Kungliga Tekniska Högskolan

DD2424 Deep Learning in Data Science
Assignment 2

Quentin LEMAIRE

May 17, 2018

1 Gradient validation

In order to validate my computation of the gradient, I compared my values with of the gradient with those of a numerically computed gradient function. To have a better estimate of the difference between both, I used the relative error. I decided to work on a subset of the data of 100 images to compute the gradients faster.

The difference with the numeric one after one epoch is very good for W2 and b2 but not that good for W1 and b1 (around 10^{-2}). However, after the second epoch, the results are very good. I got the following relative results for the second epoch:

- For grad W1: 2.1284063891132074e-07 and grad W2: 3.831611925597542e-08
- For grad b1: 4.992344709137718e-08 and grad b2: 1.4075725945565216e-06

The results are quite good. Those results were with the faster version of the numerically computed gradient thus less precise version. Therefore, I ran the same tests with the slower version to comfort my idea that it was right. I got the following results:

- For grad W1: 9.659418778649868e-08 and grad W2: 6.139451379078318e-09
- For grad b1: 1.9368577223303455e-08 and grad b2: 3.8762329077326405e-11

The results are even better, the relative error is very low. The result are similar for the following epochs. So even if it wasn't that good for the first epoch, we can consider it correct as it remains good after.

Finally, I tried to overfit the training data for a small subset of 200 images and for 200 epochs and a learning rate of 0.01, the loss on the training set went from 2.3 to 0.25. It is an other proof that the gradients work correctly.

2 Momentum term

Then, I added the momentum term to W1, W2, b1 and b2. I computed the same test of 200 epochs with a subset of images to see if I was converging faster toward 0.25 of loss.

- For a rho equal to 0.5, I had to wait 100 epochs instead of 200 to get a loss of 0.25 and I was at 0.02648724.
- For a rho equal to 0.90, I had to wait 22 epochs instead of 200 to get a loss of 0.25 and I was at 0.0016287.
- For a rho equal to 0.99, I had to wait 9 epochs instead of 200 but after that, the error increased and I got a NaN.

Therefore, we can see that the convergence is really faster with the momentum.

To illustrate this with some figures, I have run the same network for a rho = 0 (therefore, with no momentum) and for a rho = 0.9 (which seemed to be the best parameters in the 3 experiments I did before) and I plotted the validation and training cost evolution for these 2 possibilities.

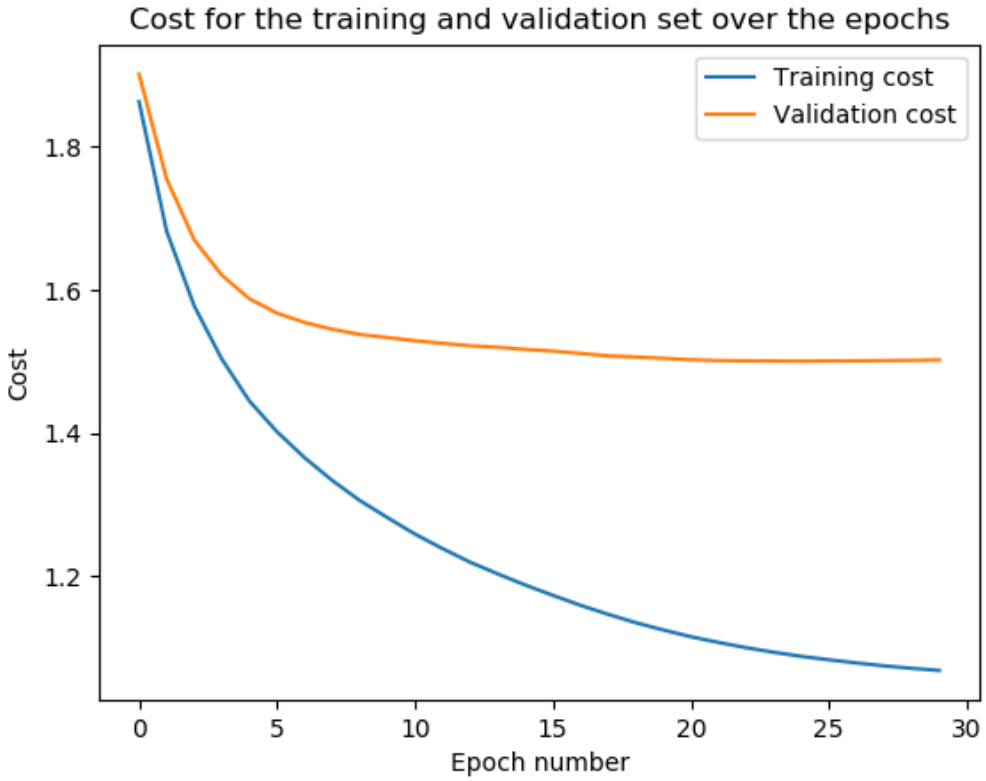

Figure 1: Validation and training loss for a rho of 0.9.

