Assignment 1 Deep Learning

Jan Alexandersson

March 27, 2020

Check gradient

```
def ComputeGradients(X,Y,W,b, lamb):
   P = EvaluateClassifier(X, W, b)
   g = -(Y-P)

   grad_W = np.matmul(g, X.T) / X.shape[1] + 2*lamb*W
   grad_b = np.matmul(g, np.ones((X.shape[1], 1))) / X.shape[1]

   return grad_W, grad_b
```

I checked the relative error between numerically computed gradient and analytically computed gradient

$$\varepsilon = \max(|g_a - g_n|./\max(\text{eps}, |g_a + g_n|)), \tag{1}$$

where eps $= 1e-6$, and we can see the result in the table below.

Batch size =	10	100
ε_b =	4.1887087217846966e-07	6.364790025496572e-07
ε_W =	1.886843874610762e-07	3.415967157102575e-07

This was when using **lambda = 0**. Since all relative errors are small we conclude that our implementeation of the analytical gradient is correct.

Results

We used the settings **n_epochs = 40, n_batch = 100** for all the runs and we got, for different values of **lambda** and **eta**, the result below.

Parameters	Training accuracy	Test accuracy
Eta = 0.1, lambda = 0	0.4487	0.2898
Eta = 0.001, lambda = 0	0.4576	0.3907
Eta = 0.001, lambda = 0.1	0.4457	0.3904
Eta = 0.001, lambda = 1	0.4017	0.3775

We can also plot the cost functions and the images representing the W matrix. These plots are presented below.

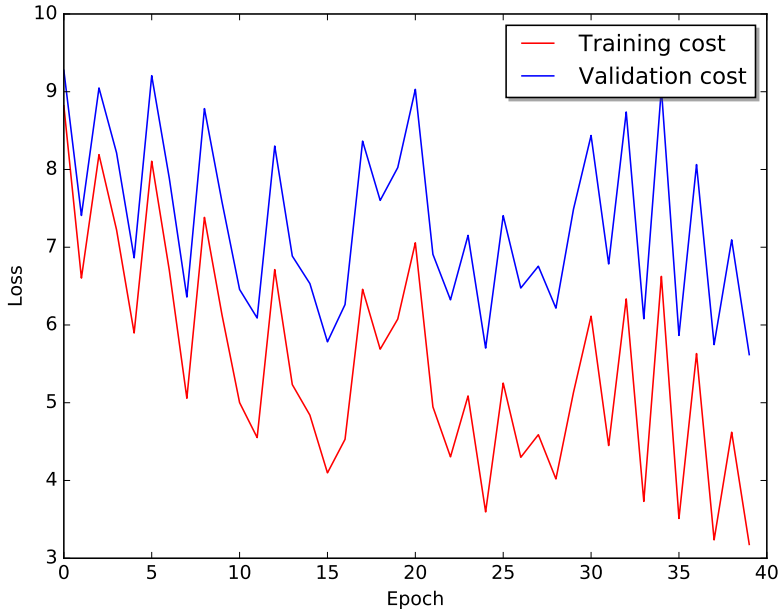

Figure 1: Cost function using **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1**.

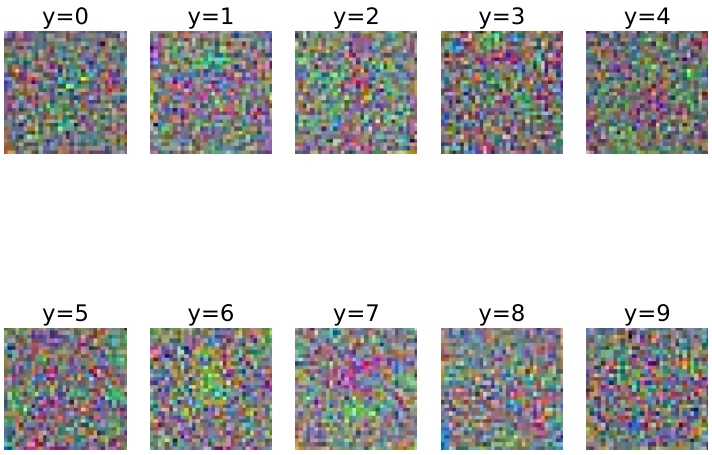

Figure 2: Images representing the W matrix using settings **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1**.

