Clément Gousseau

Deep Learning in Data Science: Assignment 1

1. Computation of the gradients

For the computation of the gradients I used the formulas presented in the lecture slides, that is to say:

1.1. With respect to b:

$$\frac{\partial J}{\partial b}(B, lambda, W, b) = \frac{1}{|X|}\sum_{(x,y)\in B}\frac{\partial l}{\partial b}(x, y, P, W)$$

$$with\ \frac{\partial l}{\partial b}(x, y, P, W) = -(y - P)\ and\ P = evaluateClassifier(x, W, b)$$

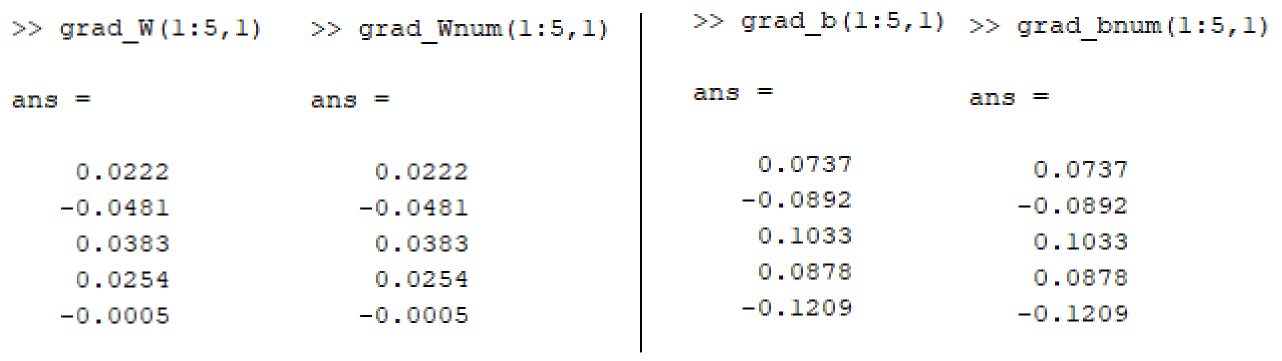
1.2. With respect to W:

$$\frac{\partial J}{\partial W}(B, lambda, W, b) = \frac{1}{|X|}\sum_{(x,y)\in B}\frac{\partial l}{\partial W}(x, y, P, W) + 2.lambda.W$$

$$with\ \frac{\partial l}{\partial W}(x, y, P, W) = -(y - P)x^T\ and\ P = evaluateClassifier(x, W, b)$$

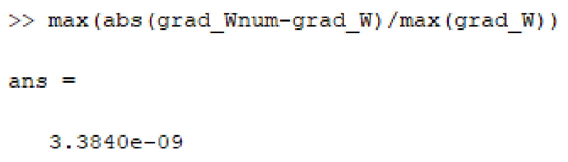
1.3. Tests

I checked the gradients with a sample of the datasets, with a random W and b, a lambda equal to 1. I computed the gradients with my function and the functions provided.

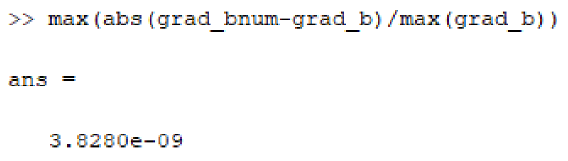


```
>> grad_W(1:5,1)     >> grad_Wnum(1:5,1)  |  >> grad_b(1:5,1)  >> grad_bnum(1:5,1)

ans =                ans =                |  ans =              ans =

    0.0222               0.0222           |      0.0737             0.0737
   -0.0481              -0.0481           |     -0.0892            -0.0892
    0.0383               0.0383           |      0.1033             0.1033
    0.0254               0.0254           |      0.0878             0.0878
   -0.0005              -0.0005           |     -0.1209            -0.1209
```

Let's compute the maximum relative error on the whole gradients:



```
>> max(abs(grad_Wnum-grad_W)/max(grad_W))

ans =

   3.3840e-09
```

