Advanced Deep Learning
Assingment 2

Ufuk Cem Birbiri

November 4, 2022

1 Questions and Answers

In assignment 1, I achieved the best test accuracy with 2 hidden layers with number of neurons 300 and 100, the learning rate of 0.1 and the number of epochs as 100 using Torch.nn.CrossEntropyLoss(). The network architecture can be seen in Figure 1.

```
# Now let us define the neural network we are using
net = torch.nn.Sequential(
    torch.nn.Linear(28*28, 300),
    torch.nn.ReLU(),
    torch.nn.Linear(300, 100),
    torch.nn.Sigmoid(),
    torch.nn.Linear(100, 10),
)

# Now we define the optimizer and the loss function
loss = torch.nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr = 0.1)
```

Figure 1: The network architecture from the Assignment 1, which has the best test accuracy.

It is important to emphasize that in some of the questions I plotted the test and train error curves of some networks. The test set is used like a validation set, it was NEVER used as a training data, and there was no back-propagation for test set i.e. the network didn't learn anything from the test set.

1.1 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.

Answer.1

The architecture of this question is shown in Figure 1. I chosed the learning rate as 0.1 since it achieved the best test accuracy result. I used Google Colab for every experiment in this assignment.

If we compare the results of learning rate= 0.1 within each other, we can see that the best test accuracy result(0.979) achieved with the batch size of 1000. When we use the batch sizes as 100, 10000 and 60000, the accuracy results is also sufficient(0.862, 0.972, 0.974 respectively). The batch size of 60000 is a classical GD method since we don't separate the training data to batches and we pass the entire dataset at once. When the batch size is 60000 the SGD optimizer in Pytorch is used a regular GD. We can say that batch size of 1000 has a better test accuracy than batch size of 60000.
The batch size of 1 didn't work well in this dataset where it has the lowest test accuracy(0.101). The

learning rate	batch size	test accuracy	training time(in seconds)
0.1	1	0.101	9810
0.1	100	0.862	6650
0.1	1000	0.979	775
0.1	10000	0.972	209
0.1	60000	0.974	155

Table 1: The details of different networks with train and test accuracy results.

reason can be the choice of the learning rate since we are feeding the network with the data 1-by-1, and this small learning could not reach the local minima.

We can say that a model converges, when additional training will not improve the model. We can measure the speed of convergence by analysing the train and test curves. The train and test curves of the networks with batch size 1, 1000, 10000, and 60000 are shown in Figure 2.

