Assignment 1

Léo Noharet - lnoharet@kth.se

April 4, 2022

Final results

I managed to write the functions to correctly compute the gradient analytically by following the lab assignment instructions. The main difficulty I had was to convert the math to python code. To test that the functions were correctly implemented, I computed the relative error as described in the assignment description, with eps set to 10^{-6}. Then, I also compared the gradients of **W** and **b** computed numerically and analytically with the help of the assert_almost_equal(actual, desired, decimal) function from the numpy.testing python library. This function verifies that the elements of the matrices *actual* and *desired* satisfy the following: $abs(desired - actual) < 1.5 * 10^{-decimal}$. [Dev] I used this with the parameter decimal set to 8, which proves that the difference is small between the two matrices.

Different Parameters and Graphs

The graphs shown in figures 1 - 10 were computed by using numpy.random.seed(400). When lambda is zero, only the cost function is shown as the loss function will be the same. The accuracy of the model for each parameter setting is as follows:
lambda=0, n_epochs=40, n_batch=100, eta=.1 Accuracy of the model = 28.84%.
lambda=0, n_epochs=40, n_batch=100, eta=.001 Accuracy of the model = 39.1%.
lambda=.1, n_epochs=40, n_batch=100, eta=.001. Accuracy of the model = 39.21%.
lambda=1, n_epochs=40, n_batch=100, eta=.001. Accuracy of the model = 37.55%.

The effect of regularization and importance of correct learning rate

The learning rate will affect the convergence speed. A higher learning rate will lead to faster convergence. However a too large learning rate could lead to the gradient descent to diverge from the optimum, or using a less efficient zig-zaggy path to the optimum. In figure 1, we can see that the learning rate is high and the cost (and loss) do not converge to a certain value. This indicates that

the GD diverges which can also be seen in the weight matrices in figure 2 which happen to have little to no pattern compared to the other weight matrices in figures 4, 7,10.

With an increased amount of regularization, the loss and cost of the validation and training data sets converge much faster to a certain value which can especially be seen in figures 8 and 9. If we increase the amount of regularization by increasing the parameter lambda, larger weight values will be penalized more since we minimize the cost function. This way, we can reduce overfitting by penalizing more larger weights.

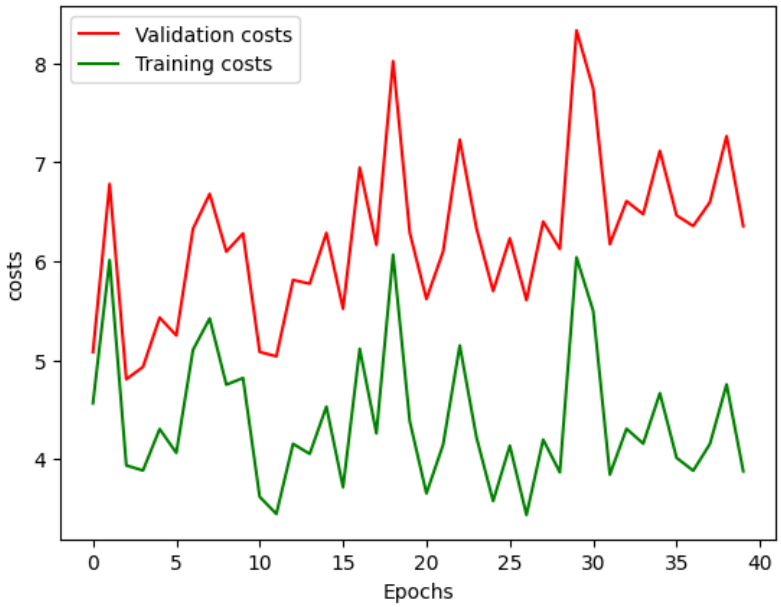

Figure 1: Cost as a function of epoch for lambda=0, n_epochs=40, n_batch=100, eta=.1

References

[Dev] Numpy Developers. numpy.testing.assert_almost_equal. `https://numpy.org/doc/stable/reference/generated/numpy.testing.assert_almost_equal.html`. Accessed: 2022-04-04.

