CS 763 (Spring 2019): Assignment 3

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

[All codes have been tested to be working on **PyTorch 1.0.0 on Python 3.6.8** running on Windows OS]

General Notes

1. Momentum (including an optional `nesterov` arg) has been implemented and used while training.
2. For `torchfile` package to be compatible with Python3, change the `xrange` string occurance in the source file of `torchfile` to `range`.
3. With the exception of `bestModel/model.bin`, rest all the saved `.bin` files through *any* of the scripts will contain numpy arrays / list of numpy arrays. [Specifically, note that these are not PyTorch tensors, even through my code is written using PyTorch tensors]
4. My code design choices *i.e.* the API is highly derived / inspired from PyTorch

List of Implemented Modules

Please refer to the docs of PyTorch for detailed explanations of what each functions does.

Convolution layers

1. `layers.Conv2d(input_size, in_channels, out_channels, kernel_size, stride=1)`

Pooling layers

1. `layers.MaxPool2d(kernel_size, stride=None)`
2. `layers.AvgPool2d(kernel_size, stride=None)`

Non-linear activations

1. `layers.ReLU()`
2. `layers.PReLU(num_parameters=1, init=0.25)`

Normalization layers

1. `layers.BatchNorm1d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=False)`
2. `layers.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=False)`

Linear layers

1. `layers.Linear(input_size, output_size)`

Dropout layers

1. `layers.Dropout(p)`

Optimizers

1. `optim.SGD(model, lr, momentum=0, dampening=0, weight_decay=0, nesterov=False)`
2. `optim.Adam(model, lr=1e-3, betas=(0.9, 0.999), eps=1e-8, weight_decay=0)`

Learning rate schedulers

1. `optim.StepLR(optimizer, step_size, gamma=0.1, last_epoch=-1)`
2. `optim.MultiStepLR(optimizer, milestones, gamma=0.1, last_epoch=-1)`
3. `optim.ExponentialLR(optimizer, gamma, last_epoch=-1)`
4. `optim.CosineAnnealingLR(optimizer, T_max, eta_min=0, last_epoch=-1)`

Data Loaders

1. `util.DataLoader(dataset, batch_size=1, shuffle=False)`

Utility Layers

1. `layers.Flatten()`

Loss functions

1. `criterion.CrossEntropyLoss()`