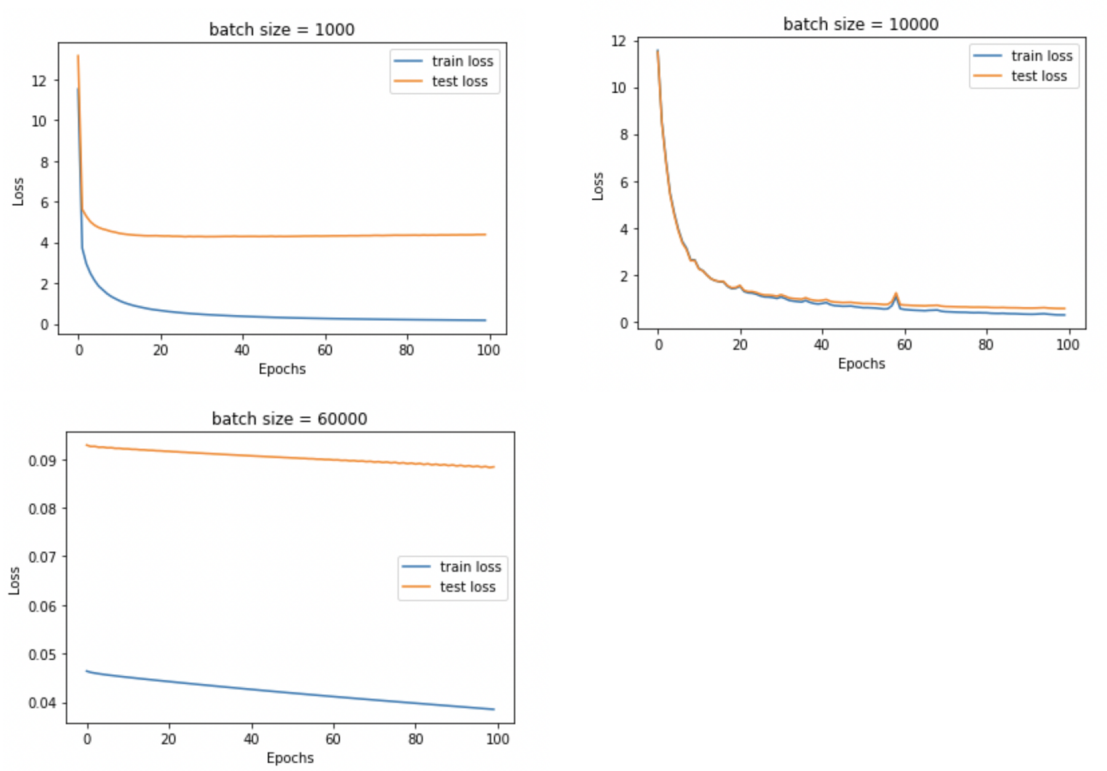

Figure 2: The training and test error curves of the networks with different batch sizes.

As we can see in the Figure 2, batch size of 1000 converges faster than the other batches i.e. it reaches the point faster when additional training does not improve the model. The SGD method updates the weights with every instance, it converges faster than GD. The GD method updates the weights only after calculating the mean loss of all the samples which is a costly operation. It will be hard to find the global minima and converge very slowly. In conclusion, we can say that smaller batch sizes converge faster than larger batch sizes.

1.2 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?

Answer.2
I initialized my network's bias and weights to zero with the code that is shown in Figure 3. The test accuracy result is 0.113, and the train accuracy result is 0.112. The test error curve and the train error curve can be seen in Figure 4.

```
#Initialize the network's weigths and bias to zero:
def init_weights(module):
  if isinstance(module, torch.nn.Linear):
    torch.nn.init.constant_(module.weight, 0)
    if module.bias is not None:
      module.bias.data.zero_()
    module.bias.data.fill_(0)
    torch.nn.init.constant_(module.bias, 0)

net.apply(init_weights)
```

Figure 3: The code snippet that initialized the weights and biases to zero in the my network.

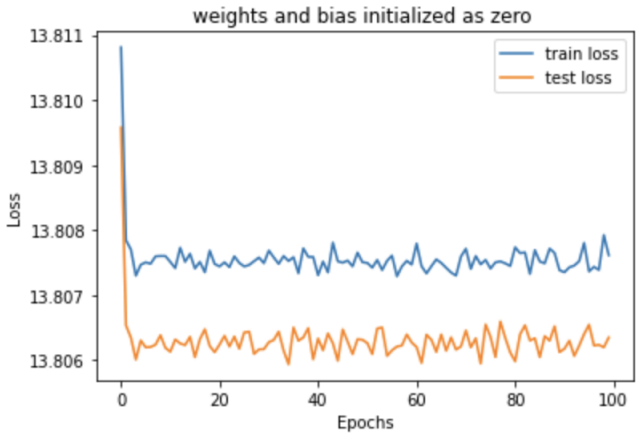

Figure 4: The test and train error curves when wee initialize the weights and bias to zero

According to the Figure 4, both of the losses didn't decrease. The test and train accuracies also are not satisfying(0.113 and 0.112 respectively). When we initialize the weights zero, the network is unable to learn anything. The reasons for that:

(i) If we use an activation function where $f(0) \neq 0$ (for example sigmoid activation), then all the weights will act together. The optimal weights that make the loss lower will not be found because all the weights are initialized as zero, therefore we will not be able to search the entire space in back-propagation to find the optimal weights.
(ii) if we have an activation function where $f(0) = 0$ (for example ReLU or Tanh) then all the outputs become 0, and the gradients of the weights become 0. The learning process will stop.
To conclude, it is not a good idea to initialize the weights to zeros since we don't enable the network to learn from the dataset. The weights acts same and they become 0.

1.3 Exercise-3

Use an Online Gradient method on the sequence of examples $(en)_{n>0}$***. Compare it with the results of a SGD method with minibatch size 1.***

- ***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.***

• ***Q2. Is the accuracy obtained on the test set comparable to the one that you obtain using SGD?***
• ***Q3. Can the convergence analysis done for SGD in Lecture 3 be adapted to prove the convergence of online methods? Why?***

Answer.3
To this exercise, I ordered the MNIST dataset in order to have first all the "zeros", then all the "ones", then all the "twos" and so on. Then I create a stream with this ordered data. This stream is not unique because the points can change indexes inside its classes. For example, all the data points that has class label 0 can have a different permutations index and they can change locations with each other without breaking the ordered labels.
After creating the stream, my network started to read the stream from beginning to end 1-by-1. First it read the 0s, then 1s, then 2s an so on. I created a for-loop in the range 60000 * 100(num epochs). I took the modula of 60000 for each ith value.
I tested the Online Gradient method with different learning rates that can be seen in Table 2. Also, the table shows two SGD methods (with batch size=1) with learning rates of and 0.1 and 0.0001 to better compare the results with Online GD.

method	**learning rate**	**test accuracy**
Online GD	0.1	0.120
Online GD	0.01	0.100
Online GD	0.001	0.293
Online GD	0.0001	0.742
SGD	0.1	0.101
SGD	0.0001	0.953

Table 2: The details of different networks with train and test accuracy results.