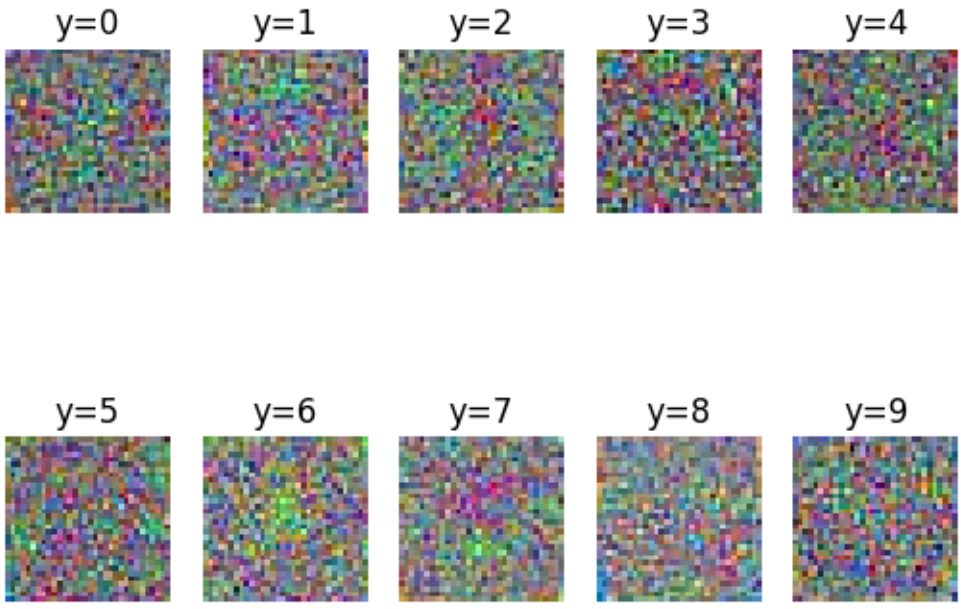

Figure 2: Weight matrices for lambda=0, n_epochs=40, n_batch=100, eta=.1

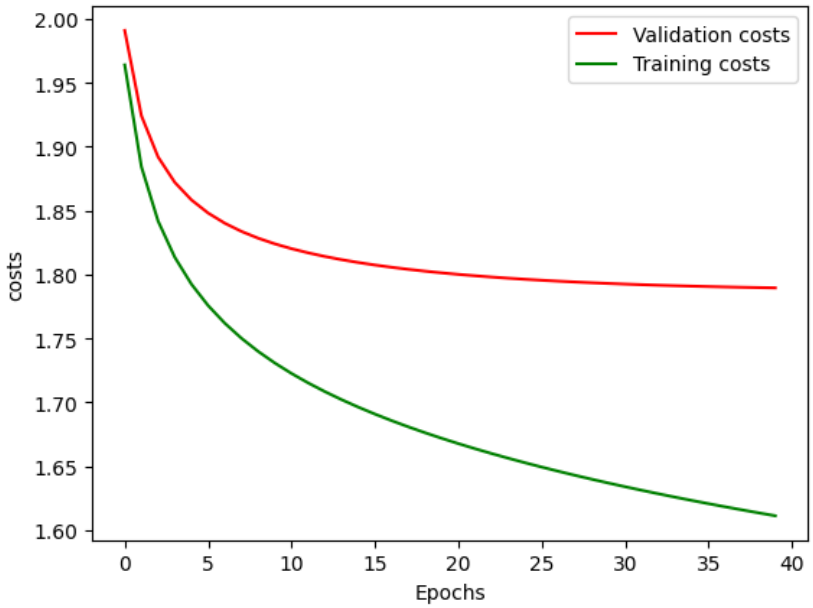

Figure 3: Cost as a function of epoch for lambda=0, n_epochs=40, n_batch=100, eta=.001

