Deep Learning in Data Science - Assignment 1 - Bonus Tasks

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. Bonus tasks can be found in this document.

1 Optimize the performance of the network

The following three additional tasks have been implemented.

1.1 Use all datasets for training

Use all the available training data for training (all five batches minus a small subset of the training images for a validation set). Decrease the size of the validation set down to ~ 1000.

By means of this procedure, some interesting results have been obtained. As it can be seen, with the first configuration, the maximum loss obtained is lower than with the previous equivalent configuration. In this case, the size of the training set is much bigger than the validation one, so the loss in the validation set, as well as in the train set, is a bit smaller, since the model is trained with more data, but the difference is not significant. The same happens with the accuracy, it improves since more data samples are employed during training, but it only increases around a 1-2%, so the improvement is not that relevant according to the increment in the training set size. It is interesting to remark that in the last case, due to the big value of λ, added to the small size of the validation set, the loss obtained is smaller in validation than in training, although it is also true that the loss retrieved is slightly bigger as λ increases.

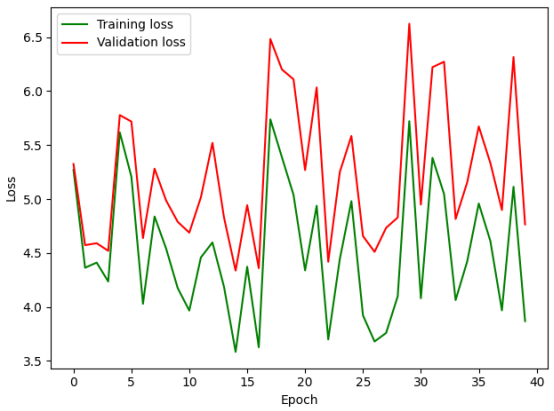

Figure 1: Loss config. 1.

Training with more data does not significantly enhance the performance of the this classifier.

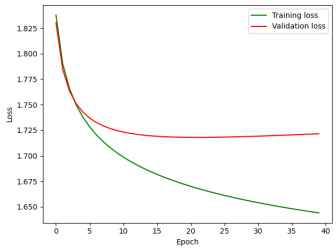

Figure 2: Loss config. 2.

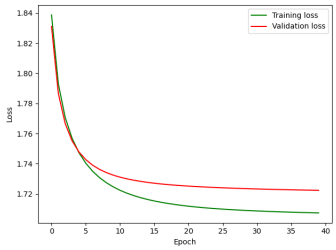

Figure 3: Loss config. 3.

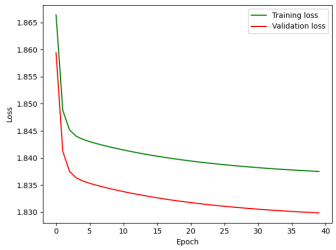

Figure 4: Loss config. 4.

1.2 Apply weight decay

Play around with decaying the learning rate by a factor ~ 0.9 after each epoch.

By means of weight decay, the loss obtained with the first configuration is much smoother, with almost no peaks and smaller values, enhancing the accuracy with these settings and obtained a small loss on the training data. Nevertheless, this change is not applicable to the other configurations, obtaining quite similar results to those aforementioned.

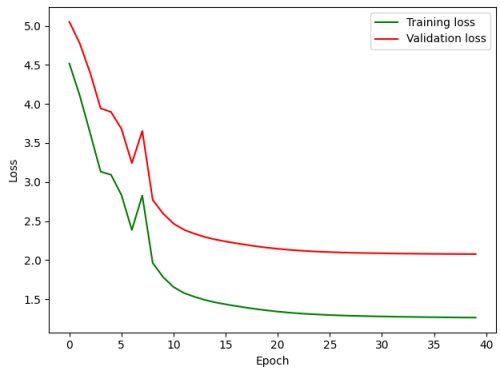

Figure 5: Loss config. 1

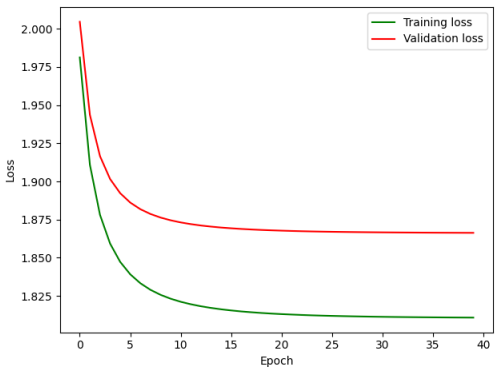

Figure 6: Loss config. 4.

1.3 Apply shuffling to the training samples

Shuffle the order of your training examples at the beginning of every epoch.

By means of this mechanism, the obtained results do not vary a lot from the configurations without this implementation, while it is true that the retrieved plots are not as smoother as those obtained before. The accuracy values obtained when classifying the test set are similar to those without this improvement and the training process takes more time due to the fact of shuffling the samples and labels each epoch.

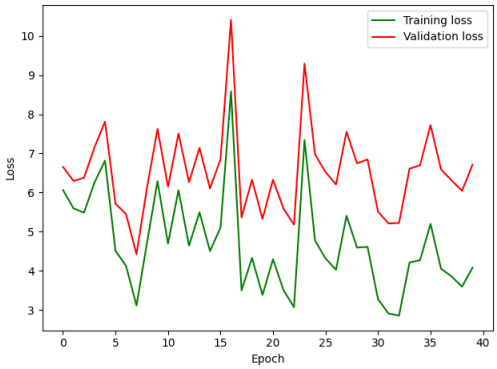

Figure 7: Loss config. 1

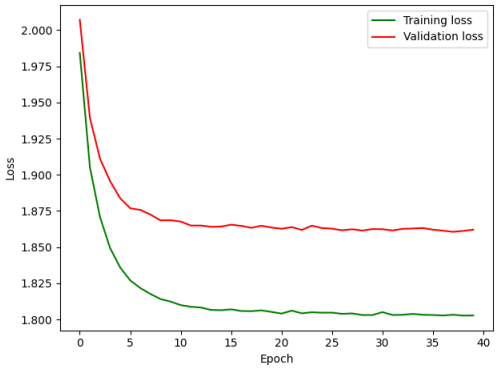

Figure 8: Loss config. 4.

