DD2424 Assignment 1
Peiyang Shi
pyshi@kth.se
9004208238

The assignment was implemented using Python. In order to further improve efficiency, the implemented heavily make uses of vectorizes matrices, avoiding any loops. The only loop is used in the implementation is the epoch loop for SGD.

Gradient calculation

The closed form of SGD was implemented and compared against numerical estimated SGD. The first method of comparison was the absolute difference - by subtracting one SGD method from another, one can see the absolute difference in calculation. If the error is small enough, it can be considered accurate. A sample of the absolute error can be seen in fig.1. The error are within 1e-7 - below the heuristic threshold 1e-6 provided in instructions.

-1.1387847281482677e-07	-1.1358478882970979e-07	-1.1376762458839562e-07	-1.1372599072710654e-07
-1.146915974657714e-07	-1.1443319848647437e-07	-1.1426983064638518e-07	-1.1347442065851032e-07
-1.086039478452594e-07	-1.0808742851448061e-07	-1.0852818137924158e-07	-1.0833656456654683e-07
-1.1483227171059374e-07	-1.1480040790559642e-07	-1.1433988854510357e-07	-1.1418044138650218e-07
-1.1450477182227414e-07	-1.142287965629l9e-07	-1.14523523578932e-07	-1.1383474322527011e-07
-1.1351344461098256e-07	-1.1315197562949963e-07	-1.1337104080522131e-07	-1.1291969840773741e-07
-1.1538289139065294e-07	-1.1501271929628687e-07	-1.1504086453763274e-07	-1.1435183451709297e-07
-1.1091681035180906e-07	-1.0986144037983953e-07	-1.1111750365666584e-07	-1.1046367359647709e-07
-1.205729492151364e-07	-1.19507929760565e-07	-1.1970719232534677e-07	-1.197914793332755e-07
-1.1142216804066463e-07	-1.1069504111602124e-07	-1.1076842808960263e-07	-1.1053588250958812e-07

Fig 1, absolute difference between SGD methods (sample)

Another SGD error metric was used. It's described by the equation below, which describes a normalized error and the dominator is capped with lower bound scalar value *eps*

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)}$$

The normalized SGD error metric is adopted for scenarios where the gradient value has become too small to be captured by floating points representation. The normalized SGD error sample results were shown in fig 2, the errors still lie below 1e-05.

5.923143186004371e-06	5.020563647876651e-06	5.75871754016898e-06	5.1016592968472036e-06
2.2797889910710777e-06	2.281599400650184e-06	2.3274174968671427e-06	2.1327909758313693e-06
1.82909670444054e-06	1.5612345132736411e-06	1.6913976763544088e-06	1.4177263822448e-06
3.777609780782098e-06	3.3901849519803516e-06	3.911851775205849e-06	4.26888552725159e-06
4.150781538649012e-05	3.691342330714154e-05	2.1294126448089422e-05	0.00012094572435247348
4.125872499273132e-06	3.1156595867350584e-06	2.6286812658551717e-06	2.758029662113412e-06
2.934128159193469e-06	3.861410772554869e-06	4.283987295500585e-06	4.6687801549777105e-06
2.7614995530397438e-05	2.7781484747489443e-05	1.073888705893849e-05	4.3099497345470866e-05
1.0522739336671825e-06	1.1233389421538804e-06	1.0619380918924997e-06	1.1319032020529562e-06
1.6756461229693165e-06	1.8889701671205742e-06	1.9393091817838358e-06	1.870203536148724e-06

Fig 2, normalized error between SGD methods (sample)

A mini-batch gradient descent process was then implemented using the SGD method. **It achieved a best result of 38.9%** without optimization. Although further optimizations for were made for the bonus and is discussed in the bonus report.

Results

The first configuration was **lambda=0, n epochs=40, n batch=100, eta=.1**. This configuration produced an accuracy of 20.6%, better than the 11% initial accuracy without training. However, the etta appeared to be too large and did not properly converge. It is understandable as there there are 100 batches with each epoch, and each will perform a 10% gradient on the original weights, and thus it cannot allow weights to stabilize and gradually diminish the gradients.

