Report Multi-Linear classifier

1 Analytical Computation of Gradient

- Successfully managed to correctly compute the gradient analytically ?
  Yes, I successfully managed to compute it.
- What tests did I run to check against numerically computed gradient ?
  I ran two tests to check my analytical computation:
  - The first one checked I computed the distance between every component of the W and b from the analytical and numerical:

    ```
    def distance(grad_a, grad_n):
        return grad_a - grad_n
    ```

    I could see clearly the difference and have a clearer idea of the problem.
  - The second one gave me the percentage of weights with a distance bigger than 1e-6 from the numerical analysis:

    ```
    def compute_relative_grad_error(grad_n, grad_a, eps):
        return (abs(grad_n - grad_a) > 1e-6).sum() * 100
                /(len(grad_n)*len(grad_n[0]))
    ```

    This way I could now if I had a lot of weights with a big distance from the numerical values.
  - Finally, I wrote a last one to check how much was the biggest distance from the numerical values.
- Results of these tests ?
  Unfortunately, the tests all failed. I spent hours trying to figure out what was wrong in the analytical gradient computation. It turned out, the computation gradient was correct but the test were wrong. The problem might have come from how I give the parameters to the numerical computation of functions.py.

2 Graph of the loss and the cost function

- Cost function:

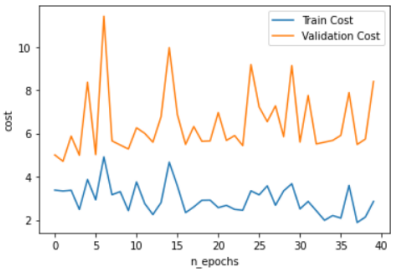

Figure 1: lambda=0, epochs=40, batch=100, eta=.1

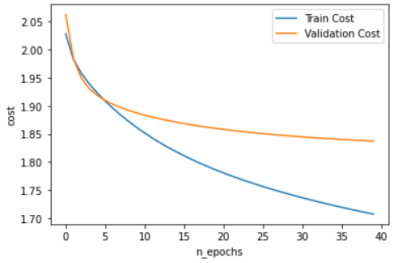

Figure 2: lambda=0, epochs=40, batch=100, eta=.001

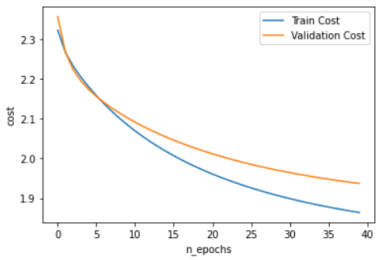

Figure 3: lambda=.1, epochs=40, batch=100, eta=.001

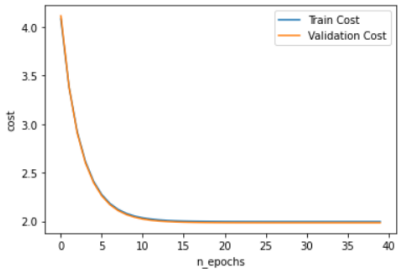

Figure 4: lambda=1, epochs=40, batch=100, eta=.001

- Weights image:

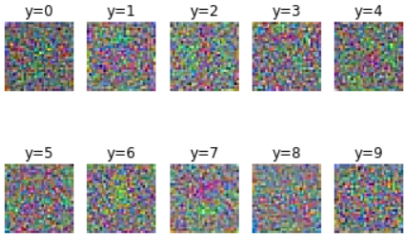

Figure 5: lambda=0, epochs=40, batch=100, eta=.1

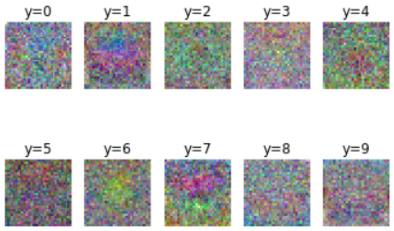

Figure 6: lambda=0, epochs=40, batch=100, eta=.001

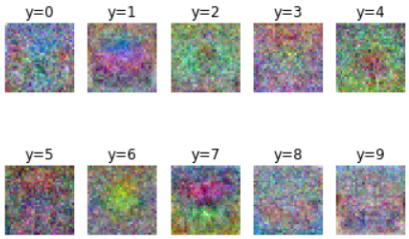

Figure 7: lambda=.1, epochs=40, batch=100, eta=.001

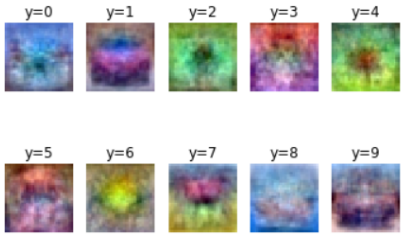

Figure 8: lambda=1, epochs=40, batch=100, eta=.001

- Accuracy on validation set:
  - Case 1: 0.2842
  - Case 2: 0.3858
  - Case 3: 0.3893
  - Case 4: 0.3645
- Increasing the regularization improve the accuracy but to some point. In fact, increasing it leads to lower weights which avoids overfitting. We can see from the graphs and accuracy that the learning rates really matters. The higher the more the gradient zig zag. We need it to be high to some degree so that it zig zag enough but not too much.