Q_1 Online Gradient method is converges better with smaller learning rates such as 0.0001 which has the highest test accuracy. The speed of convergence is better understood if we look at the test-train error curves. In the Figure 5, the test and train error curves of Online GD and SGD is shown. The results of SGD and Online GD with learning rate of 0.1 are not good in Table 2, so I did the second SGD experiment(learning rate=0.0001) to be able to compare with Online GD.

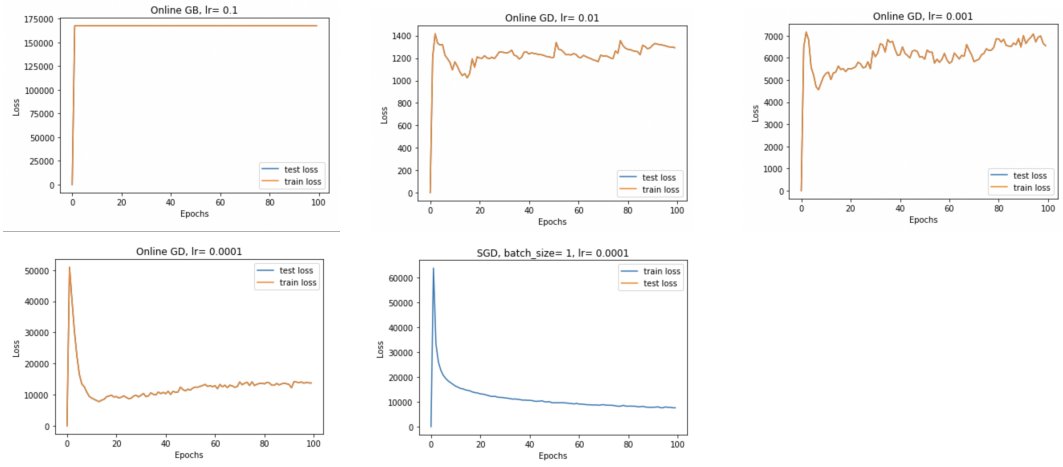

Figure 5: Test and train error curves of Online GD and SGD.

I guess we cannot talk about speed of convergence in Online Gradient method with the learning rates 0.1, 0.01, and 0.001 because they didn't converge well according to Figure 5. Actually we also cannot say that learning rate of 0.0001 in Online GD converged since the training loss decreased at the

beginning but it increases after. On the other hand, the SGD converges very well and the training error decreases. I conclude that the speed of convergence is better in SGD than Online gradient method according to these results.
In Online GD, we are not feeding the network with random data points, in contrast we are feeding each class one-by-by. So the network first learns one class members at a time in a heterogeneous way. Also in Online Gradient methods, the training data is not fixed. Online GD is not a good training technique because it is more efficient and more easy to find the global minima if we use random data points instead of ordered data.

Q_2 The accuracy of both SGD and Online GD with learning rate of 0.1 is very bad(0.101 and 0.120 respectively). The choice of 0.1 is not a appropriate learning rate for these models since it is too big and the networks cannot reach the local/global minima.
The Online GD method has the best test accuracy(0.742) with learning rate of 0.0001 which can be comparable with the SGD(0.953). But it is still lower than SGD.
Again, because we don't feed the network randomly in Online GD, the accuracy results are not sufficient. On the other hand in SGD, we are randomly using data points in which the network can convergence faster and better.

Q_3 In the convergence analysis of SGD we let $f_i : R^N \rightarrow R$ be such that for all i =1,...,m the following assumptions holds:
1. $f_i \in C^1\ (R^N; R)$
2. ∇f_i is L_i-Lipshitz; (smoothness)
3. ...
I don't write the other options since there is a problem with the number 2. We say that SGD gradient function is a convex and smooth function. In Online Gradient method it is not always the same case. Since we are not using random data points and also the training data is not fixed, we cannot assume that it will be a convex and smooth function.

