ASSIGNMENT- 02

Architecture: MLP with 2 Hidden layers with 600 neurons per each layer

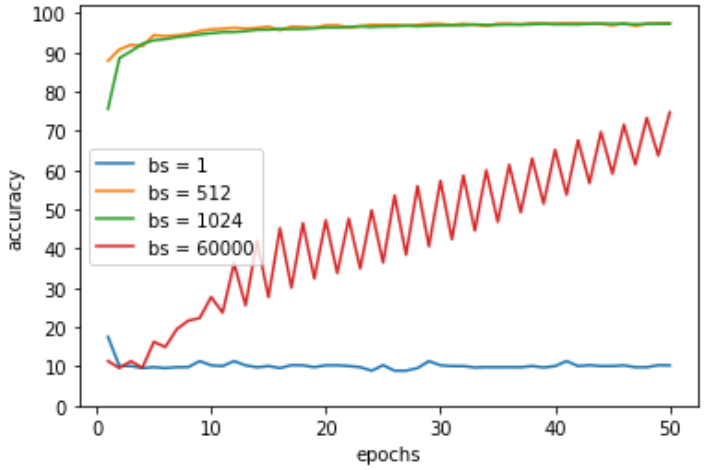
Exercise 1.

Modify the code of mnist-assignment.py to train the chosen network using SGD with minibatches of different sizes (starting from size 1). Discuss how the SGD method with various minibatch sizes compares in terms of speed of convergence and final accuracy on test set with GD.

I experimented on 4 batch sizes. The SGD method with batch sizes 512 and 1024 performed the best (both in terms of accuracy and speed of convergence)compared to that of with batch sizes 1 and 60000 (GD); Overall, Batch size of 1024 outperforms 512 by slight margin. But the difference is not much and hence it is very reasonable to chooose 512 as the optimum Batch size for computational purposes.

The speed of convergence of the Gradient Descent is comparatively linear (slow). This is because the weights are corrected only after all the traiing data is seen by the model.

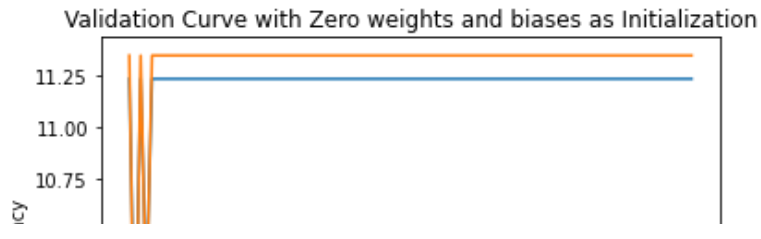
The choice of Batch Size 1 performs the worst since the weights are re-calibrated after each training data. This results in very high bias model. Since, each data sample vary a lot with respect to each other (most cases) the model cannot generalize very well to the heterogenity of new samples because of severe over-fitting.

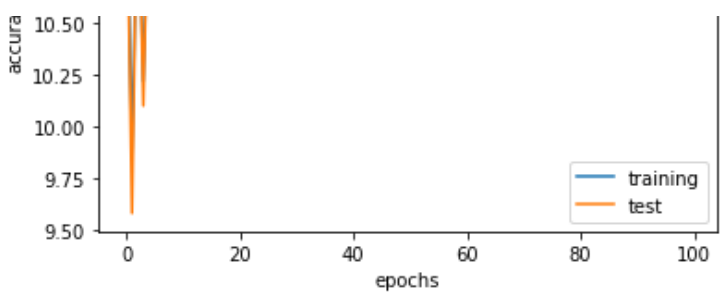


Exercise 2.

Initialize all the weights and all biases to a common value (for instance 0). Try to train the network either with GD or SGD. What happens? Why?

As it can be seen from the plot below, the accuracy of the Gradient Descent model with all weights and biases intialized to zero performs very poorly. Infact, the accuracy is same in each Epoch as if only certain data is being correctly classified. This is due to the fact that, all the weights are zero which means all the gradients in all directions are same resulting in neurons learning same features in every iteration. Additionally, as per the standard literature, if we initialize the network with a constant weight then there will be a problem of exploading gradient (high initial value) and vanishing gradient (low value initialization).





Exercise 3.

Q1:

Can you find values of the learning rate for which this online method is convergent? If so what is the speed of convergence compared to SGD? Discuss.

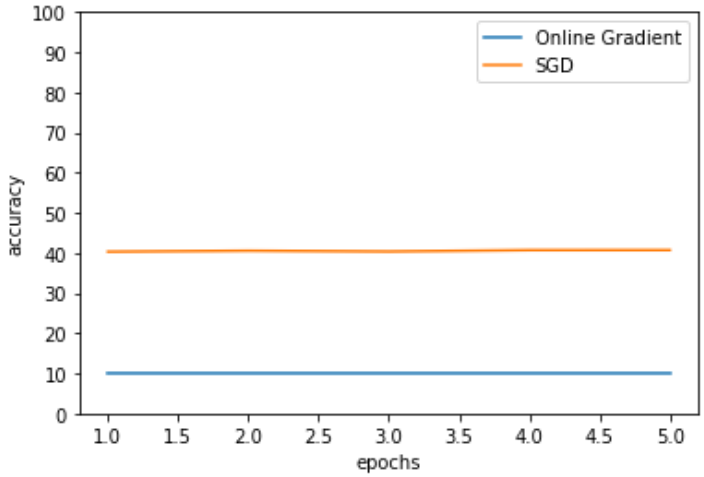
I have selected the Online Gradient and SGD for $n = 20000$. (i.e., stream of e_n sequence is generated throughout each Epoch continuosly for values $n > 60000$)

Even after very low values of Learning Rates such as $0.001, 0.0001, 0.00005$, the Online Method did not converge for 10 Epochs but it's performance started to increase gradually but still very slow.