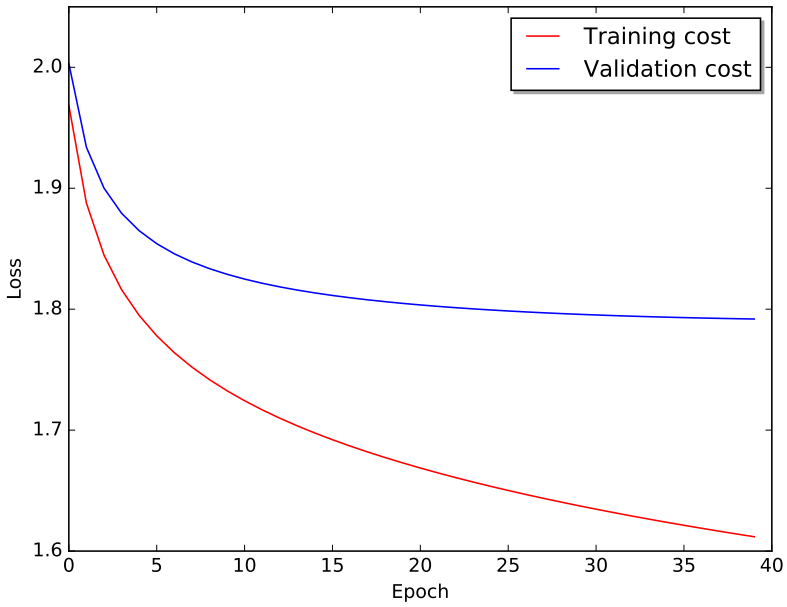

Figure 3: Cost function using **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.001**.

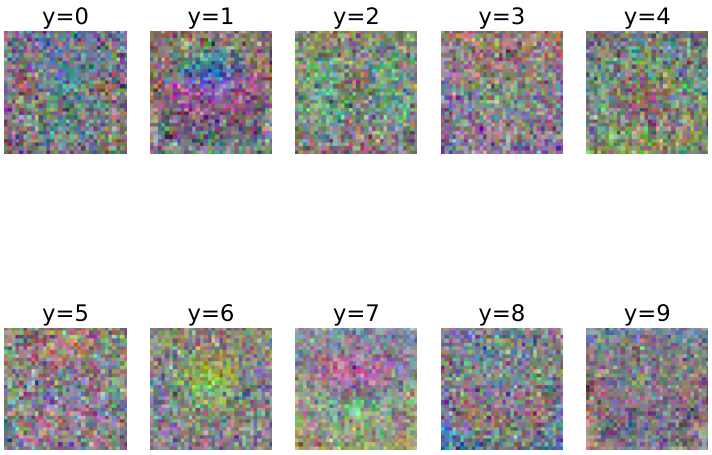

Figure 4: Images representing the W matrix using settings **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.001**.

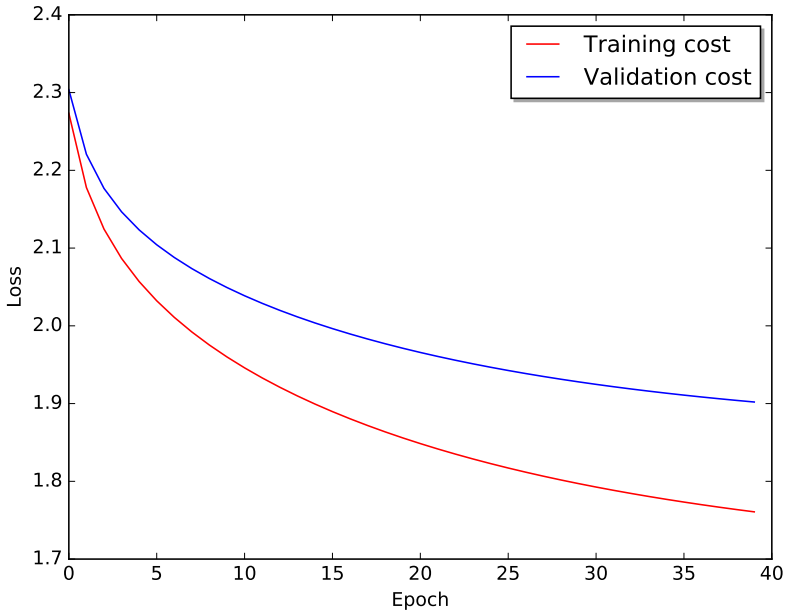

Figure 5: Cost function using **lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.001**.

