Deep Learning Assignment 2

Jochem Loedeman
12995282

1 Recurrent Neural Networks

1.1 Vanilla RNNs

Question 1.1

(a)

$$\frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{W}_{ph}} = \frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{p}^{(T)}} \frac{\partial \boldsymbol{p}^{(T)}}{\partial \boldsymbol{W}_{ph}}$$

In the computational graph, there is no path from outputs of recurrent hidden states to the loss at T. Therefore, no sum over previous states is present in the expression for the gradient.

(b)

$$\frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{W}_{hh}} = \frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{p}^{(T)}} \frac{\partial \boldsymbol{p}^{(T)}}{\partial \boldsymbol{h}^{(T)}} \frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{W}_{hh}} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{p}^{(T)}} \frac{\partial \boldsymbol{p}^{(T)}}{\partial \boldsymbol{h}^{(T)}} \frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{h}^{(i)}} \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{W}_{hh}}$$

$$= \sum_{i=0}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial \boldsymbol{p}^{(T)}} \frac{\partial \boldsymbol{p}^{(T)}}{\partial \boldsymbol{h}^{(T)}} \left(\prod_{j=i+1}^{T} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{h}^{(j-1)}} \right) \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{W}_{hh}}$$

In this case, there are paths from nodes in previous timesteps computed with $\boldsymbol{W}_{hh}$ to the loss at T. Therefore, we should account for these dependencies through the chain rule, giving rise to sums and products within the expression for the gradient.

(c) The first and second gradients are respectively independent and dependent on previous timesteps, as was already explained through the connectivity of the computational graph. When training this network for a large number of timesteps, we have to account for very long-term dependencies of the recurrent states through the product in the expression for the second gradient. This product of Jacobians can cause the total gradient to become very small or very large, making learning difficult.

1.2 Long Short-Term Memory (LSTM) network

Question 1.2

(a)

- *Input modulation gate:*
  Proposes an updated cell state. This gate has a tanh activation, which is important to keep the activations of the internal state zero-centered. If for example we used the sigmoid, these states would increase over time.

- *Input gate:*
  Regulates how much of the new value proposed by the input modulation gate is actually used in the updated cell state. Uses a sigmoid to obtain a gating value between 0 and 1.

- *Forget gate:*
  Regulates how much of the previous cell state is retained in the updated cell state. Uses

a sigmoid to obtain a gating value between 0 and 1, which corresponds to completely forgetting or strongly remembering the previous state.

- *Output gate:*
  Regulates how much of the nonlinearized current cell state is passed as output of the cell. Uses a sigmoid to obtain a gating value between 0 and 1.

(b)

$$N_{\text{total}} = 4\underbrace{\left(N_{\text{hidden}} \cdot N_{\text{input}} + N_{\text{hidden}} \cdot N_{\text{hidden}} + N_{\text{hidden}}\right)}_{\text{gate input weights, recurrent weights and biases}}$$
$$+\underbrace{N_{\text{output}} \cdot N_{\text{hidden}}}_{\text{output weights}} + \underbrace{N_{\text{output}}}_{\text{output bias}}$$

1.3 LSTMs in PyTorch

Question 1.3

The required LSTM model was implemented in PyTorch, and evaluated using the supplied `train` function and the binary palindrome data. Accuracies were obtained for sequence lengths of 10 and 20, where each hyperparameter setting was evaluated for 3 different random seeds (0, 1 and 2). From these results, the mean and standard deviation for each accuracy value was obtained. This yielded two accuracy curves, one for each sequence length setting, together with the corresponding standard deviations. They are shown in Figure 1, where the shaded areas correspond to the accuracy values that lie within 1 standard deviation on either side of the obtained accuracy value. Notice that for $T = 20$ the standard deviation increases strongly after 1000 steps. At this point, some training runs make a large jump in accuracy, whereas others fall behind. This behaviour seems to be characteristic of the model on this data; the accuracy is stable around 0.55 when suddenly, the model improves relatively quickly to perfect accuracy.