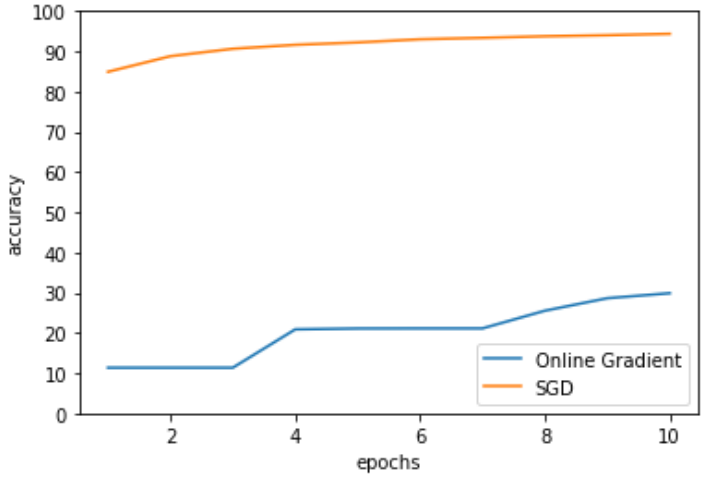
For LR= 0.001 both SGD and online gradient are appeared to be stuck (Pseudo convergence) with no improvement in their performance. This can be due to large learning rate because of which gradient optimization is ineffective.

On the contrary, SGD converges for smaller learning rates.

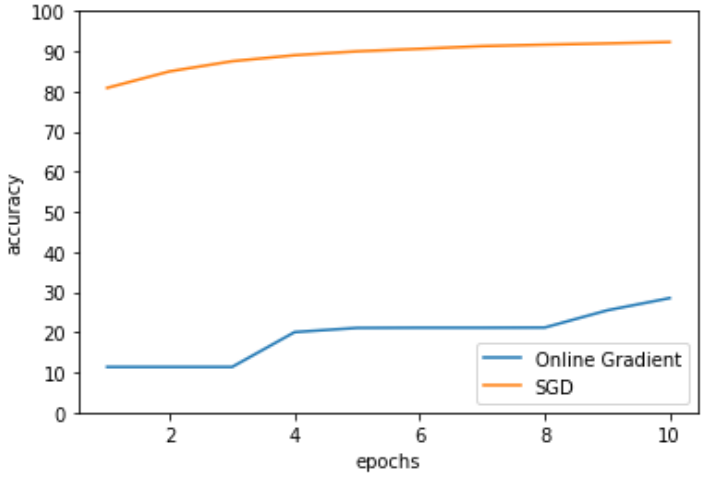
- **With Learning Rate:** 0.001



- **With Learning Rate:** 0.0001



- **With Learning Rate:** 0.00005



Q2.

Is the accuracy obtained on the test set comparable to the one that you obtain using SGD?

No, as seen in the plots, the accuracy of SGD and Online Grading are not at all comparable. This can be partially because of the sorted data that is presented during online gradient. Since the model is only used to Zero's first and then one's, two's and so on. The weights are dominated by few or one particular classes atleast until the model has seen all classes of data. This shifting in re-weighting requires lot of Epochs and the model experiences some kind of vanishing or Exploading gradient problem.

Q3.

Can the convergence analysis done for SGD in Lecture 3 be adapted to prove the convergence of online methods? Why?

No, the analysis cannot be adapted since the samples of training data in Online Methods are not at random i.e., they are not independent and identically distributed.

Exercise 4.

Set up an experiment on MNIST to compare SGD and ADAM in terms of

- **Sensibility to the learning rate (in ADAM the learning rate is the parameters that in the slides of Lecture 3 is indicated as τ and in the original article is denotes as α. In the PyTorch documentation it is called γ and in the code it is the variable that you set by specifying the argument lr of the Adam optimizer class.)**
- **Speed of convergence.**
- **Final accuracy on test set.**

- **Sensibility to the learning rate:**

Both SGD and ADAM are not very sensitive to Learning Rate but this also means that they can use higher Learning rates than other conventional models which often require very low Learning Rate.

SGD performed better for higher Learning Rates (LR) but for higher LR values like 0.4 and more, SGD became unstable after 20 Epochs.

- **Speed of convergence.**

Both SGD and ADAM converges very quickly (around 10 Epoch for SGD) but ADAM appears to be oscillating around with a small variance.

- **Final accuracy on test set.**

ADAM is very poor optimizer for MNIST dataset as it has only 10% test accuracy whereas SGD has a very superior more 90% of test accuracy

superior more 90% of test accuracy.