in $\log_{10}$ scale.

Once 10 values are retrieved, the two which provide the best two accuracies are used to define a more specific search. The distance between those two values is used to define a new interval; this interval is defined as [$best_\lambda$ - dist, $best_\lambda$ + dist], also in $\log_{10}$ scale. By means of this interval it is possible to look for values around the best ones provided by the previous search. Other 10 values were sampled from the interval.

The results obtained applying this procedure are the following:

- Coarse search:
  - Best accuracies in validation: [0.5394, 0.5392, 0.5392]
  - Best lambdas in validation ($\log_{10}$ scale): [-2.30937357, -2.33853783, -2.03849986]
- Fine search:
  - Improved accuracies in validation: [0.5404, 0.5394, 0.5392]
  - Improved lambdas in validation ($\log_{10}$ scale): [-2.3258368, -2.30937357, -2.33853783]

Training with the lambda value that provided the best accuracy result over the validation set, -2.3258368, and training during three cycles, the accuracy obtained over the test data has been equal to 53.82%, a value that outperforms the results obtained with the provided configuration in previous sections.

5 Experiment *"Sensitivity to initialization"*

This experiment consists of initializing the weights of our neural network by means of a normal distribution with different variances. The selected normal distribution has zero mean, and the sigma values employed have been 1e-1, 1e-3 and 1e-4. The configuration of the employed network, which only varies on whether employing or not batch normalization, is the following:

- Configuration: η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = 0.005$, $n_s = 5 * 45000/n_{batch}$

The obtained results are shown below:

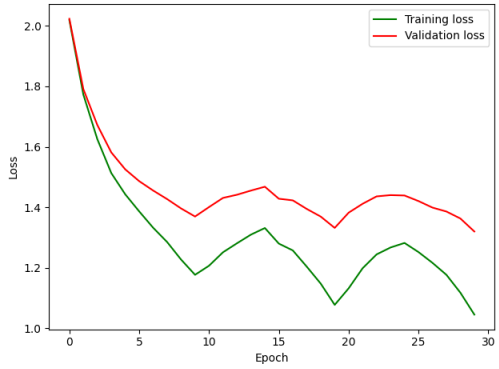

Figure 5: Cost over training epochs with batch normalization and σ=1e-1.

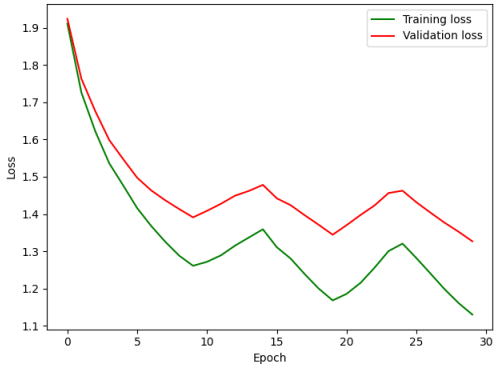

Figure 6: Cost over training epochs without batch normalization and σ=1e-1.

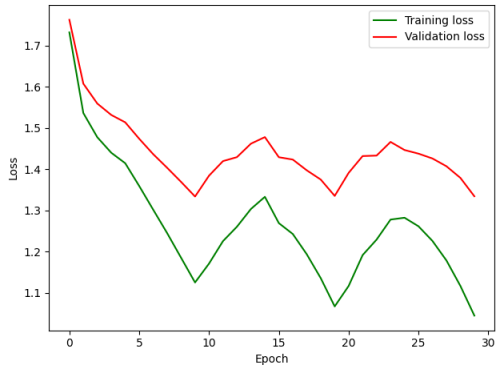

Figure 7: Cost over training epochs with batch normalization and σ=1e-3.

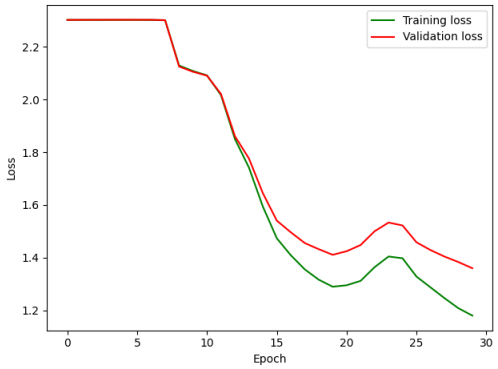

Figure 8: Cost over training epochs without batch normalization and σ=1e-3.

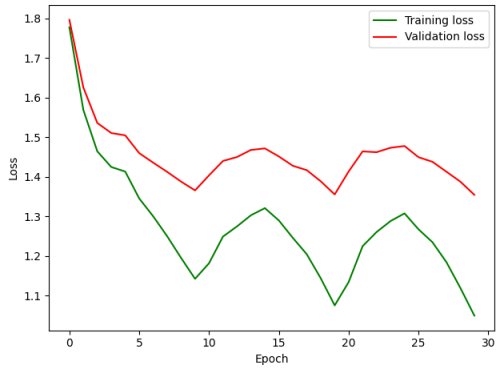

Figure 9: Cost over training epochs with batch normalization and σ=1e-4.

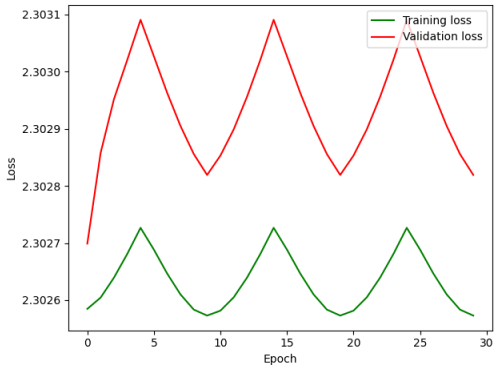

Figure 10: Cost over training epochs without batch normalization and σ=1e-4.

As can be seen, the value of sigma triggers important changes in the loss obtained during the training process. For the case which does not implement batch normalization, the smaller the sigma, the bigger the loss. During the entire experiment, the value of sigma eventually decreases, being quite small in the last case. Therefore, in this last experiment, the values used to initialize the weights are rather close to each other, implying this the impossibility of proper training for the network. Nevertheless, employing batch normalization importantly improves the results obtained. The retrieved loss barely changes, independently of the value of sigma used to initialize the weights.

References

[1] A. Krizhevsky. The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 21-03-2020.