CS324: Deep Learning
Assignment 3

Luca Rossi

April 25, 2019

1 Part I: PyTorch LSTM

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 four gates: the input modulation gate g, the input gate i, the forget gate f, and the output gate o. 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 Submission instructions

Create a ZIP archive with the results 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 Sakai. 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 19th of May 2019 at 23:55 (Beijing Time).