As can be seen in these two plots, the shapes the lines define and their behaviour are almost identical to the ones without shuffling. In the first plot is not possible to state any main difference, but in the second one, the lines are not as smooth, as they present some small peaks. This is due to the effect of shuffling the data samples.

2 Train network by minimizing the SVM multi-class loss

To accomplish this task, it has been necessary to use a different loss function and, therefore, a different cost function. This also implies that the computed gradients are also different from the ones employed before. Applying this new network to the previous configurations has provided the following results:

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

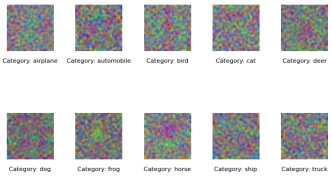

Figure 9: Weights after training.

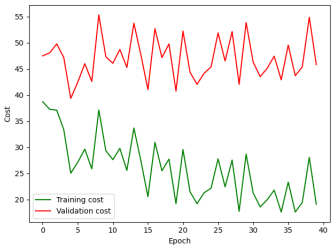

Figure 10: Cost over the training epochs.

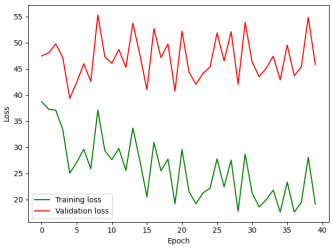

Figure 11: Loss over the training epochs.

In this case, the shapes of the plots are similar to the first configuration using the cross-entropy loss, but with much higher values. The weights are similar, quite blurred. The obtained accuracy over the test set is 28.92%.

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

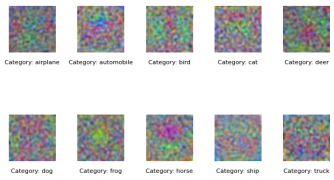

Figure 12: Weights after training.

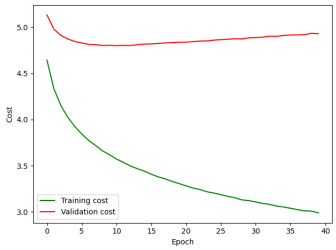

Figure 13: Cost over the training epochs.

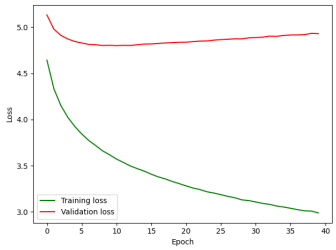

Figure 14: Loss over the training epochs.

Once a smaller η value is employed, the results are much better. It can be seen how the loss has significantly been reduced. Nonetheless, the weights do not show a great improvement compared to those of the previous configuration. When tested over the test set, the accuracy has been equal to 34.98%, almost a 5% worse than the one obtained with the cross-entropy method.

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

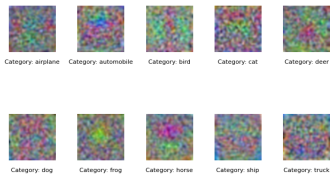

Figure 15: Weights after training.

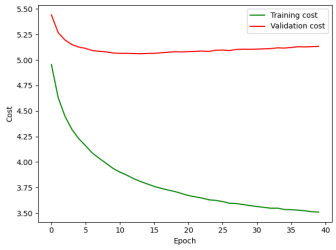

Figure 16: Cost over the training epochs.

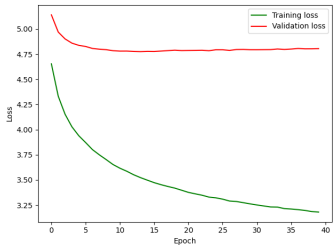

Figure 17: Loss over the training epochs.

When applying a λ value of 0.1, the cost plot shows slightly higher values, as expected, but an improvement can be seen in the weights, which now show a better defined colors. The accuracy over the test set has been equal to 35.81% in this case.

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

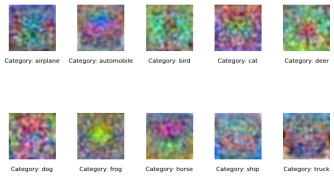

Figure 18: Weights after training.

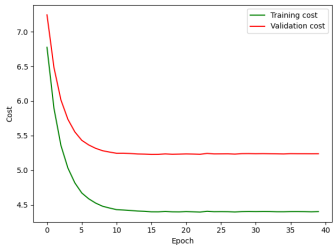

Figure 19: Cost over the training epochs.

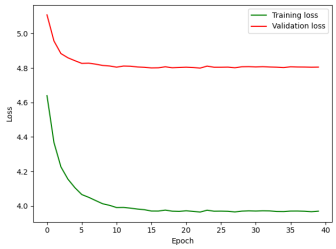

Figure 20: Loss over the training epochs.

Finally, employing a λ equal to 1, the convergence in the cost is reached around 10-15 epochs and the weights are better defined than in the other configurations. The retrieved accuracy is equal to 36.03%, also better than in the other cases.

It seems that employing SVM multi-class loss provides slightly worse results. Also, the weights are not as well defined as in the best ones obtained with the cross-entropy loss. Nevertheless, here, the use of λ different from zero provides better results, meanwhile in the cross-entropy one, a λ different from zero does not suppose big changes or even worsens the accuracy.

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.