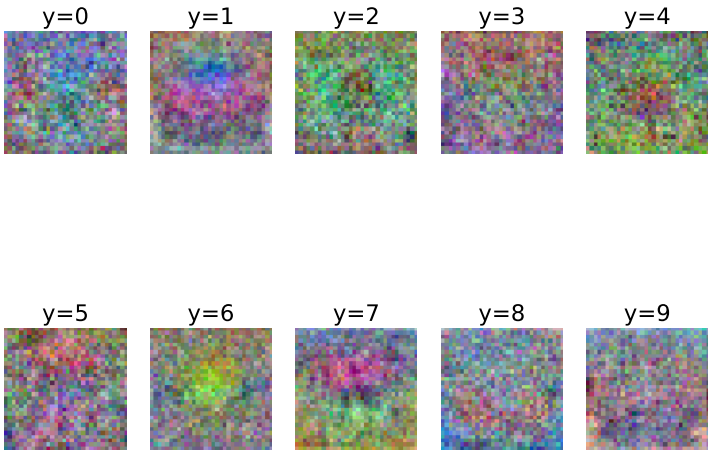

Figure 6: Images representing the W matrix using settings **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.001**.

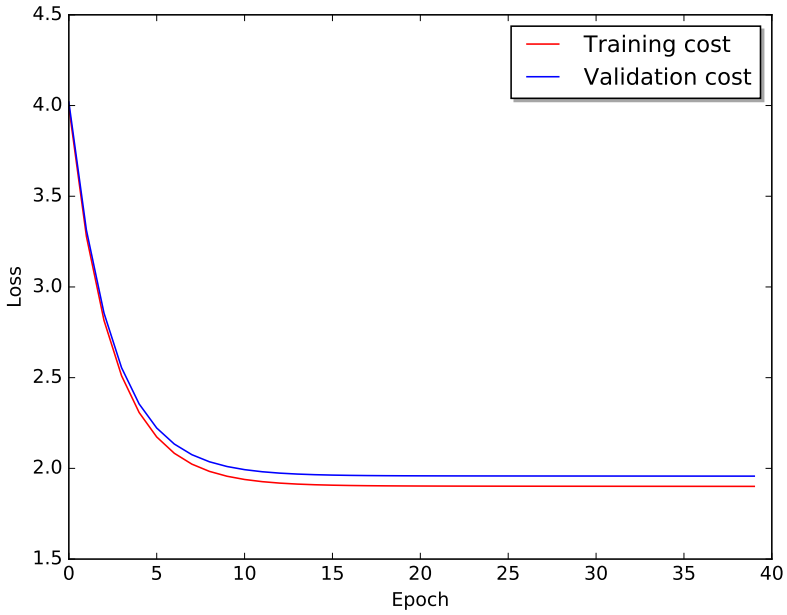

Figure 7: Cost function using **lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.001**.

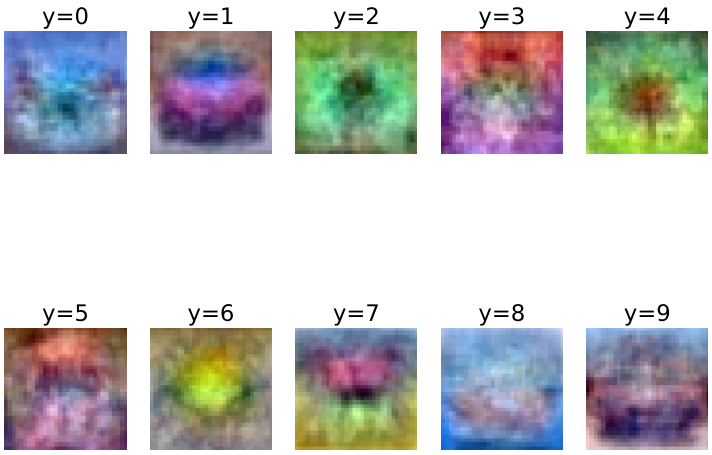

Figure 8: Images representing the W matrix using settings **lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.001**.

When increasing the learning rate the cost function converge faster but is will oscillate be more unstable, which we can see in Figure 1. The learning rate is too large and we will jump around the minumum and not stay at the minimum when we perform the minibatch gradient decent. A lower learning rate will cause the cost function to converge slower but the curve will be more stable and our results are more reliable.

When increasing the regularisation term we get a smoother representation of the W matrix, which we see in Figure 8. This is because when increasing the regularisation term we restrict the model and will generalize more, however a too large value of the regularisation term will generalize too much and reulst in dropped test accuracy while a too low value will cause overfitting.