Course: DD2424- Assignment 1

Sevket Melih Zenciroglu – smzen@kth.se

Exercise 1: Training a multi-linear classifier

TASK - 1

Top-level: Read in and store the training, validation and test data.
DONE

TASK - 2

Top-level: Compute the mean and standard deviation vector for the training data and then normalize the training, validation and test data w.r.t. these mean and standard deviation vectors.
DONE

TASK - 3

Top-Level: After reading in and pre-processing the data, you can initialize the parameters of the model W and b as you now know what size they should be. W has size Kxd and b is Kx1. Initialize each entry to have Gaussian random values with zero mean and standard deviation .01.
DONE

TASK - 4

Top-level: Check the function runs on a subset of the training data given a random initialization of the network's parameters: P = EvaluateClassifier(trainX(:, 1:100), W, b)
DONE

TASK - 5

Write the function that computes the cost function given by equation J = ComputeCost(X, Y, W, b, lambda)
DONE

TASK - 6

Write a function that computes the accuracy of the network's predictions given by equation (4) on a set of data.
DONE

TASK - 7

Write the function that evaluates, for a mini-batch, the gradients of the cost function w.r.t. W and b, that is equations (10, 11).
DONE – Details below

TASK - 8

Once you have the gradient computations debugged you are now ready to write the code to perform the mini-batch gradient descent algorithm to learn the network's parameters
DONE – Details below

TASK - 7 – Details

I have successfully managed to write the functions to correctly compute the gradient analytically. I used the suggested tests in the assignment document.

Test-1: I examined the absolute differences between analytical and numerical computations. All these absolute differences where less than 1.06 e-08 and the mean for the differences was 1.17 e-08 for

grad-W values. For grad-b values, the case was similar, all the values were smaller than 5.49 e-08 and the mean of the difference was 4.48 e-08.

Test-2: I computed the relative error between analytical and numerical computations. As you can see in the below table, all the values (differences) were smaller than 0.01, even this value was a bit high, when I checked the mean, it was 3.77 e-06, so the smallest difference was an outlier for gradient-W. For gradient-b, the values were smaller than 5.59 e-06 and the mean was 1.68 e-06.

Test-1

```
asgn1.Task7_test1()
```

```
grad_W_difference_MEAN = 1.1744294772342884e-08
grad_W_difference_MIN = 4.2462113079150265e-09
grad_W_difference_MAX = 2.0602097774297112e-08

grad_W_MIN = 2.6074914276936644e-07
grad_W_num_MIN = 2.7267077484793845e-07
grad_W_MAX = 0.06195850560934546
grad_W_num_MAX = 0.06195849211465543

grad_b_difference_MEAN = 4.485301018791743e-08
grad_b_difference_MIN = 3.3935816448615874e-08
grad_b_difference_MAX = 5.488295274670707e-08

grad_b_MIN = 0.0045187974034357455
grad_b_num_MIN = [0.00451885]

grad_b_MAX = 0.07088109784271754
grad_b_num_MAX = [0.07088115]
```

Test-2

```
asgn1.Task7_test2()
```

```
gradient_W_error_check_2-MAX: 0.011921632078572001
gradient_W_error_check_2-MIN: 8.468693465709592e-08
gradient_W_error_check_2-MEAN: 3.7668925332927018e-06
gradient_W_error_check_2-STD: 9.124804643697527e-05

gradient_b_error_check_2-MAX: 5.594272526021969e-06
gradient_b_error_check_2-MIN: 3.4259577097674394e-07
gradient_b_error_check_2-MEAN: 1.6819530245468294e-06
gradient_b_error_check_2-STD: 1.6728861311950061e-06
```

TASK - 8 – Details

If we don't use any regularization, we observe a big difference between training and the validation data results. However, once the regularization impact is increased, the gap between those two values becomes smaller. The downside of this, on the other hand, we end up with higher costs and lower accuracies when the high regularization is in the picture. So, with the increased regularization, we can say, we sacrifice from better numbers/figures, but our results are being more consistent between training and validation. Moreover, higher regularization (lambda_cost=1, figureGroup-4, W_4 and b_4) finds the solution faster (at around epoch-10 for the figures shared whereas at figureGroup-3 the solution still improves itself up until epoch-40) but as mentioned with worse values.

Having a high value for the learning rate is not good as it is clearly seen in the first graphs below. It is difficult for the network to learn with high learning rates since it will miss the local minima. Small learning rate leads to a slower conversion to the solution but with better results.