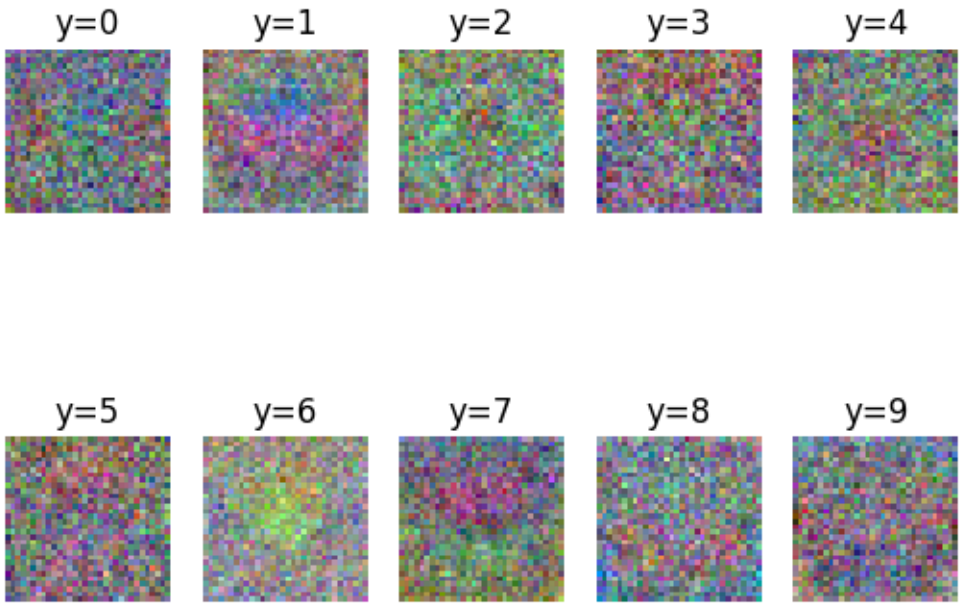

Figure 4: Weight matrices for lambda=0, n_epochs=40, n_batch=100, eta=.001

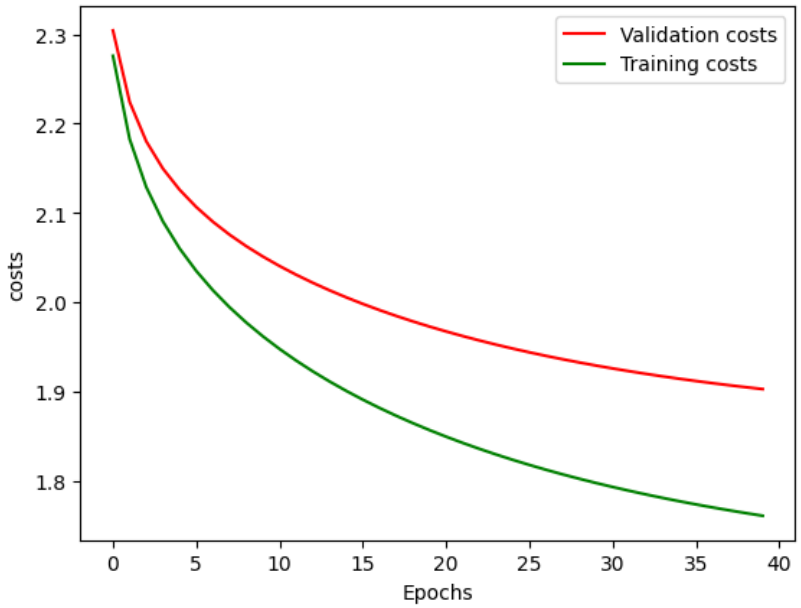

Figure 5: Cost as a function of epoch for lambda=.1, n_epochs=40, n_batch=100, eta=.001

