Deep Learning Assignment

Deadline : 11th November 2020

1 Tutorials

Some relevant tutorials to familiarise yourself with pytorch :

- `https://pytorch.org/tutorials/beginner/pytorch_with_examples.html`
- `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`
- `https://youtu.be/6SlgtELqOWc?t=3077`
- `https://github.com/kuangliu/pytorch-cifar` (Not a tutorial but repo might help)
- Pytorch Documentation : `https://pytorch.org/docs/stable/index.html`

Jargon you should be comfortable with : Tensors, Optimiser, Models, Autograd or Backward Pass, Activations, ...

2 XOR

2.1 Data Generation

Firstly generate 10K points in $[-1, 1] \times [-1, 1]$ using

```
np.random.seed(0)
data = 2*np.random.uniform(size=(10000,2)) - 1
```

Label points according to their quadrant. First and Third quadrant being labeled 0 and Second and Fourth quadrant being labeled 1

2.2 Exercises

1. Write a Dataset module for the XOR data (3 sets train, validation and test,70:15:15 respectively)
   - Inherit from `torch.utils.data.Dataset`
   - Define `__init__,__getitem__,__len__`
2. Define the Dataloader with batchsize of 16
   - Go through all the arguments of dataloader like `drop_last,shuffle,batch_size`
3. Define the Dataloader with batchsize of 16
4. Define the Neural Network Model
   - Inherit from : `torch.nn.Module`
   - 2 Linear layers, ReLU activation after first layer, single numeric output
   - Variable hidden layer size as input to model (4 as default hidden layer size)
5. Loss function define as `torch.nn.CrossEntropyLoss`
   - Explore other possible loss functions like `LogSoftmax + NLLLoss` vs `CrossEntropyLoss` and `MSELoss`
6. Optimizer : Use SGD optimizer with learning rate of 1e-3
   - See `zero_grad(), step()`
7. Write the main training loop and validation loops for n epochs
   - See `loss.backward(), model.forward()`
   - Use `model.train(), model.eval(),torch.no_grad()`
   - Validate for each epoch, run for 100 epochs
8. Plot the following :
   - (a) Training and Validation loss vs epoch in a single plot
   - (b) Training and Validation accuracy vs epoch in a single plot
   - (c) Best Validation loss vs Hidden Layer size (use hidden size to be (2,4,6,8,10))
   - (d) Best Validation loss vs learning rate used (use learning rates in (1e-5,1e-4,1e-3,1e-2,1e-1)) for max number of 20 epochs
   - (e) Plot test set predicted labels for best validation model. Report accuracy and loss for the same.

3 MNIST

3.1 Data Generation

Look at torchvision library on how to load MNIST dataset. Train set is of size 50K samples, test/val of 10K samples

3.2 Excercises

Look into `torch.flatten()`
Same steps as for XOR, use default hidden layer size as 128
Plot the following :

1. Training and Validation loss vs epoch in a single plot
2. Training and Validation accuracy vs epoch in a single plot
3. Best Validation loss vs Hidden Layer size (use hidden size to be (32,64,128,256,512)) for max number of 20 epochs
4. Best Validation loss vs learning rate used (use learning rates in (1e-5,1e-4,1e-3,1e-2,1e-1))

4 Extra Credit

1. Use ConvNets instead of LinearLayer for MNIST
2. Use Dropout after first layer in MNIST
3. Use alternate activations like `Softmax` and `Tanh` and present a comparative analysis
4. Use `optim.lr_scheduler.LambdaLR()`
5. Use Tensorboard to monitor loss

5 Submission Details

Create 2 notebooks for the assignment, namely `xor.ipynb` and `mnist.ipynb`. For submission download these notebooks as `.pdf` and `.py` with code and plots intermixed (make sure to generate all the plots and report all numbers instead of just the providing code for the same). Your submission should look something like this :

```
rollnumber
├─ xor.pdf
├─ mnist.pdf
├─ xor.py
└─ mnist.py
```

Zip or tar.gz this folder and name it `rollnumber.zip` or `rollnumber.tar.gz` appropriately.