CSCI 6961/4961: Homework 5

Assigned Friday November 12 2021. Due by 11:59pm Monday November 22 2021.

Create a Jupyter notebook for this assignment, and use Python 3. Write documented, readable and clear code (e.g. use reasonable variable names). Submit this notebook. You will be graded primarily based on the solutions and answers visible in the notebook, but the notebook must be runnable. You may drop to only 10 training epochs if you find 20 prohibitive, but no fewer. You may find it useful to run on Google Colab so you can use GPUs.

1. [100 points] You will evaluate, empirically, the impact of using batch normalization in training an MLP.
   - Create Pytorch classes `basicMLP`, `betweenMLP`, and `afterMLP` that implement neural networks.
     * `basicMLP` accepts 28×28 grayscale images as input, flattens them to vectors, and passes them through three fully connected hidden layers with 100 neurons each. Use tanh activations in each hidden layer. The output layer is a linear layer that returns 10 logits. You may find `nn.Sequential` convenient.
     * `betweenMLP` is a similar network, with the exception that a `BatchNorm1d` object is inserted between the `nn.Linear` and `nn.Tanh` objects in each hidden layer. This placing of the batch normalization layers was recommended in class.
     * `afterMLP` is a similar network to `basicMLP`, with the exception that a `BatchNorm1d` object is placed after the `nn.Tanh` object in each hidden layer. This placing was contraindicated in class.
   - Determine if a gpu is available, and set a variable `device` accordingly. Load the FashionMNIST train and validation (test) datasets using the `ToTensor` transformation and create dataloaders for each, using per-epoch shuffling and a batch size of 128.
   - Write a function `train_epoch(model, criterion, optimizer, trainloader)` that trains the given model for one epoch using the given optimizer and loss function criterion, with minibatches from `trainloader`. This function should first place the model in train mode and should move the minibatches to the training device.
   - Write a function `validate(model, criterion, valloader)` that evaluates the validation loss and accuracy by accumulating over the minibatches in `valloader`. This function should first place the model in evaluation mode and should move the minibatches to the training device. Return a tuple `(loss, accuracy)` of the computed loss and accuracy.
   - Create networks from each of the classes, and move them to the training device. Create Adam optimizers for each of the networks, using learning rate .01. Train each model for 20 epochs with cross-entropy loss, storing the validation losses and accuracies after each epoch.
   - Plot the validation accuracies (one figure) and losses (another figure) as line plots for each of the models; provide appropriate titles, axes labels, and legends. Use `r^`, `g*`, and `k+` as the point style for `basicMLP`, `betweenMLP`, and `afterMLP`, respectively. Use the solid line-style for all three models. Use the labels “No BN”, “BN between”, and “BN after” in the legends.
   - Comment on your observations. Also, explain why it may be problematic to set `shuffle=False` (the default setting) in your training dataloader when using batch normalization.

2. (CSCI 6961 students) [50 points]

   Hypothesis: the benefits of batch normalization are seen more with higher step sizes. Design an experiment using your code from above to investigate this hypothesis. Provide appropriate evidence supporting your (yes or no) conclusion in the form of plot(s), and explain how your experimental setup and the resulting plots helped you to reach your conclusion. Give an intuitive explanation of why this phenomenon does or does not hold.