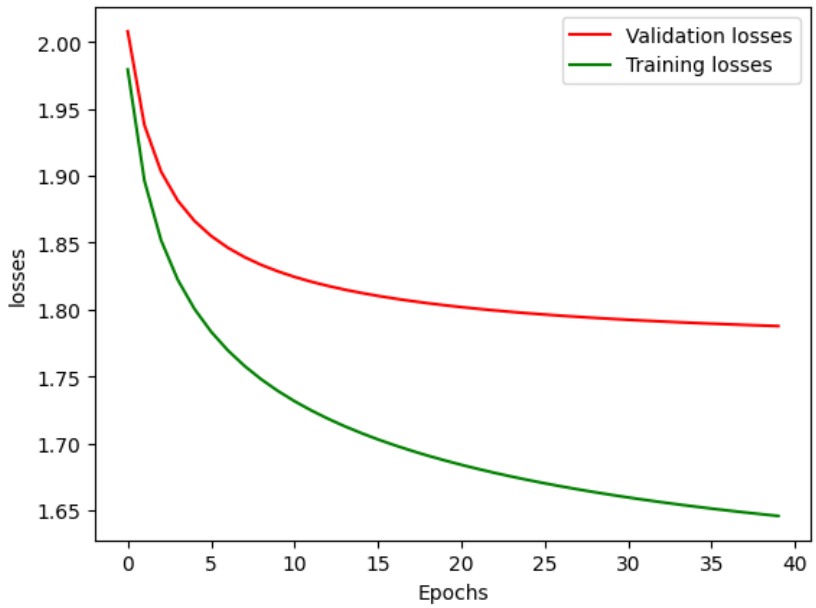

Figure 6: Loss as a function of epoch for lambda=.1, n_epochs=40, n_batch=100, eta=.001

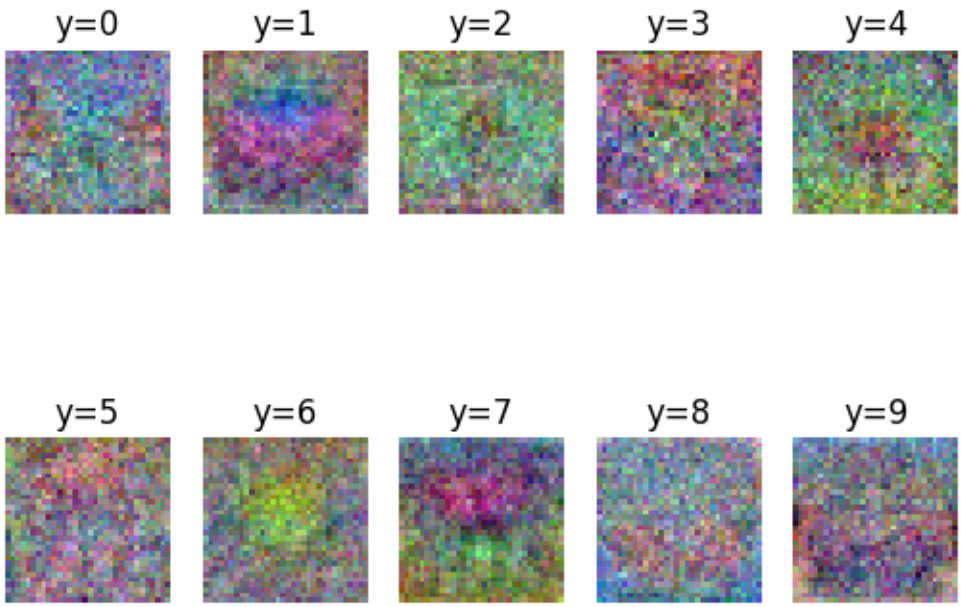

Figure 7: Weight matrices for lambda=.1, n_epochs=40, n_batch=100, eta=.001

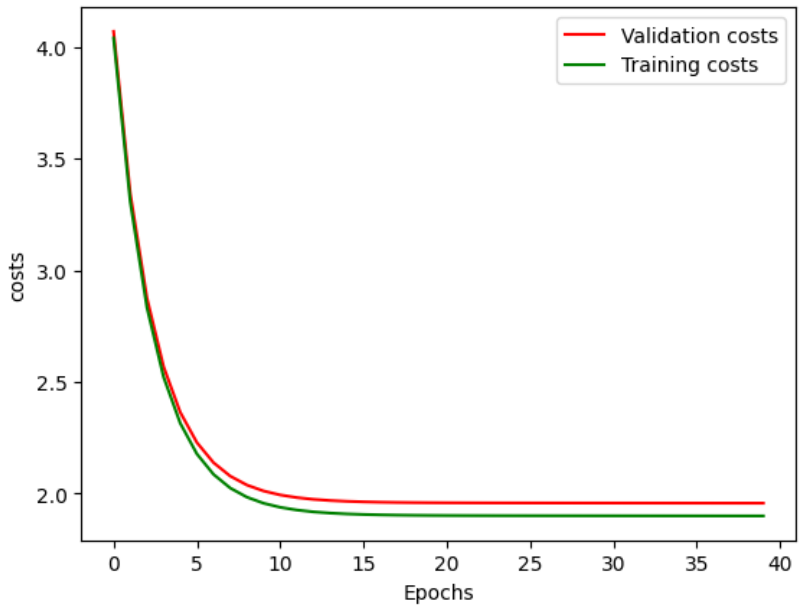

Figure 8: Cost as a function of epoch for lambda=1, n_epochs=40, n_batch=100, eta=.001