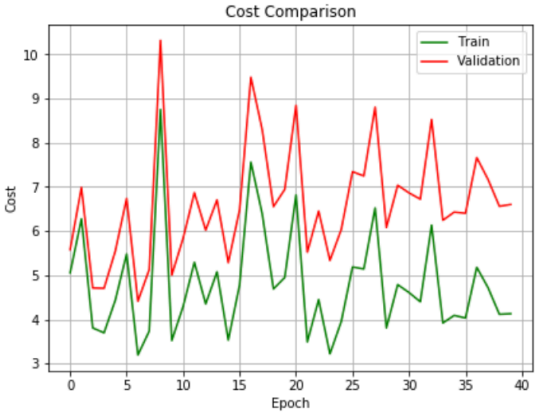
lambda	**epochs**	**batch**	**eta**	**Accuracy (%)**
0	40	100	.1	27.24
0	40	100	.001	39.03
.1	40	100	.001	39.2
1	40	100	.001	37.35

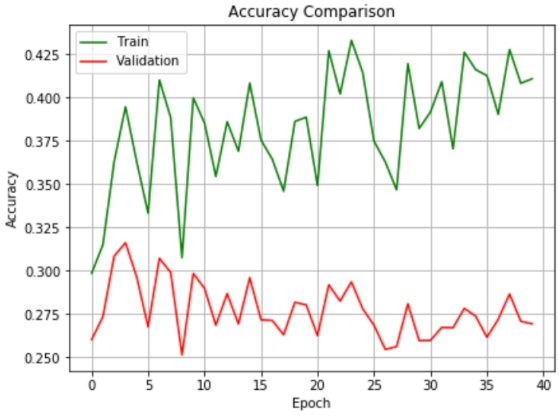
```
1 W_1, b_1 = asgn1_task8.Task8(comp_task8, GDparams_1)
```

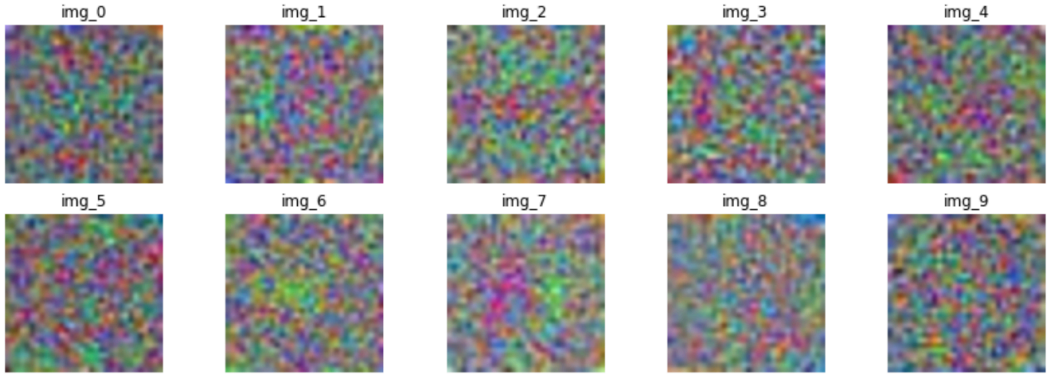
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.1, n_epocs=40, lambda_cost=0]

