CS324: Deep Learning
Assignment 3

Jianguo Zhang

May 2, 2020

1 Part I: PyTorch LSTM (40 points)

As you've seen in the previous assignment, RNNs are unable to memorise long sequences. So, in this first part of assignment 3, you will improve on your palindrome task by implementing an LSTM instead. The file **dataset.py** contains the class PalindromeDataset. It's exactly the same as in the previous assignment, but I've included it here again for your convenience. You will use this dataset to sample the mini-batches and train your LSTM. Your LSTM will have to follow the structure defined by the equations:

$$g^{(t)} = \tanh(W_{gx}x^{(t)} + W_{gh}h^{(t-1)} + b_g) \tag{1}$$

$$i^{(t)} = \sigma(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + b_i) \tag{2}$$

$$f^{(t)} = \sigma(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + b_f) \tag{3}$$

$$o^{(t)} = \sigma(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + b_o) \tag{4}$$

$$c^{(t)} = g^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)} \tag{5}$$

$$h^{(t)} = \tanh(c^{(t)}) \odot o^{(t)} \tag{6}$$

$$p^{(t)} = (W_{ph}h^{(t)} + b_p) \tag{7}$$

$$\tilde{y}^{(t)} = \text{softmax}(p^{(t)}), \tag{8}$$

where $\odot$ denotes the element-wise multiplication. As you see, the structure is similar to that of the RNN implemented in the previous assignment, but with the addition of three gates: the input gate i, the forget gate f, and the output gate o (and the input modulation g). As in the previous assignment, initialise $h^{(0)}$ to the vector of all zeros and compute the cross-entropy loss over the last time-step, i.e.,

$$\mathcal{L} = -\sum_{k=1}^{K} y_k \log(\tilde{y}_k^{(T)}), \tag{9}$$

where k runs over the classes ($K = 10$ digits in total), y_k is a one-hot encoding vector.

1.1 Task 1

Implement the LSTM without using *torch.nn.LSTM*. Follow the skeleton provided in **train.py** (for the training) and **lstm.py** (to define the model). For the forward pass you will need to use a *for* loop to step through time and apply the recurrence equations that define the network behaviour. For the backward pass you can rely on Pytorch automatic differentiation and use the RMSProp optimiser for tuning the weights.

1.2 Task 2

Given the LSTM implemented in Task 1 and a palindrome of length T, the network should be able to predict the T-th digit given the preceding $T-1$ ones. You should be able to obtain close to perfect accuracy with $T = 5$ and the default parameters provided in the python files. Note that you might need to adjust the parameters, particularly the learning rate, when increasing the sequence length. You should observe a better performance when compared to the RNN you've implemented in the previous assignment.

2 Part II: Generative Adversarial Networks (60 points)

In the second part of this assignment you will implement a GAN (see slides for the GAN lecture) that generates images similar to those of the training set. You will use a standard Normal distribution as the noise distribution. As we've seen in the lecture, training the GAN involves playing a minmax game between the generator and discriminator. In other words, our optimization objective is

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_p[\log D(X)] + \mathbb{E}_q[\log(1 - D(G(Z)))] \tag{10}$$

2.1 Task 1

Build your GAN in PyTorch using the template `my_gan.py`. Train it on the MNIST dataset provided in the `dataset` folder (the python file inside can be used to download the data and create the traninig and test sets).

2.2 Task 2

Sample 25 images from your trained GAN and include these in a jupyter notebook. Do this at the start of training, halfway through training and after training has terminated.

2.3 Task 3

Sample 2 images from your GAN (make sure that they are of different classes). Interpolate between these two digits in latent space and include the results in your jupyter notebook. Use 7 interpolation steps, resulting in 9 images (including start and end point).

3 Submission instructions

The submission will include:

- A written report describing what you did, the results and your analysis (please use words or latex to write the report.).
- Code for producing **all** results for all parts and tasks.
- Instructions on how to run the code (a jupyter notebook for running instructions will be fine).

Create a ZIP archive with the submission of Assignment 3 (all parts and tasks). Give the ZIP file the name **studentnumber_assignment3.zip**, where you insert your student number. Please submit the archive through the Blackboard.

Make sure all files needed to run your code are included or you may be given 0 points for it.

The deadline for assignment 3 (all parts and all tasks) is the 29th of May 2020 at 23:55 (Beijing Time).