1.4 Exercise-4

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

- ***Sensibility to the learning rate***
- ***Speed of convergence.***
- ***Final accuracy on test set.***

Answer.4

I tested both of the optimizers with different learning rates. The results can be seen in Table 3. I also plotted the test and train error curves to better understand the speed of convergence(Figure 6).Now let's analyse the results:

1. Sensibility to the learning rate The SGD optimizer is less sensible to different learning rates since it has sufficient test accuracies with 0.001, 0.01 and 0.1. On the other hand, the Adam optimizer has a good result only with one learning rate(0.001) according to Table 3.
We can say that SGD is more tolerable to different learning rates than Adam.

2. Speed of convergence We can say that a model converges, when additional training will not improve the model. The train and test curves of ADAM and SGD optimizers are shown in Figure 6.

It is not a good idea to compare Adam and SGD with other learning rates because Adam didn't show any convergence. We can talk about speed of convergence only in Adam optimizer network with learning rate of 0.001. We will analyse both the Adam and SGD with the learning rate of 0.001. The convergence speed is greater in Adam since it both the train and test curves reached the minima faster

optimizer	learning rate	test accuracy
Adam	0.001	0.982
Adam	0.01	0.841
Adam	0.1	0.098
Adam	1	0.160
Adam	10	0.103
SGD	0.001	0.927
SGD	0.01	0.973
SGD	0.1	0.971
SGD	1	0.552
SGD	10	0.118

Table 3: Accuracy results and learning rates different networks that use SGD or Adam optimizers.

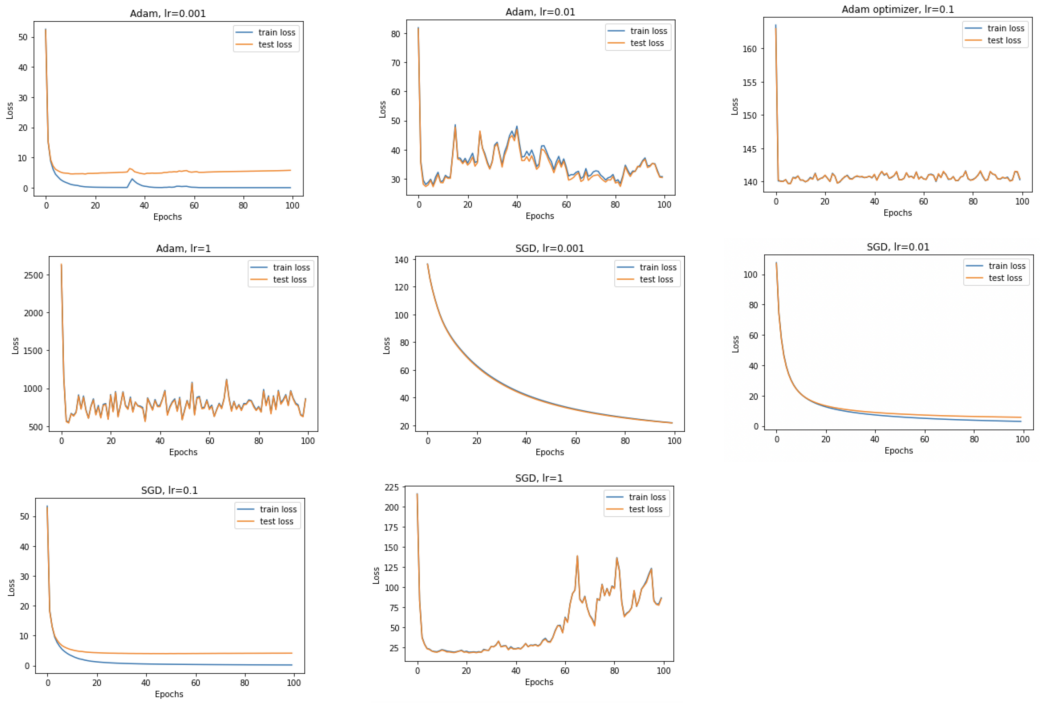

Figure 6: The training and test error curves of the network for SGD and ADAM optimizers with different learning rates.

than the SGD. We can conclude that the Adam optimizer has faster training speed than SGD and converges faster.

3. Final accuracy on test set The best accuracy of of Adam is 0.982 when learning rate is 0.001. The accuracies are so poor with the learning rates [10, 1, 0.1] of Adam that is shown in Table 3. On the other hand, the SGD optimizer is more stable with different learning rates and it has the best accuracy result(0.973) with learning rate is 0.01.
If we fix the learning rate and compare the results of Adam and SGD, we can say that generalization performance of SGD is better than Adam (except learning rate of 0.001).