Test data final accuracy: 27.24%
```








```
Calculation time: 0:01:06.487666
```

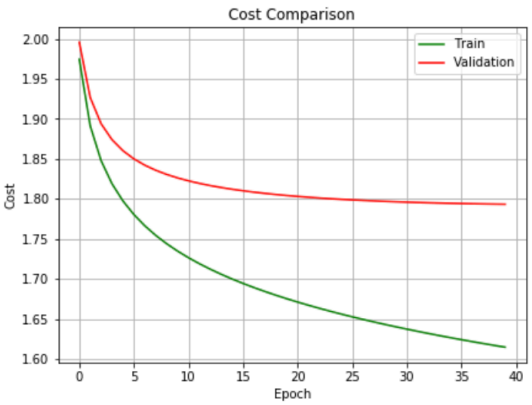
<<<<< ---------- >>>>>

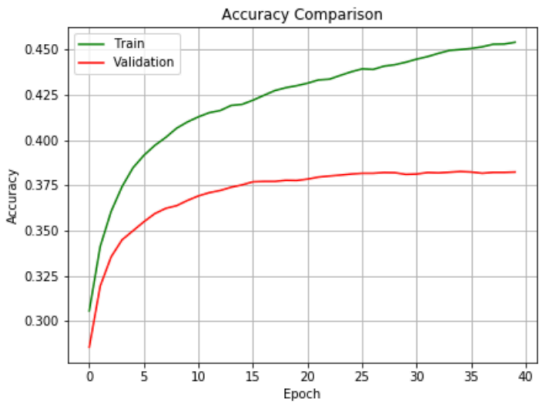
```
1 W_2, b_2 = asgn1_task8.Task8(comp_task8, GDparams_2)
```

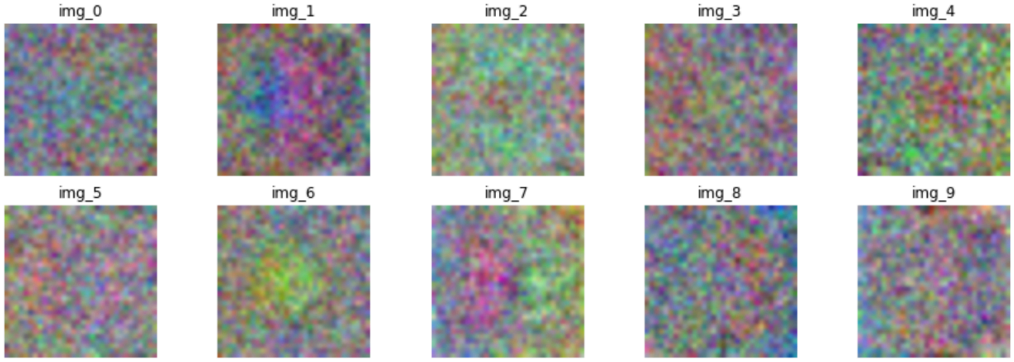
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.001, n_epocs=40, lambda_cost=0]

Test data final accuracy: 39.03%
```







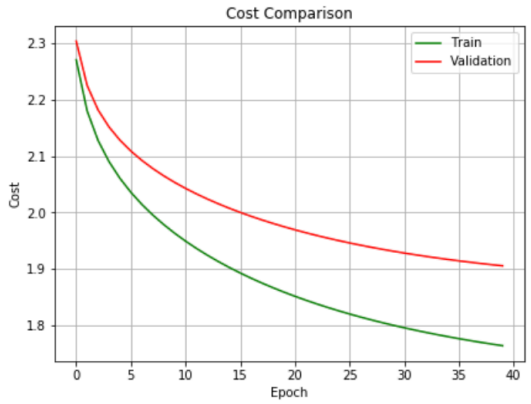

```
Calculation time: 0:01:04.465185
```

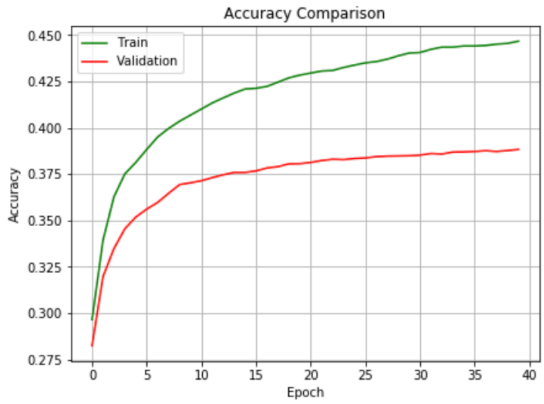
```
1 W_3, b_3 = asgn1_task8.Task8(comp_task8, GDparams_3)
```

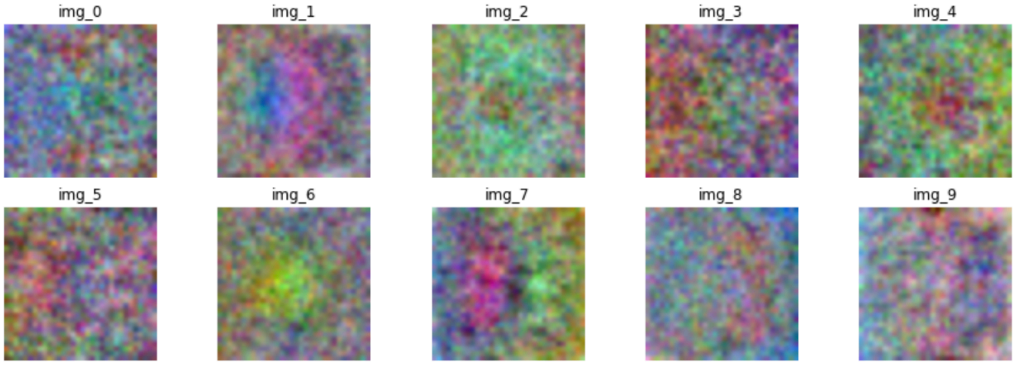
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.001, n_epocs=40, lambda_cost=0.1]