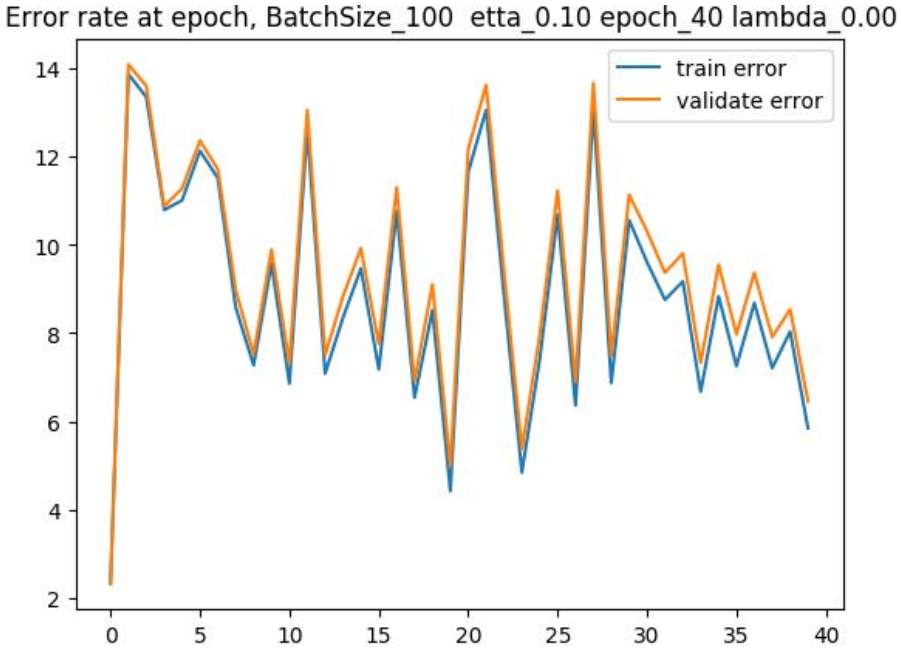


Fig.3

The weights were also plotted and there are enough variety and diversity in the weights, in an ideal situation, the weights would classify one CIFAR-10 object per node. Though in this configuration, learning could not stabilize, and the weights representation carried little meaning.

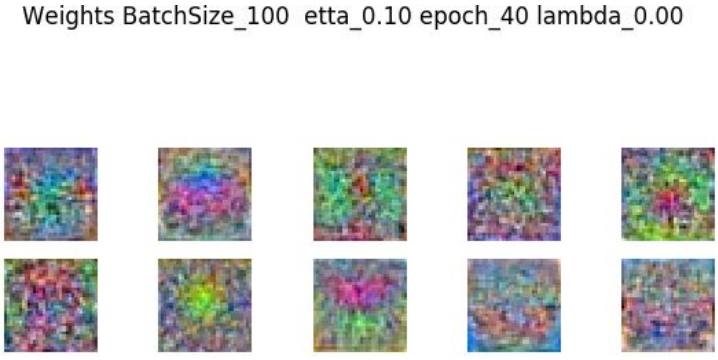
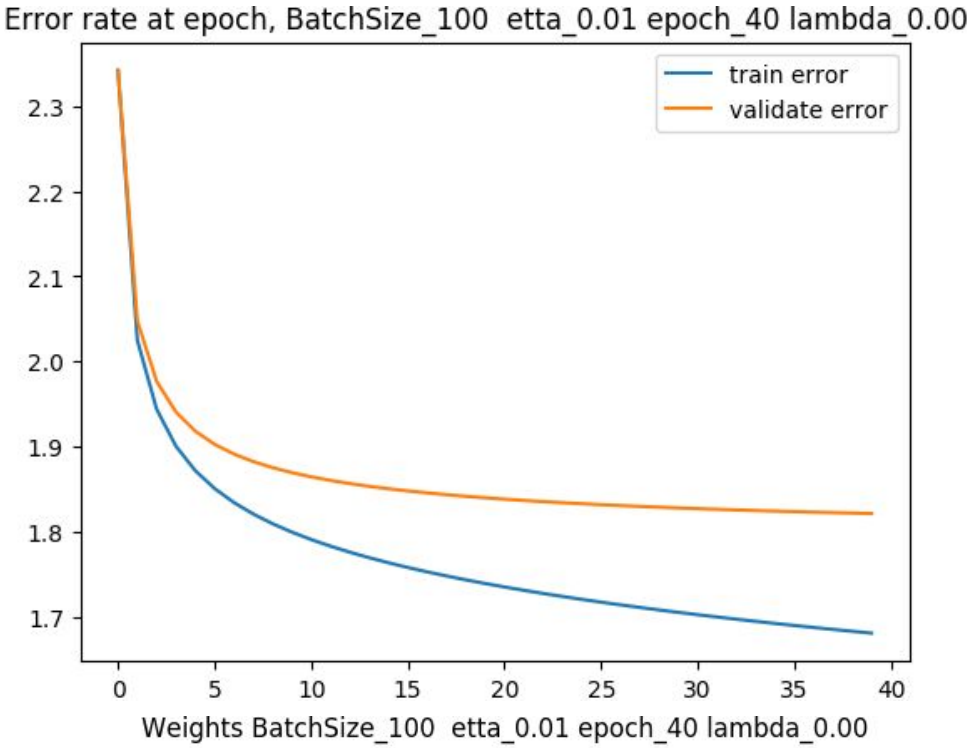
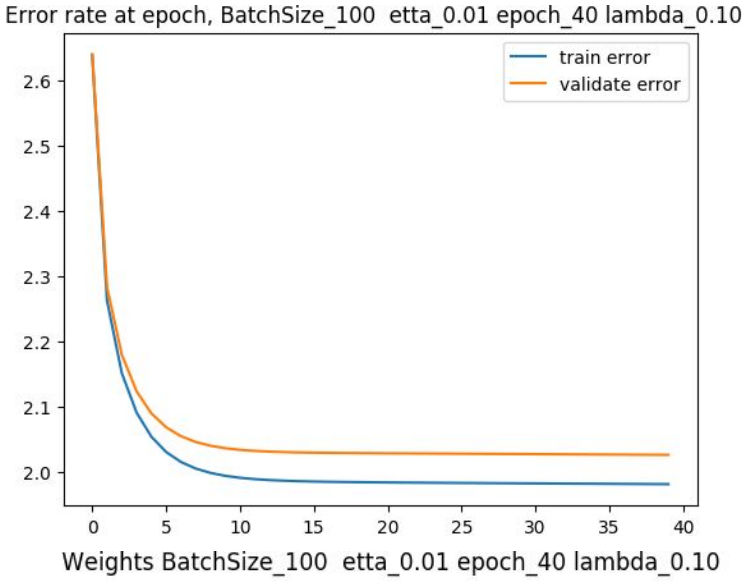


