Deep Learning in Data Science - Assignment 1

Fernando García Sanz

March 26, 2020

Abstract

The scope of this assignment consists in building and training a single layer network which 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 have been used to allow computing the results in a reasonable amount of time. The performed tests are the following:

	Analytical - Numerical	Analytical - Slow
Weight Gradient	-1.0445614490988497e-06	3.6930873220585263e-13
Bias Gradient	-4.414246746586165e-08	-6.7654215563095475e-18

Table 1: Difference between analytical and numerical calculations of gradients with generating bias and weights with seed 42 over the 20 first samples.

	Analytical - Numerical	Analytical - Slow
Weight Gradient	-1.0445693714870665e-06	8.025052580499346e-13
Bias Gradient	-4.467537450651444e-08	-1.3322675855315797e-10

Table 2: Difference between analytical and numerical calculations of gradients with generating bias and weights with seed 400 over the 20 first samples.

	Analytical - Numerical	Analytical - Slow
Weight Gradient	-1.043279170419492e-06	-1.2931366255743222e-12
Bias Gradient	-4.3787196094165205e-08	1.7763568099099513e-10

Table 3: Difference between analytical and numerical calculations of gradients with generating bias and weights with seed 42 over the 20 random samples.

	Analytical - Numerical	Analytical - Slow
Weight Gradient	-1.0446670364197038e-06	1.0916248975319378e-12
Bias Gradient	-4.4586556666051466e-08	-1.3322676006019898e-10

Table 4: Difference between analytical and numerical calculations of gradients with generating bias and weights with seed 400 over the 20 random samples.

As it can be seen, the differences between the analytically calculated and the numerical ones are quite small, from the order of 10^{-6}. Moreover, if we take into account that the slow calculation is more precise, and it is in this case where the differences are smaller, we can assume that the analytical calculation of the gradients is well performed.

2 Testing in different configurations

The built network has been tested with different configurations, providing these results:

- Configuration 1: $\lambda = 0$, number of epochs = 40, size of the mini-batch = 100, $\eta = 0.1$:

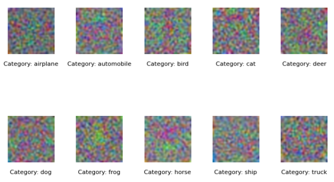

Figure 1: Weights after training.

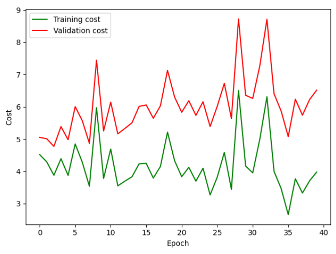

Figure 2: Cost over the training epochs.

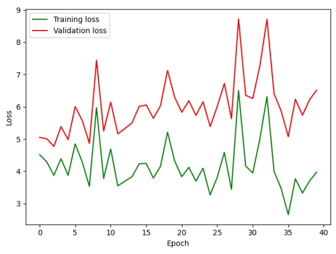

Figure 3: Loss over the training epochs.

As it can be seen, the weights obtained after the training process are very blurry when displayed. Cost and loss plots are equal since the λ value is equal to zero, so no regularization is performed. The shape of these plots presents a lot of peaks. This behaviour may be explained as a consequence of the big value assigned to η, which triggers taking bigger steps during the training process, causing this variation in both loss and cost over the epochs, being the values of the validation set bigger than the training ones, as expected. The accuracy obtained on test data after the training process is equal to 29.30%.

- Configuration 2: $\lambda = 0$, number of epochs = 40, size of the mini-batch = 100, $\eta = 0.001$:

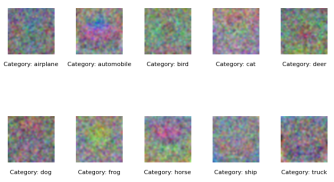

Figure 4: Weights after training.

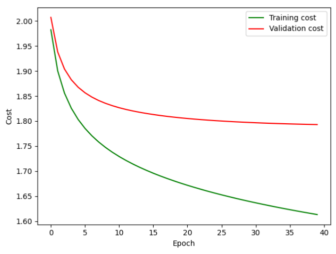

Figure 5: Cost over the training epochs.

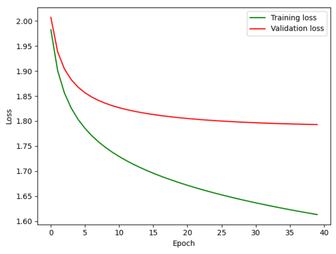

Figure 6: Loss over the training epochs.

