Chapter 10

Questions for Sequence Modelling: Recurrent and Recursive Nets

10.1 Questions for Unfolding Computational Graphs

10.1.1

The authors state that *"Much as almost any function can be considered a feed forward neural network, essentially any function involving recurrence can be considered a recurrent neural network."* Do you agree with this statement?

10.1.2

When predicting the future from the past, an RNN learns to use its hidden state to summarize the history of observations. This is in general a necessarily lossy summary, since the history sequence can be any length and the hidden state is finite. Give an example when this is irrelevant for the prediction task. Give an example when this make a big Difference.

10.1.3

The unfolded view of a neural network illustrate how information flows forward in time. Is this necessarily the same time direction as in the data (i.e. from an observed time-series)? If yes, explain why. If no, give a counter-example.

10.1.4

If we were to imagine neural networks as implementing algorithms using a Control-Flow Graph, how would you characterize the difference between the GFG implemented by a Feed forward Neural Networks compared to a Recurrent Neural Networks?

10.2 Questions for Recurrent Neural Networks

10.2.1

The authors assume hyperbolic tangent units, not ReLU which in earlier chapters has been the most common one. Why do you think that is?

10.2.2

What is the fundamental difference between the two deep RNNs described by the equations 10.1 and 10.2 below?

$$\begin{aligned}
\boldsymbol{h}_1^{(t)} &= \tanh(\boldsymbol{b}_1 + W_1 \boldsymbol{h}_1^{(t-1)} + U_1 \boldsymbol{x}^{(t)}) \\
\boldsymbol{h}_2^{(t)} &= \tanh(\boldsymbol{b}_2 + W_2 \boldsymbol{h}_2^{(t-1)} + U_2 \boldsymbol{h}_1^{(t)}) \\
&\vdots \\
\boldsymbol{h}_l^{(t)} &= \tanh(\boldsymbol{b}_l + W_l \boldsymbol{h}_l^{(t-1)} + U_l \boldsymbol{h}_{l-1}^{(t)}) \\
\boldsymbol{o}^{(t)} &= c + V \boldsymbol{h}_l^{(t)}
\end{aligned} \tag{10.1}$$

$$\begin{aligned}
\boldsymbol{h}_1^{(t)} &= \tanh(\boldsymbol{b}_1 + W_1 \boldsymbol{x}^{(t-1)} + U_1 \boldsymbol{x}^{(t)}) \\
\boldsymbol{h}_2^{(t)} &= \tanh(\boldsymbol{b}_2 + W_2 \boldsymbol{h}_1^{(t-1)} + U_2 \boldsymbol{h}_1^{(t)}) \\
&\vdots \\
\boldsymbol{h}_l^{(t)} &= \tanh(\boldsymbol{b}_l + W_l \boldsymbol{h}_{l-1}^{(t-1)} + U_l \boldsymbol{h}_{l-1}^{(t)}) \\
\boldsymbol{o}^{(t)} &= c + V \boldsymbol{h}_l^{(t)}
\end{aligned} \tag{10.2}$$

10.2.3

Below (10.3) is a reproduction of equation 10.14 from the book. Under what independence assumptions is it modelling the joint probability $P(y^{(1)}, y^{(2)}, \ldots, y^{(1)})$?

$$-\sum_t \log p_{\text{model}}(y^{(t)} | \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\} \tag{10.3}$$

10.2.4

A common setup for training RNNs on sequence data (text and time-series) is to use next-step prediction (see equation 10.4 below). In what way (if any) is this different from the network described by figure 10.4 of the book, and is it an application of teacher forcing?

$$
\begin{aligned}
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{b} + W\boldsymbol{h}^{(t-1)} + U\boldsymbol{x}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + V\boldsymbol{h}^{(t)} \\
\boldsymbol{y}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}) \\
L^{(t)} &= -\sum_{c=1}^{C} \mathbf{1}_{\boldsymbol{x}^{(t+1)}} \log p(y_c^{(t)})
\end{aligned}
\tag{10.4}
$$

(Some notes on the loss, in this case we assume that $\boldsymbol{x}$ is a vector encoding of discrete values from the set C, if the next $\boldsymbol{x}$ is c, the indicator function $\mathbf{1}_{\boldsymbol{x}^{(t+1)}}$ takes the value 1, otherwise 0. This means that the loss will only be for the value of $\boldsymbol{y}^{(t)}$ which corresponds to the probability of the next value of $\boldsymbol{x}$)