DS-GA 1008: Deep Learning, Spring 2020
Homework Assignment 3

Due: Tuesday, March 31, 2020 at 11:59pm

`If you get, give. If you learn, teach. Maya Angelou (1928 - 2014)`

1. Fundamentals

1.1. Convolutions

For the following transformations from dimensionalities A to B, specify any combination of convolutions (with particular choices of kernels, stride), pooling, and other related modules that could get you from dimensionality A to B. [3 pts]

1. A:$(64, 111, 111) \rightarrow$ B:$(4, 234, 234)$
   We first apply a pooling with module MaxPool2d with kernel size=3,stride=3, no padding, and then apply a fractionally-strided convolution (deconvolution) with module ConvTranspose2d with input channels 64, output channels 4, kernel size=7, stride=4, and no padding. Then we apply UpSample with a scale factor 1.55.

2. A:$(256, 56, 56) \rightarrow$ B:$(14, 126, 126)$
   We first apply a pooling with module MaxPool2d with kernel size=3,stride=3, no padding, and then apply a fractionally-strided convolution (deconvolution) with module ConvTranspose2d with input channels 256, output channels 14, kernel size=7, stride=4, and padding=5.

3. A:$(256, 56, 56) \rightarrow$ B:$(42, 72, 72)$
   We first apply a fractionally-strided convolution (deconvolution) with module ConvTranspose2d with input channels 256, output channels 42, kernel size=5, stride=4, and padding=4. Then apply a pooling with module MaxPool2d with kernel size=3,stride=3, no padding,

1.2. Dropout

Dropout is a very popular regularization technique.

(a) List the `torch.nn` module corresponding to 2D dropout. [2 pts]
nn.Dropout, nn.AlphaDropout accept inputs with any shape. nn.Dropout2d accepts shape of (Number of samples,number of Channels,Height,Width).

(b) Read on what dropout is and give a short explanation on what it does and why it is useful. You may find this chapter https://www.deeplearningbook.org/contents/regularization.html and the original paper helpful. [6 pts]
Training neural network on a large number of parameters is prone to cause overfitting. Just like the intuition behind the random forest, building an ensemble of neural networks with various configurations might reduce overfitting. However, it is impractical

to create this ensemble. Instead, we drop out, i.e. deactivate neuron nodes at random with a possibility associated with each node in each training iteration. We generate a different neural network configuration by deactivating some nodes. Dropout, therefore, simulates an ensemble of the neural network, and thus reduces the overfitting and decreases the generalization error.

1.3. Batch Norm

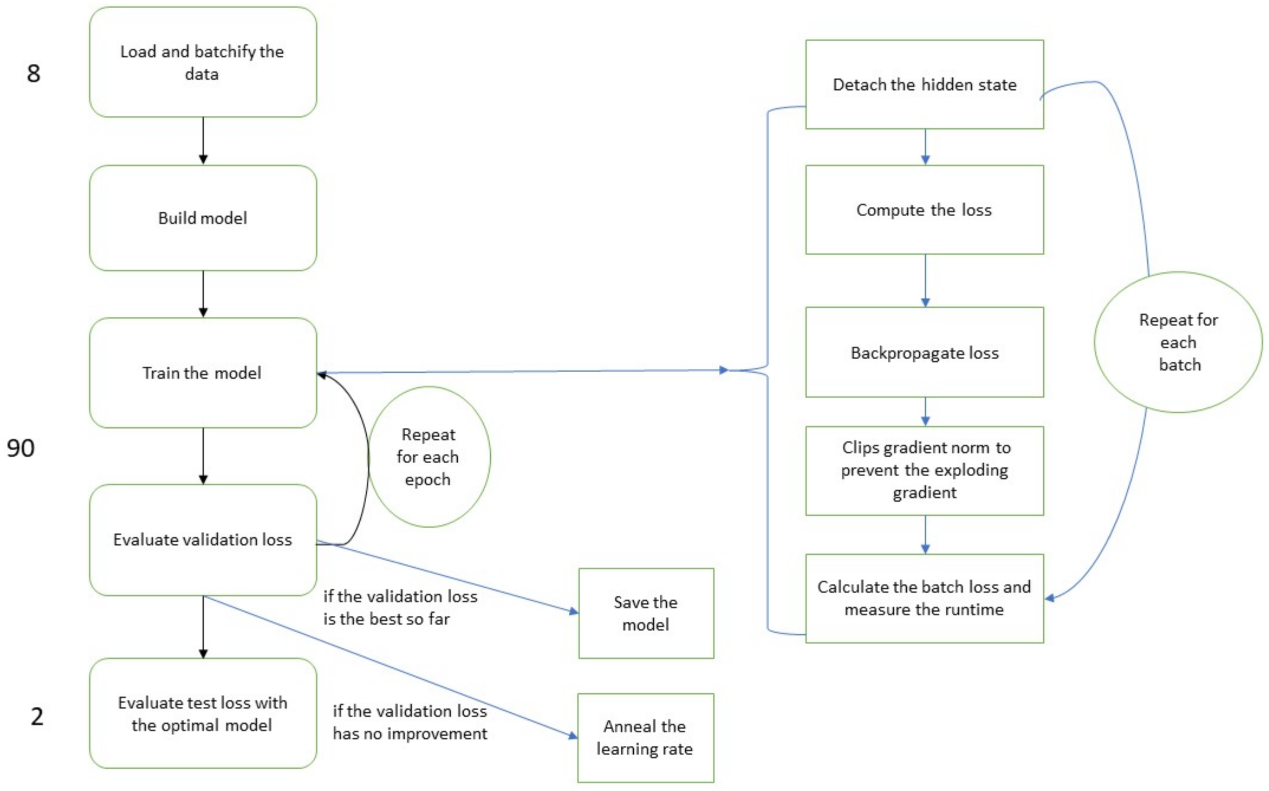
(a) What does mini-batch refer to in the context of deep learning? [3 pts]
We might run out of the memory when we train the neural network on a large amount of data each epoch. We might get a noisy result if we train the neural network on just on one instance per epoch, which is also computationally expensive. To balance this trade-off, we train a subset of the data in each epoch, and this subset is called mini-batch. The mini-batch size is the number of training instances in each epoch.