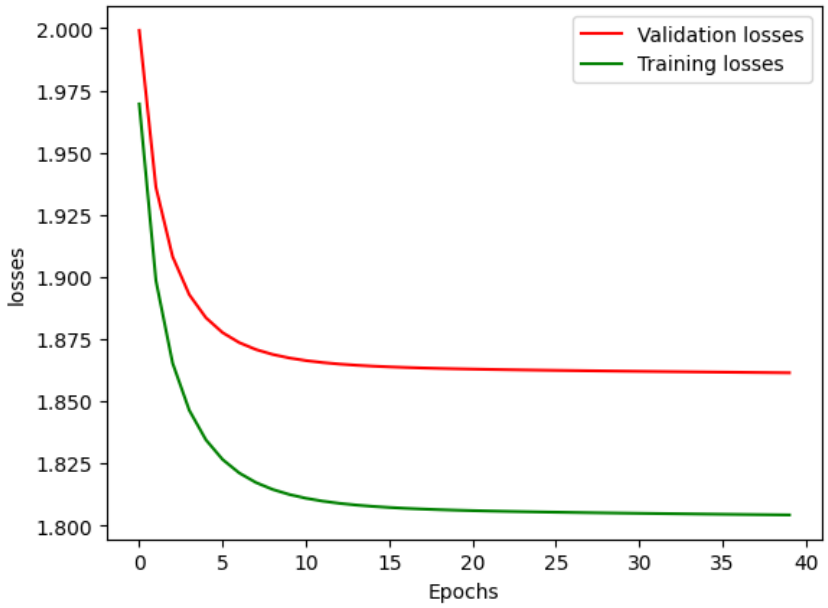

Figure 9: Loss as a function of epoch for lambda=1, n_epochs=40, n_batch=100, eta=.001

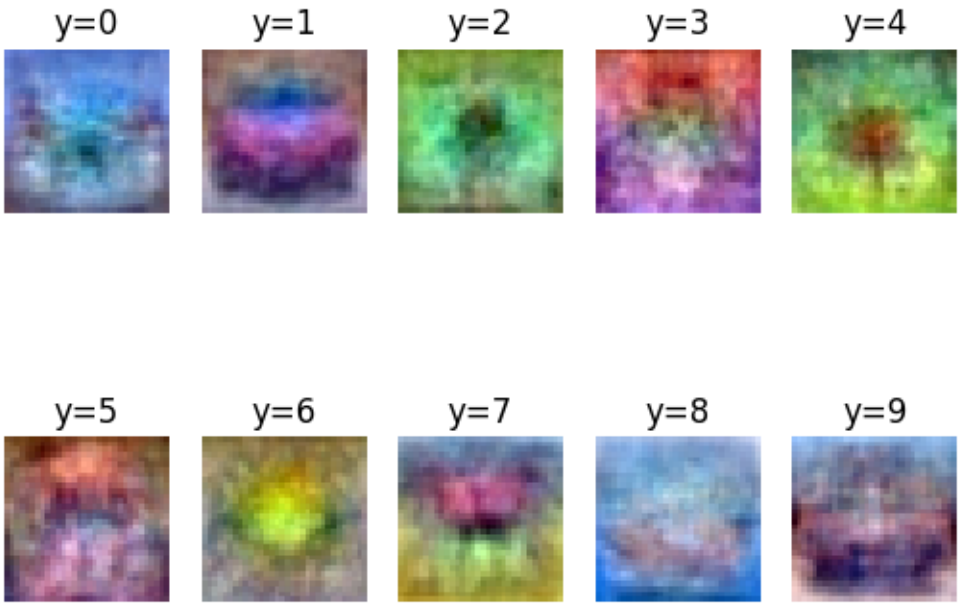

Figure 10: Weight matrices for lambda=1, n_epochs=40, n_batch=100, eta=.001