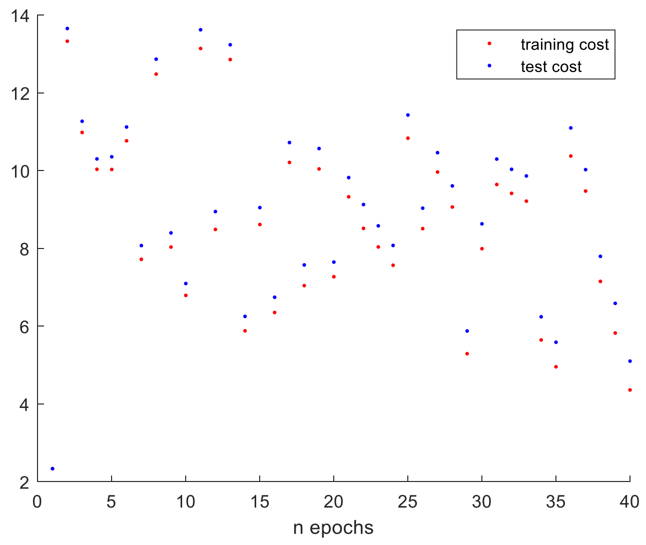


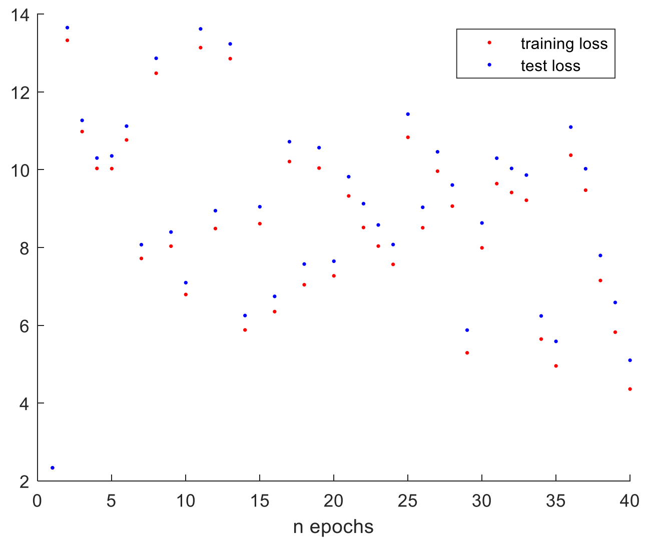
```
>> max(abs(grad_bnum-grad_b)/max(grad_b))

ans =

   3.8280e-09
```

This relative error is very small; we can consider the function computeGradients is acceptable to compute the gradients.

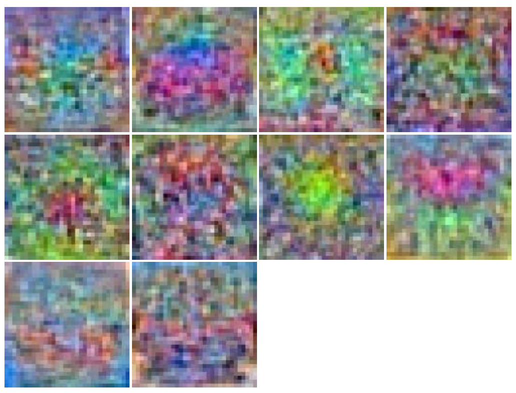
2. Results with different parameters settings