(b) Read on what batch norm is and give a short explanation on what it does and why it is useful. You might find this blog post and the original paper helpful. [6 pts]
If the features are not in a consistent range, then the associated weights will also have an uneven distribution. Thus, when we do back propagation, the size of update will also be inconsistent. As a result, the learning algorithm might keep oscillating until converging to the minima. After normalization, the inputs are on the same scale, same to the weights, which accelerates the training. In the neural network, we normalize the output of the activation function in each hidden neuron on each batch of instances through subtracting the batch mean and dividing it by the batch standard deviation. This reduces the covariance shift and thus improves the neural network stability. Batch normalization also make the neural network layers more independent of each other. It also reduces overfitting via regularization, so we usually combine batch normalization with drop out.

2. Language Modeling

This exercise explores the code from the `word_language_model` example in *PyTorch*.

(a) Go through the code and draw a block diagram / flow chart (see this tutorial on flowcharts in latex) which highlights the main interacting components, illustrates their functionality, and provides an estimate of their computational time percentage (rough profiling). [5 pts]


8
Load and batchify the data
Build model
Train the model
90
Evaluate validation loss
Repeat for each epoch
2
Evaluate test loss with the optimal model
if the validation loss is the best so far
Save the model
if the validation loss has no improvement
Anneal the learning rate
Detach the hidden state
Compute the loss
Backpropagate loss
Clips gradient norm to prevent the exploding gradient
Calculate the batch loss and measure the runtime
Repeat for each batch

If we have 100 units of total time, then the most computation time (around 90 percents I guess) will be spent on training and evaluating the model. The loading and processing data runtime depends on the dataset size. Here I estimate it to be 8 percent of the total time.

(b) Find and explain where and how the back-propagation through time (BPTT) takes place (you may need to delve into *PyTorch* source code). [5 pts]
From the documentation of the backward function in the torch/autograd/__init__.py, it says that "the graph is differentiated using the chain rule. If any of tensors are non-scalar (i.e. their data has more than one element) and require gradient, then the

Jacobian-vector product would be computed, in this case the function additionally requires specifying grad tensors. It should be a sequence of matching length, that contains the "vector" in the Jacobian-vector product, usually the gradient of the differentiated function w.r.t. corresponding tensors." In pytorch, if we call forward, a new graph with new memory is generated. If we call backward(), as the gradients are computed, the graph is destroyed.

(c) Describe why we need the `repackage_hidden(h)` function, and how it works. [5 pts]
At the beginning of each batch, we detach the hidden state from their history to prevent the model from backpropagating all the way to the start of the dataset. This also frees up memory for the following iteration. In the repackage hidden function, we detach all of the tensors in a hidden layer to stop keeping track of subsequent gradients.