Now it can be observed a big difference in all the plots compared to those obtained before. Weights representation shows more clear colors and some shapes start to appear. The cost and loss plots are still equal since λ is still zero, but the retrieved values are much smaller. These better results can be seen when performing the testing phase: the obtained accuracy is equal to 39.27%. Therefore, it can be seen that significantly reducing the value of η provides better results.

- Configuration 3: $\lambda = 0.1$, number of epochs = 40, size of the mini-batch = 100, $\eta = 0.001$:

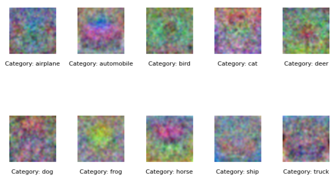

Figure 7: Weights after training.

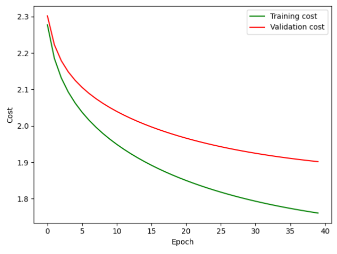

Figure 8: Cost over the training epochs.

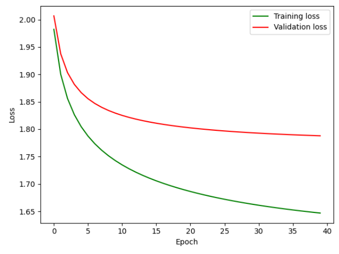

Figure 9: Loss over the training epochs.

With this configuration it can be seen that the weights displayed are even more clear than in the previous settings. Cost and loss plots display similar shapes to those of the configuration 2. Nevertheless, the plots are not equal now due to the use of regularization, which generates slightly bigger values compared to the one obtained before. The obtained accuracy over the test set is equal to 39.59%, very similar to the one of the previous configuration.

- Configuration 4: $\lambda = 1$, number of epochs = 40, size of the mini-batch = 100, $\eta = 0.001$:

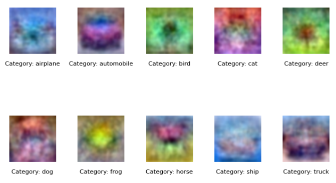

Figure 10: Weights after training.

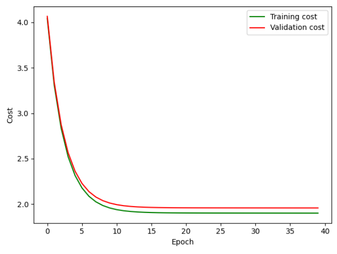

Figure 11: Cost over the training epochs.

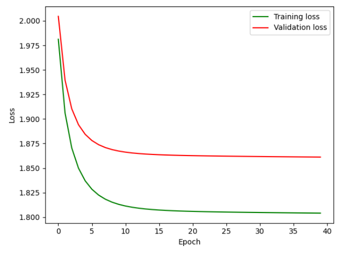

Figure 12: Loss over the training epochs.

In this last case an even bigger regularization term has been used, obtaining interesting results. It can be seen that the displayed weights are quite clear now, being able to see some clear shapes, mainly in the categories of *automobile* and *horse*, which clearly evoke to these items. The plotted values in cost and loss graphs are slightly bigger than those of the two previous configurations, but it can also be seen that the training process seems to converge around some values after 10-15 epochs, behaviour that could not be seen in the previous settings. This is due to the importance of regularization in this configuration, which makes the network focus only on the main features: this explains why the weights are more clear, convergence is achieved and cost and loss are bigger than in the previous configurations. The obtained accuracy over test data with these settings has been equal to 38.01%, also a bit lower than before.

2.1 Conclusion of the results

After analyzing the results obtained with each one of the different configurations, the following conclusions are obtained: a huge value of the learning rate η provokes huge oscillations in the cost and loss values. Due to this instability the generalization of the system is not good; the system has not been properly trained since the step taken when updating the weights is too big. When reducing this rate, the obtained results are much better; the training stage is more progressive, better tuning the weights over the iterations and, therefore, providing a better generalization. Now, according to the regularization, it has been seen that a small regularization factor λ can improve the generalization of the networks and the weights definition. Nevertheless, using a factor too big reduces the accuracy of the network over unseen data, probably because of an excessive reduction of the network complexity. It can also be seen that when a huge regularization factor is employed, the weights represent better the main features of the different categories when they are displayed.

References

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

[2] N. Pattanasri. Vectorized impl. of the svm loss and gradient update. `https://mlxai.github.io/2017/01/06/vectorized-implementation-of-svm-loss-and-gradient-update.html`. Accessed: 26-03-2020.