Lambda=0, n_epochs=40, n_batch=100, eta=0.1

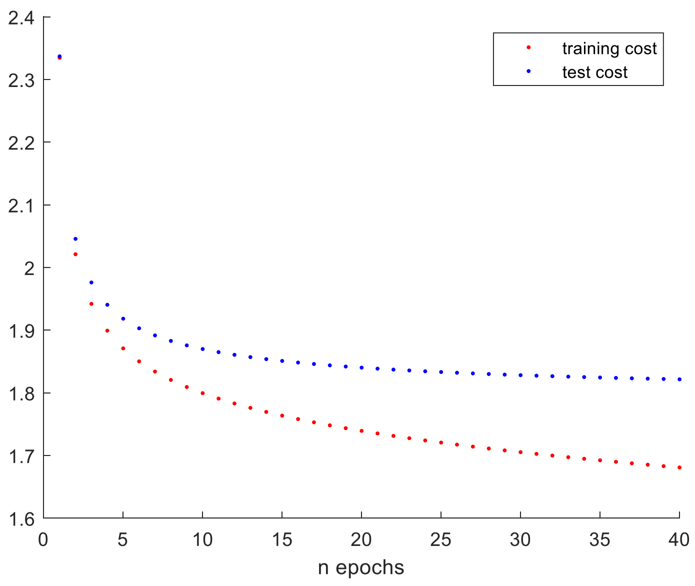


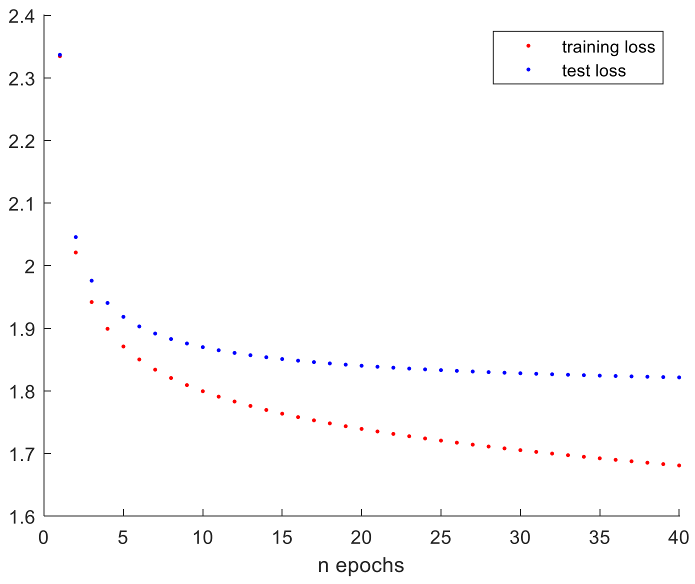


Accuracy: 16,39%

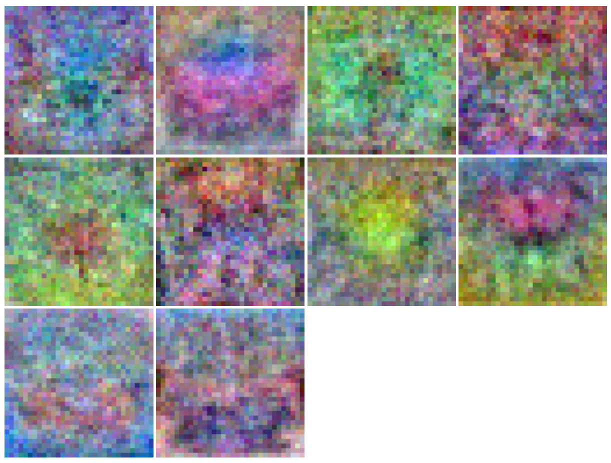


Lambda=0, n_epochs=40, n_batch=100, eta=0.01

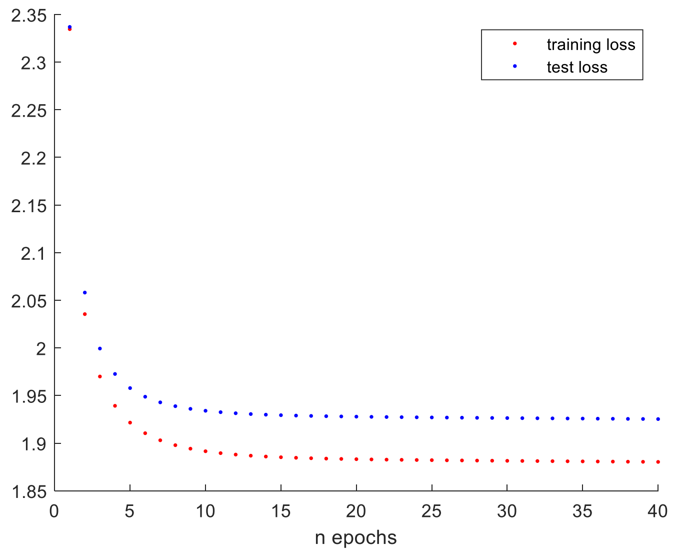


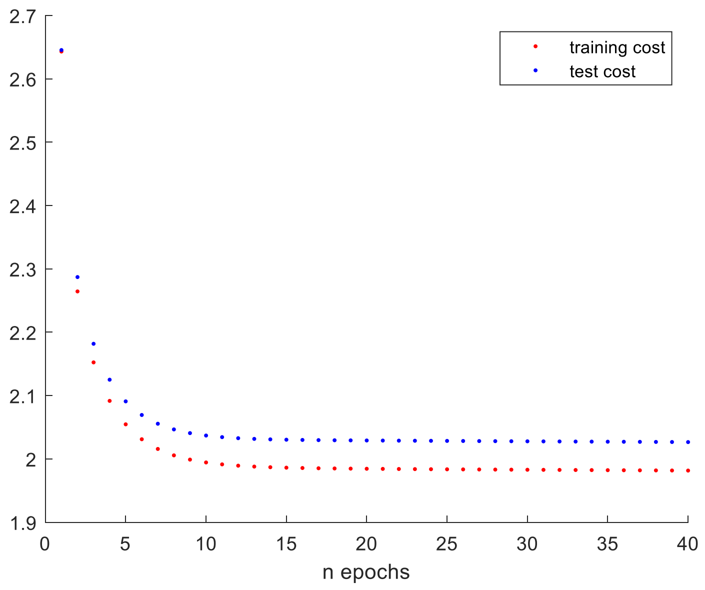


Accuracy: 36,3%

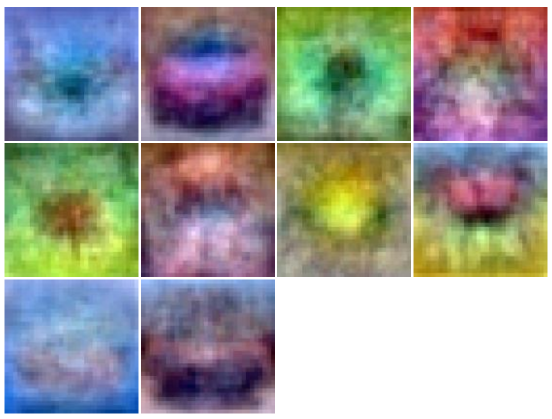


Lambda=0.1, n_epochs=40, n_batch=100, eta=0.01

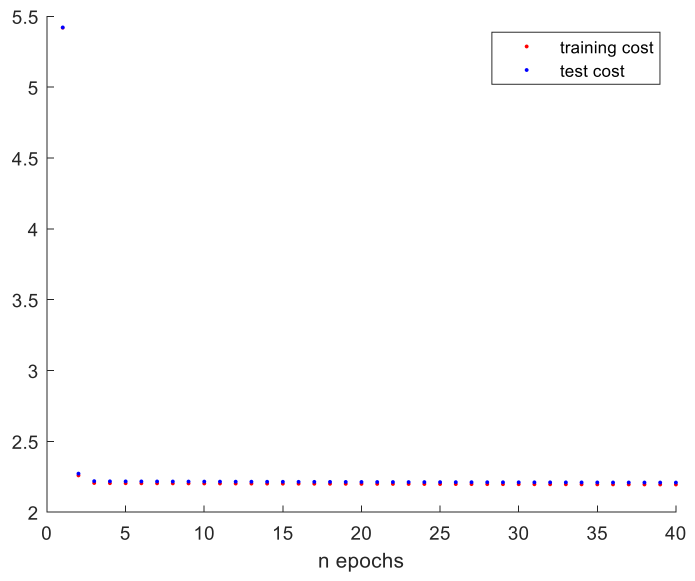


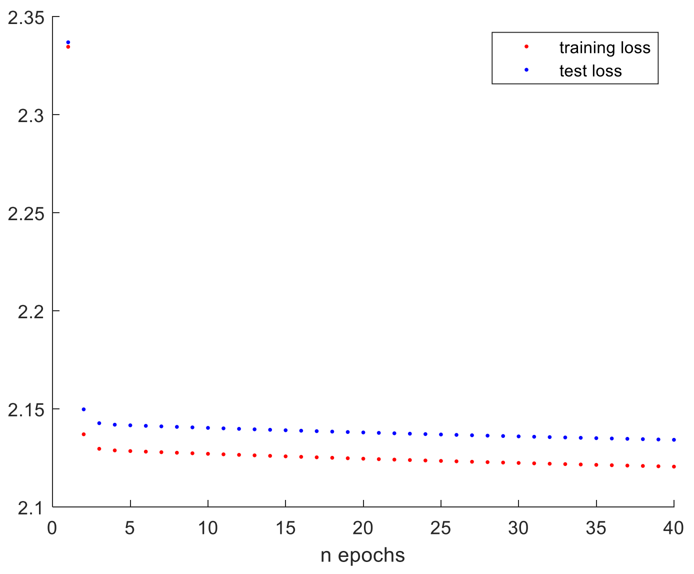


Accuracy: 32.04%

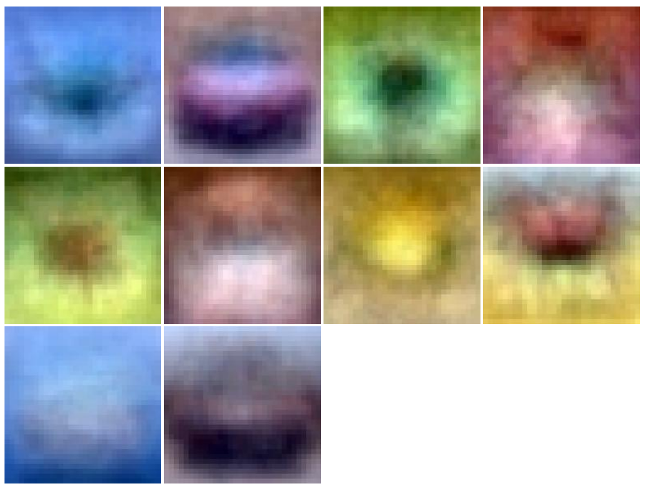


Lambda=1, n_epochs=40, n_batch=100, eta=0.01





Accuracy: 21.30%



3. Analysis

Regularization:

When lambda is equal to zero the cost and loss functions become very low for the training sample but the gap between the results of the test set and the training set increase after each iteration. We can imagine this is a case of overfitting. When lambda increases the gap between the training set and the test set gets smaller.

Learning rate:

With a large learning rate (eta=0.1) the cost function and the loss function are very noisy. We can imagine the step at each gradient descent is too large to find the minimum so the algorithm is not stable.