IE 643: Assignment #3

Name: Vamsi Krishna Reddy Satti **Roll Number:** 160050064

Notes about Code

- All the classes nearly follow PyTorch's API. Specifically note that, CrossEntropyLoss class in my code is similar to the behavior of the same in PyTorch.
- You can run the script by using the command (assuming `IE643_160050064_main.py` is in the current directory)

  > python IE643_160050064_main.py --data [path-to-csv-data-file] --experiments 1f 1h 2b

  - Experiments valid are '1e', '1f', '1g', '1h', '2b', '2c' which correspond to the questions in the problem statement. Choose any subset of experiments to run and pass them as above in lexicographical order to train the network and generate the corresponding plots.
- The code has been tested to be working on Python 3.7.3 on a Ubuntu 18.04 machine.
- The script is silent (doesn't write anything to `stdout`) and takes around 20 minutes to run on my machine.

Observations and Answers

1(e)

The loss values of two different loss functions doesn't give a lot of insight. One thing can be said that the learning rate is quite low for MSE loss and hence it doesn't change much. The CE Loss changes but the decrease is quite minimally visible on the plot due to a relatively larger MSE loss on the plot. (More on this below). The difference in scales of the two losses is not to be taken seriously since each has their own interpretation.

1(f) and 1(g)

For the MSE Loss, we observe that learning rate of 0.1 behaves optimally among the set of learning rates considered. And in case of CE Loss, both 0.1 and 0.01 perform quite well and the better among them can only be decided by running for larger epochs. For now, 0.01 is assumed to be better for the next part 1(h).

All the other learning rates show almost no improvement in loss, indicating they are too low for any significant update in parameters of the model using backpropagation's gradient. In particular we note that CE Loss observes a quicker improvement in it's loss compared to MSE Loss across same learning rates.

1(h)

The plots in case of both MSE and CE Loss are remarkably similar. In either case, we can see that lower batch size leads to faster convergence. We strongly note here that, this is primarily due to more 'gradient updates' per epoch in case of smaller batch-sizes. Though 50 epochs are small to determine the better batch-size, in general too small batch-sizes lead to unstable optimization since the optimizer generally rely on the fact that an ideal batch of data has the same data distribution as the complete dataset, and smaller batches are prone to violate this condition more. Just from the information in the plot, 100-200 seems to be a decent choice of batch-size.

2(b) and 2(c)

For either loss functions, the learning rate must be chosen which has better convergence on validation data until the end of epochs. Also, training is usually stopped when the validation error starts increasing.

In this case, 0.1 learning rate seems to work well for both MSE and CE Loss functions. Also, lower loss functions don't show any significant improvement in terms of both training and validation error rates. Also note that the training and val error curves are almost the same, which indicates a good sign that our data split was good and they represent the same distribution.

Also we observe that MSE Loss is not a good choice for classification tasks and it has been able to achieve nearly 72% accuracy (error rate 28%) compared to CE Loss which went upto 94% in accuracy which makes sense due to theoretical probabilistic motivations for choosing CE Loss.

2(d)

Yes, they are different because of the fact that larger learning rates may easily overshoot towards local optimas rather than generalizing towards a stronger optimum. Again, a too small learning rate may get the optimizer to get stuck in feeble optimas which are possibly disparities in train data only. Again, the trade-off is handled by choosing an appropriate learning rate as seen from the validation error plots.

In this case, the learning of 0.1 seems to be the better choice for both the Losses. This is because, even though the training loss my decrease well with a particular learning rate, it my be overfitting which can be identified by checking out the validation error curves. Also note that the training and val error curves are almost the same, which indicates a good sign that our data split was good and they represent the same distribution.

My network performs with an accuracy of nearly 94% (error rate 6%) at the end of 50 epochs using CE Loss and learning rate of 0.1.