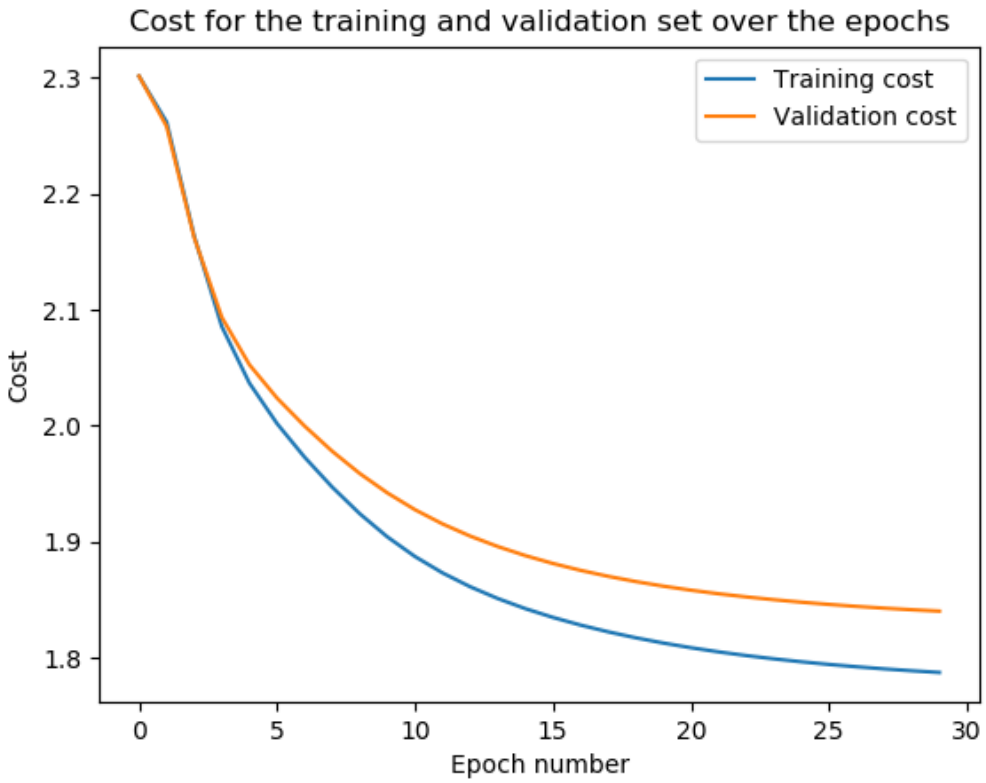

Figure 2: Validation and training loss for a rho of 0.

We can see that the momentum improves significantly the training. With the same number of epoch and the same hyper-parameters (except for rho), we are converging much more faster with the momentum and we are getting a loss significantly lower.

3 Coarse search

First, I search for hyper-parameters on the log scale. I tried values of eta between 0.9 and 0.00001 and lambda between 0.1 and 0.000001. For some of the values, the result is really bad, we can see in the following figure that the learning rate is too high in this case (for a lambda of 0.000001).

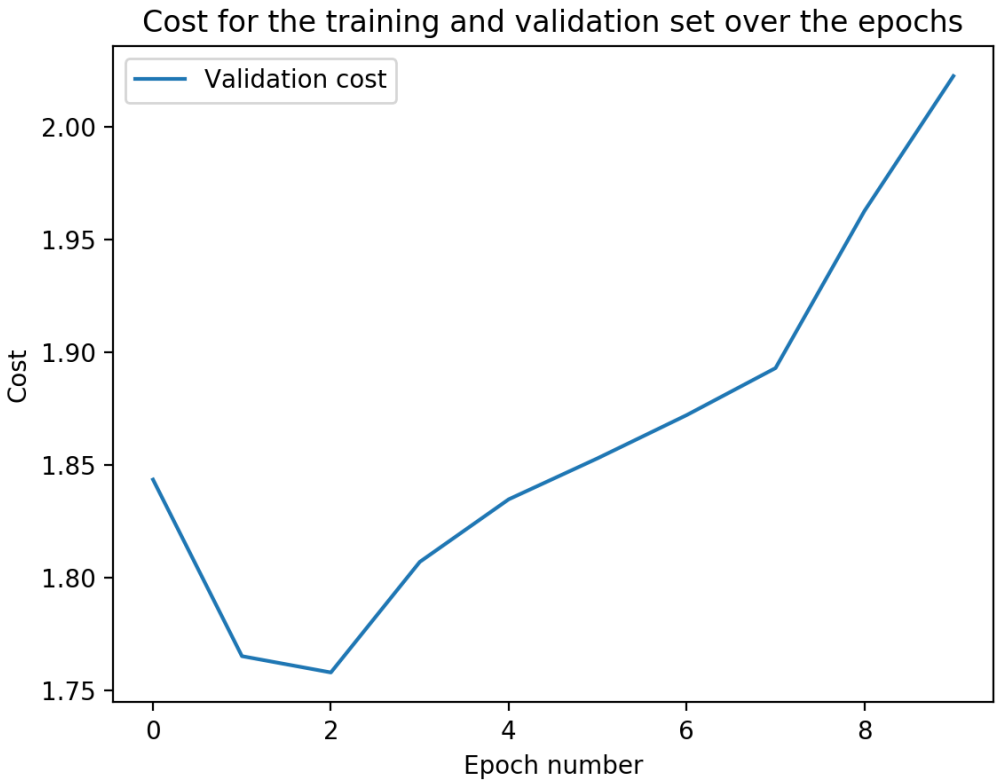

Figure 3: Validation loss for a 0.1 learning rate.

