Saaif Ahmed

saaifza2@illinois.edu

Chapter 3 Reflection

1. Please give a brief summary of the chapter?

The chapter begins by introducing the concept of RNN. In the scenario where we have a neural network specifically a DNN, it is difficult to incorporate data over time. Or another way is to say a DNN cannot effectively predict through time, as new data comes in. For example, it is hard for a DNN to have a user predict if they will have a heart attack without updating the model's input parameter each time, they update the EHR. This is because the DNN structure is not dynamic, so incorporating a changing element over time is impossible. But for this we have an RNN or a recurrent neural network. This RNN is focused on outputting states of prediction after a certain number of epochs. In the RNN model, for a given step, that output is then fed into the next step to create the next output in the series. Doing this creates some problems as the gradients between steps of the RNN can become excessively large or incredibly small, known as the exploding and vanishing gradient problem. The way to mitigate these changes is the discussed implementation of the RNN.

The chapter starts with the LSTM model and a breakdown of the forward mathematics. The LSTM includes a forget gate and a cell gate in addition to the standard input and output gates of a RNN. These two additional gates capture extended dependencies among events that run back-to-back. This can prevent the previously mentioned problems with RNN and their gradients. The LSTM is a complex implementation of an RNN, and there is a less complex more efficient option available. The chapter then discusses the GRU implementation. In this way, the forget gate and the input gate are joined together and a unified hidden representation to represent the state is created. Both are used in the real world and have similar performance.

An RNN has the architecture to handle different sequential scenarios such as Many to One, One to Many, and Many to Many. For example, a "Many to Many" scenario could be a RNN taking weeks sequentially to determine if a patient will have a heart attack that week. That's multiple inputs that provide multiple outputs in a sequence. To train an RNN we partition out the long sequence into smaller steps such as 20 segments, and each segment has their gradient calculated in parallel. This step is handled by modern day tools.

2. What improvements do you want to see in this chapter? Please elaborate on them

In the chapter I would like to see a more in-depth description of the math. If the chapter is going to reference the sigmoid functions numerous times throughout the chapter, it is fine to simplify it to Sigmoid() and the same for Tanh(), however at least once it would be beneficial to write out what the sigmoid function is in pure algebra. This would make it easier for new readers to internalize and appreciate the math going on in these calculations. Especially early on, readers need to be familiar with Sigmoid() to understand it will output 1 or 0, and in chapter 3 readers may not have internalized that yet. In addition, I would have liked to see a back propagation

calculated out, even it is a simple RNN. This would help people realize that finding the gradient through time is incredibly difficult and help people figure out how the segment splitting worked.

3. What are the typos in this chapter?

- Page 14 of the pdf. “Using the provided dataset…” has weird spacing. This sentence should be rephrased so that this spacing doesn’t occur. It’s very jarring for the reader.

No other changes were noticed.

4. Which part of the chapter do you like most?

The part of the chapter I liked the most is the Extension and Training the RNN Model section. Showing how the RNN can be implemented to handle different use case scenarios that a normal DNN cannot handle was helpful. All tools were created to solve a problem, and without an apt description of the problem, the full usage of a tool cannot be discovered. Having the textbook show the reader the scenarios that an RNN was made to solve generated a deeper understanding of the material.

5. What are the most useful things you learned from this chapter?

The most useful part of this chapter was the code blocks. As we get deeper into the study of neural networks and deep learning techniques, knowing how to use the tools of study effectively is a great asset. PyTorch is an extremely powerful library, and it bears mentioning that seeing example code for a realistic example task helps students understand how to use this software for their own studies. The example at the end of the chapter which walks through an entire example is perfect for explaining how one uses PyTorch to make an RNN.

References

Xiao, C., Sun, J. (2021). Introduction. In: Introduction to Deep Learning for Healthcare. Springer, Cham. https://doi-org.proxy2.library.illinois.edu/10.1007/978-3-030-82184-5_1