Test data final accuracy: 39.2%
```








```
Calculation time: 0:01:05.784027
```

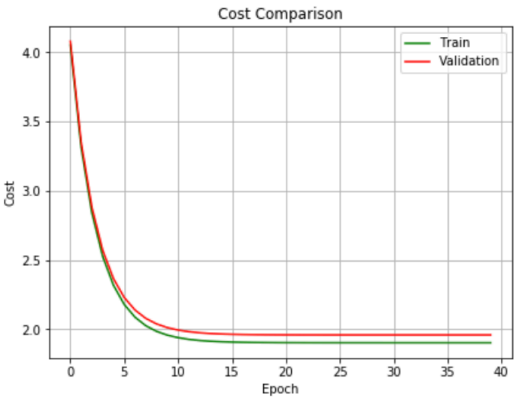
<<<<< ---------- >>>>>

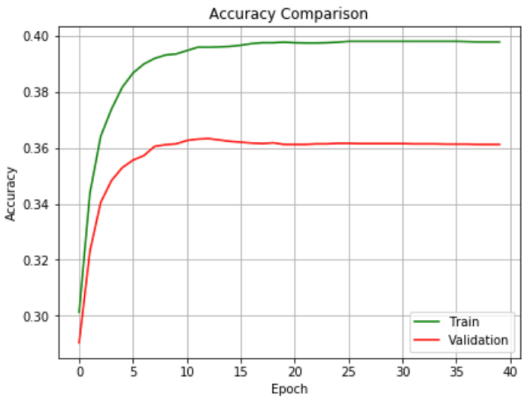
```
1 W_4, b_4 = asgn1_task8.Task8(comp_task8, GDparams_4)
```

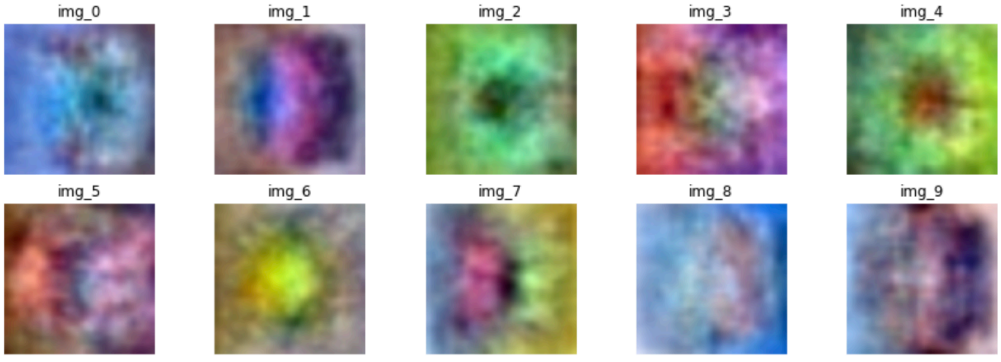
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.001, n_epocs=40, lambda_cost=1]

Test data final accuracy: 37.35%
```







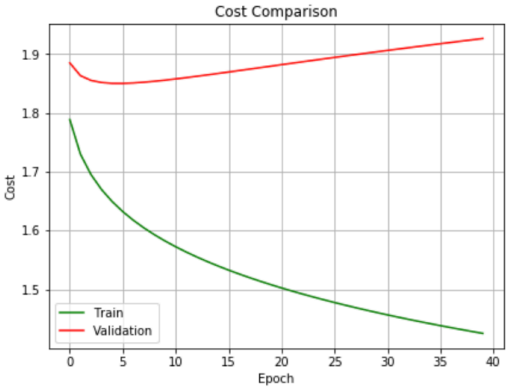

```
Calculation time: 0:01:07.847391
```

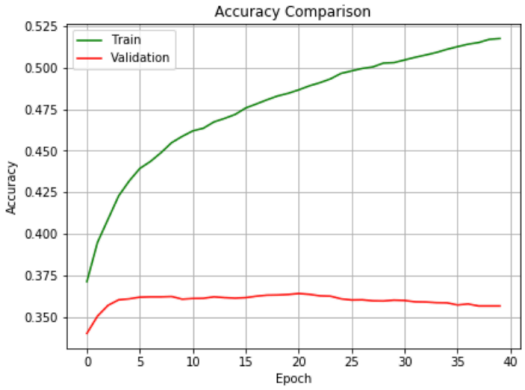
```
W_5, b_5 = asgn1_task8.Task8(comp_task8, [100, 0.01, 40, 0])
```