I found out that we can look for a learning rate between 0.1 and 0.001 and a lambda between 0 and 0.01. Those ranges gave correct results.

4 Fine search

I then did a random search between those borders. After the first 100 random searches, I got a result with a loss of 1.70207628 and a final accuracy of 0.4375. I also saw that 0.1 might be too high for the regularization term and that 0.001 might be too low for a learning rate.

Therefore, I decided to compute an other random search for eta between 0.1 and 0.005 and lambda between 0 and 0.005. After that, I did a new one with eta between 0.05 and 0.005 because it will too high.

Finally, I did a random search around precise values that I found during the last one. I put eta between 0.02 and 0.035 and lambda between 0 and 0.00001.

For my three best results I had:

- eta: 0.0148, lambda: 0.00023, cost: 1.63899456 and accuracy: 0.4375

- eta: 0.0239, lambda: 7.36e-06 cost: 1.61645057 and accuracy: 0.4458
- eta: 0.0218 lambda: 2.018e-06 cost: 1.61191793 and accuracy: 0.4453

Therefore, I am going to take eta = 0.024 and lambda = 7e-06 for the end of the assignment.

5 Best performances

As the error was increasing after just a few epochs (I chose my parameters according to 10 epochs), I put the weight decay factor to 0.9 instead of 0.95. I got a final error on the validation set of 1.52091212 and an accuracy of 0.4723.

Here is the figure with the validation loss and the training cost:

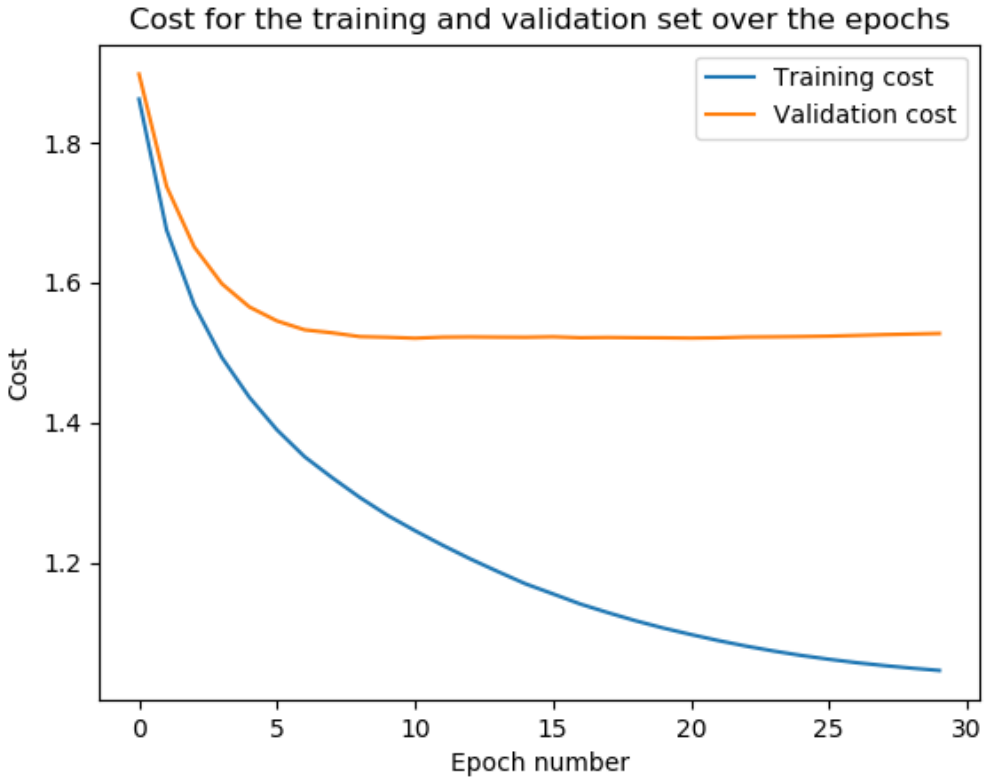

Figure 4: Training and validation cost.

Unfortunately, we can see that there is a big step between the training and validation cost. The is due to a big overfitting of the train set, it might be because 10 epochs on a subset wasn't enough to be able to see the overfitting in the results. I will try to deal with this problem and to find better parameters in the bonus part of this assignment.