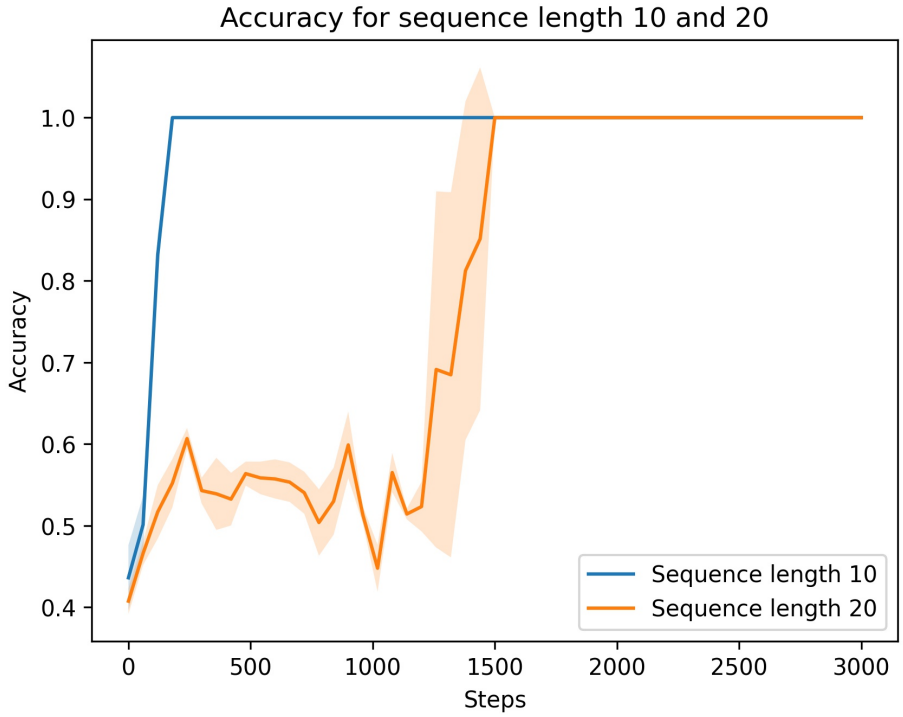

Figure 1: Accuracy of the LSTM on the binary palindrome dataset for $T = 10$ and $T = 20$

Question 1.4

The peepLSTM model was implemented in PyTorch, using the same hyperparameter settings as before. The results are shown in Figure 2. For $T = 20$ the model converges to perfect accuracy already around 500 steps, which is significantly faster than the original LSTM model. Furthermore, the uncertainty of the accuracy values is much smaller. Apparently, setting the gate values based on the cell states rather than the LSTM outputs is beneficial for the performance of the model on this data set.

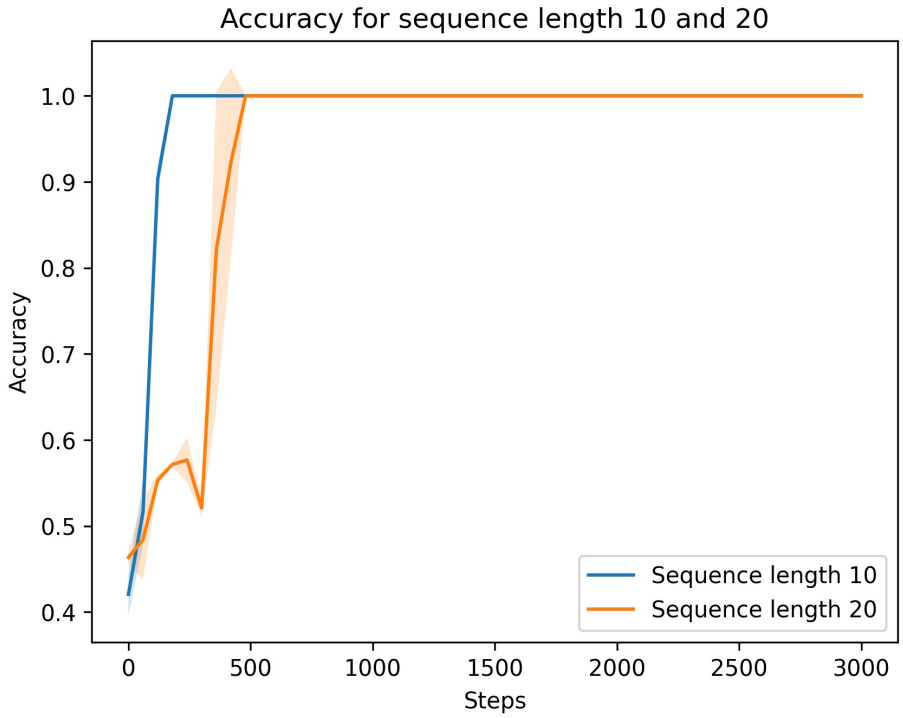

Figure 2: Accuracy of the peepLSTM on the binary palindrome dataset for $T = 10$ and $T = 20$

2 Recurrent Nets as Generative Model

Question 2.1

(a)