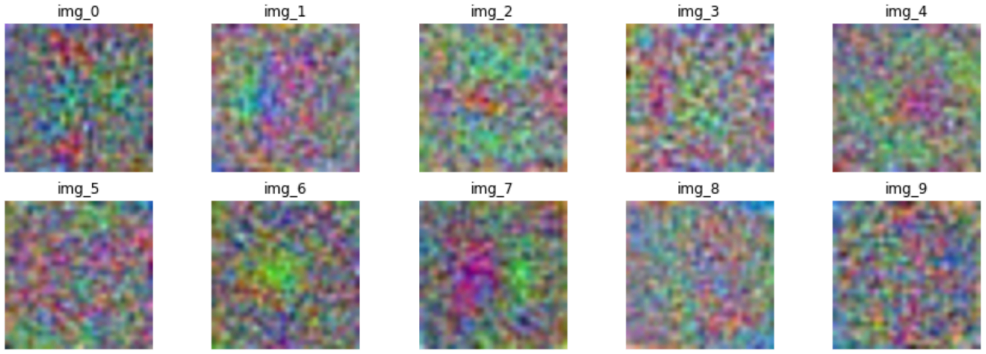
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.01, n_epocs=40, lambda_cost=0]

Test data final accuracy: 35.73%
```








```
Calculation time: 0:01:12.254742
```

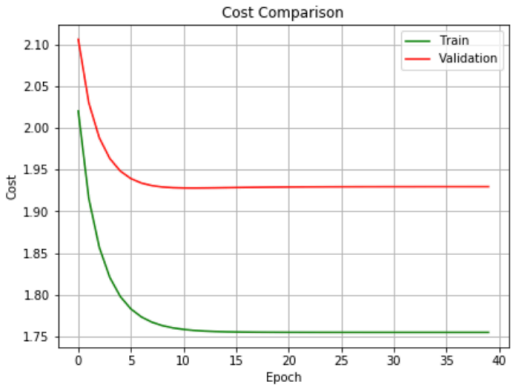
<<<<< ---------- >>>>>

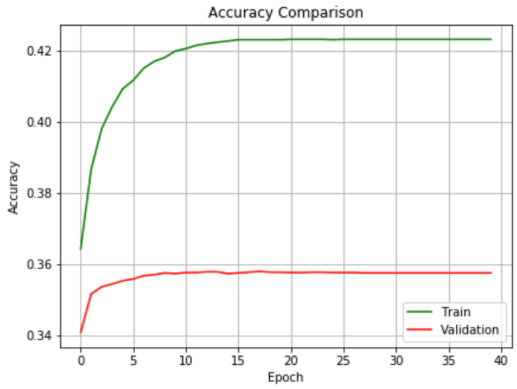
```
W_6, b_6 = asgn1_task8.Task8(comp_task8, [100, 0.01, 40, 0.1])
```

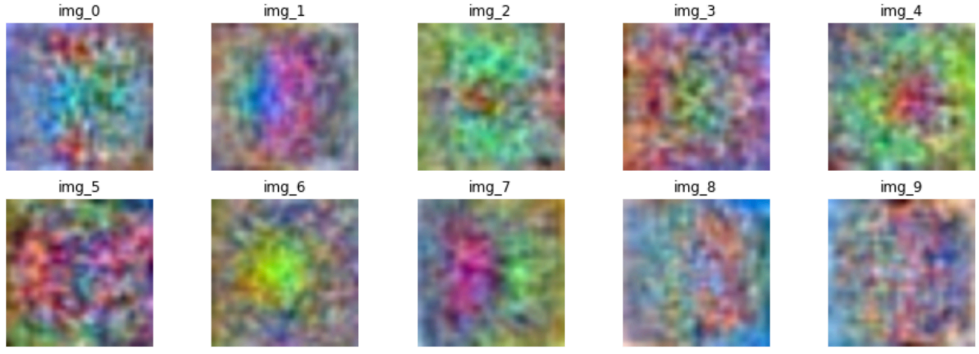
```
************ CostCalculations ************

GDparams: [n_batch=100, eta=0.01, n_epocs=40, lambda_cost=0.1]

Test data final accuracy: 35.86%
```








```
Calculation time: 0:01:04.821940
```