Fig. 4

Second configuration was **lambda=0, n epochs=40, n batch=100, eta=.01**. This configuration produced an accuracy of 37.76%. Training cost reduces in a logarithmic-like rate, appears to be converging around 1.7 for the training set, however, 40 epochs did not appear to be enough for the system to converge. Since the learning rate in this configuration was an order of magnitude lower (etta 0.1 vs etta 0.01), it is believed that the slower learning rate was allowing the system to stabilize, allowing gradient to reduce each epoch. The weight vector resembles that of the first configuration in fig 4. This configuration leads to a slightly more 'grainy' weights, as in weights between neighbours tend to have more random than the previous configurations. It is believed to be a result of the random initialization which weights began as grainy noisy and training gradually smoothes out weights.



Third configuration was **lambda=.1, n epochs=40, n batch=100, eta=.01**. This configuration has a regularization Lambda activated and produced an accuracy of 33.15%, which is less accurate than without regularization. Training cost again reduces in a logarithmic-like rate but converges just below 2.0 while validation error converges around 2.05 as seen the figure below. An interesting observation is that the convergence occurs sooner than that without regularization factor. In the previous configuration, the convergence did not reach even at 40th epoch while with regularization the system converged around 10th epoch. This is believed that the regularization factor applies a "force" on weights, which acts similar to that property of a bound for the weights, and thus gradients were descending faster because its range is soft bounded. The less accuracy is believed that, intuitively and analogously, the regularization soft bound was penalizing even the optimal configuration, thus the fast convergence converged sacrifices accuracy for speed.



Fourth configuration was **lambda=1, n epochs=40, n batch=100, eta=.01**. This is an extreme setup of the regularization term. It only produced an accuracy of 24.94%. The learning converged within the first three epochs. Extending the discussion from previous configuration, the regularization can be understood as creating a force on the weight space, creating a soft bound. The higher the regularization factor, the smaller the softbound will be, which will converge much faster at the sacrifice of the accuracy of weight.

The visualized weight representation can also be seen below. They resemble beautiful pieces of abstract modern art. Though the discussion of art is out of the scope of this report, one can observe a notable difference comparing to the second configuration: Smoother edges and more differentiated colors amongst one another. Each weight visualization be seen to have a vague center area (which is usually where CIFAR-10 objects are located), however the “blurriness” can be insight to why accuracy was rather low.