Deep Learning Assignment 2

The objective of this assignment is to build and train a neural network using Gluon Mxnet(cpu-version)(https://mxnet.incubator.apache.org/versions/master/install/index.html) and experiment with various hyperparameters and optimizers.

References for mxnet tutorial: https://gluon.mxnet.io/

Dataset :

Data: https://goo.gl/dYFTP7

● train_images: 60000 examples of 28x28 grayscale image. The data is of size 60000 x 784. Each pixel has a single intensity value which is an integer between 0 and 255.
● train_labels: 60000 labels from 10 classes for the images in given in train_images. Class details are mentioned below.
● test_images: 10000 examples of 28x28 grayscale image. The data is of size 10000 x 784. Each pixel has a single intensity value which is an integer between 0 and 255.
● test_labels: 10000 labels from 10 classes for the images in given in test_images.

Class details are mentioned below.

Labels: Each training and test example is assigned to one of the following labels:
● 0 T-shirt/top
● 1 Trouser
● 2 Pullover
● 3 Dress
● 4 Coat
● 5 Sandal
● 6 Shirt
● 7 Sneaker
● 8 Bag
● 9 Ankle boot

Important Note: *Please make sure that the data for training and testing is placed in '../data/' directory i.e. present in the parent directory of python files being created. You don't have to submit this data files in your final submission. Also do not change the names of any files.*

Code Outline: Code outline for all files in implementation section is provided here: https://github.com/cs60010/Tutorials/blob/master/code/data_loader.py

Load Data: Function for reading data from zip files have been implemented in the code outline that is provided. It will work if all the zip files for train and test data are in the data directory.

Tasks :

Task(a) : You will have to build a neural network with the following details(use relu activation for hidden layers and softmax for output layer):

Network 1 : 5 hidden layers and 1 output layer [deep and narrow] (512,128,64,32,16)
Network 2 : 3 hidden layers and 1 output layer [shallow and wide] (1024,512,256)

Plot the comparative training loss and validation over epochs for the two networks.

Task(b) : Perform the following experiments on the network 2.

Experiment 1: Use Normal, Xavier and Orthogonal initialization.
Experiment 2: Use Batch Normalization
Experiment 3: Use Different regularization techniques: Dropout(0.1, 0.4, 0.6)
Experiment 4: Use different optimizer SGD, Nesterov's accelerated momentum, AdaDelta, Adagrad, RmsProp, Adam. **NOTE:** You cannot use SGD and Nesterov's accelerated momentum APIs of Mxnet. You have to write separate modules for these optimizers. You are allowed to use the API for others optimization and compare the results with your implemented optimizers.

Plot the comparative training loss over epochs for 4 experiments vs vanilla netowrk 2.

Task(c) : Use the fully trained model corresponding to *Network 2* for this task. After training the above NN, get the hidden layer representation of layer 1 for all the data-points and use it as input for training a logistic regression based classifier for the data. Use the same train-test split for training the logistic regression classifier. Report the results on the test set for it. Similarly do the same for hidden layers 2 and 3. For implementing logistic regression you can directly use Scikit Learn Logistic Regression classifier.
[1]http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
[2] https://gluon.mxnet.io/chapter02_supervised-learning/softmax-regression-scratch.html
Your logistic regression classifier should take the layer activations as features and the imageclass labels as ground truth labels. Train the classifier on your dataset's training set and test on the dataset's test set.

Submit a brief report on what inferences you could draw from Tasks and Experiments. (in pdf only). Explanations should be short (1-3 sentences).

Constraints and Considerations:

1. You should not submit data. Always create a train-validation split (70:30) on the training set and train and test on appropriate data only. You must train properly with proper stopping criterion. All decisions you take for model parameters must be logical.
2. Use of Python3 is preferable.

3. Include all training and testing codes. The files should be named as <RollNo>_Assignment2_task_<a/b/c>.py' must be present in your “code/” folder. It takes in some command line arguments. ‘**python 17CS72P03_Assignment2_task_a.py --train**’ should train the model, ‘**python 17CS72P03_Assignment2_task_a.py --test**’. should load all model weights, test the model and report the accuracy on the test data.
4. Store all weights of your model in "weights/" folder. The weights folder must be inside the directory in which you are coding and must be submitted to Moodle.
5. If the weights are exceeding the upload constraints imposed by Moodle (may not happen), then upload the weights in some public site like github and download them first, by writing appropriate code for it.
6. Name the ZIP file as well the folder that you compressed to get the zip as "<Roll No.>_Assignment2.zip". Upload the zip file with codes, weights and report to moodle. Note that the zip file should following structure:

    17CS72P03_Assignment2
        |-- code
        |-- weights
        |-- report