(d) Why is there a `--tied (tie the word embedding and softmax weights)` option? [5 pts]
Based on the paper, tying together the input and the output matrices by reusing the input word embedding matrix as the output projection matrix will greatly decreases the number of parameters, i.e. the size of the neural network, without harming the performance. ("Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling" (Inan et al. 2016)). Moreover, as the number of parameter decreases, the model is less likely to be overfitting.

(e) Compare LSTM and GRU performance (validation perplexity and training time) for different values of the following parameters: number of epochs, number of layers, hidden units size, input embedding dimensionality, BPTT temporal interval, and non-linearities (pick just 3 of these parameters and experiment with 2-3 different values for each). [5 pts]
Number of epochs:
Running LSTM and GRU with 15 epochs in the default setting, for the last epoch, we get
LSTM: time: 83.78s ,valid loss 4.81 , valid ppl 122.91,
GRU: time: 82.70s , valid loss 5.11, valid ppl 164.98.
epoch=5: LSTM: time: 82.65s , valid loss 5.04 , valid ppl 154.82.
GRU: time: 82.78s , valid loss 5.51 , valid ppl 245.96
epoch=10: LSTM:time: 83.60s , valid loss 4.83 , valid ppl 125.04.
GRU: time: 82.76s, valid loss 5.19 ,valid ppl 179.54.
The time of LSTM and GRU are close, and the time is consistent as the number of epoch varies. LSTM generally performs better than GRU,i.e. LSTM has lower validation loss and ppl.
Number of layers with epoch=10:
Number of layers=4:
LSTM: time: 111.55s , valid loss 4.96 , valid ppl 142.16
GRU: time: 110.52s — valid loss 7.08 — valid ppl 1188.84

Number of layers=2:Please refer to the result of epoch=10 above.
When there are 4 layers, the validation ppl of GRU becomes incredbly large. LSTM generally performs better than GRU,i.e. LSTM has lower validation loss and ppl.
Size of word embeddings with epoch=10:
size of word embeddings=200: Please refer to the result above.
size of word embeddings=300:
LSTM: time: 88.12s , valid loss 4.97 , valid ppl 144.47
GRU: time: 86.83s , valid loss 5.15 , valid ppl 173.19
size of word embeddings=600:
LSTM: time: 98.98s ,valid loss 4.82, valid ppl 124.56
GRU: time: 97.43s , valid loss 5.07 ,valid ppl 159.34
The time of LSTM and GRU are close, and it takes longer time to run the model as the number of epoch increases. The validation loss for both LSTM and GRU decreases as word embeddings size increases. LSTM generally performs better than GRU,i.e. LSTM has lower validation loss and ppl.

(f) Why do we compute performance on a test set as well? What is this number good for? [5 pts]
We use the validation set to evaluate the model that fit on the training set while tuning model hyperparameters. Based on the validation loss, we tune parameters such as learning rate and select the optimal configuration with the best validation loss. Therefore, validation set is biased for us to measure the generalization error, we need to compute performance on the test set, which provides an unbiased evaluation of the ultimate tuned model.

DS-GA 1008: Deep Learning, Spring 2020
Homework Assignment 3

Due: Tuesday, March 31, 2020 at 11:59pm

3. PyTorch

If you haven't already, install most recent versions of Python (3.6 or higher), PyTorch (we recommend using conda for the installation), and Jupyter.

Complete the programming exercises provided in the homework 3 directory of the course Google Drive folder (link).

4. Evaluation

Homework is worth a total of 100 points.

- Part 1 - 50 points
- Part 2 - 50 points

5. Submission

You are required to write up your solutions to Part 1 using markdown or LaTeX.
Please submit the homework on the NYU classes assignment page. Please upload the following:

- `First-name_Last-name_netID_A3.tex` (or `.md`) file for Part 1
- `First-name_Last-name_netID_A3.pdf` file for Part 1
- `First-name_Last-name_netID_A3_code.ipynb` file for Part 2
- `First-name_Last-name_netID_A3_code.pdf` file for Part 2
  (you can use Jupyter Notebook's "File → Download as → PDF" feature)

6. Disclaimers

You are allowed to discuss problems with other students in the class but have to write up your solutions on your own.
As feedback might be provided during the first days, the current homework assignment might be undergoing some minor changes. We'll notify you if this happens.