RECURRENT NEURAL NETWORKS (RNNS): A GENTLE INTRODUCTION AND OVERVIEW

Nguyen Hoang Tan
Department of Computer Science
University of Information Technology
Linh Trung, Thu Duc
21521313@gm.uit.edu.vn

ABSTRACT

State-of-the-art solutions in the areas of "Language Modelling & Generating Text", "Speech Recognition", "Generating Image Descriptions" or "Video Tagging" have been using Recurrent Neural Networks as the foundation for their approaches. Understanding the underlying concepts is therefore of tremendous importance if we want to keep up with recent or upcoming publications in those areas. In this work we give a short overview over some of the most important concepts in the realm of Recurrent Neural Networks which enables readers to easily understand the fundamentals such as but not limited to "Backpropagation through Time" or "Long Short-Term Memory Units" as well as some of the more recent advances like the "Attention Mechanism" or "Pointer Networks". We also give recommendations for further reading regarding more complex topics where it is necessary.

1 Introduction & Notation

Recurrent Neural Networks (RNNs) are a type of neural network architecture which is mainly used to detect patterns in a sequence of data. Such data can be handwriting, genomes, text or numerical time series which are often produced in industry settings (e.g. stock markets or sensors). However, they are also applicable to images if these get respectively decomposed into a series of patches and treated as a sequence. On a higher level, RNNs find applications in Language Modelling & Generating Text, Speech Recognition, Generating Image Descriptions or Video Tagging. What differentiates Recurrent Neural Networks from Feedforward Neural Networks also known as Multi-Layer Perceptrons (MLPs) is how information gets passed through the network. While Feedforward Networks pass information through the network without cycles, the RNN has cycles and transmits information back into itself. This enables them to extend the functionality of Feedforward Networks to also take into account previous inputs $X_{0:t-1}$ and not only the current input X_t. This difference is visualised on a high level in Figure 1. Note, that here the option of having multiple hidden layers is aggregated to one Hidden Layer block H. This block can obviously be extended to multiple hidden layers.

1.1 Model Architecture

We can describe this process of passing information from the previous iteration to the hidden layer with the mathematical notation proposed in. For that, we denote the hidden state and the input at time step t respecively as $\mathbf{H}_t \in \mathbb{R}^{n \times h}$ and $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ where n is number of samples, d is the number of inputs of each sample and h is the number of hidden units. Further, we use a weight matrix $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$, hidden-state-to-hidden-state matrix $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ and a bias parameter $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$. Lastly, all these informations get passed to a activation function ϕ which is usually a logistic sigmoid or tanh function to prepair the gradients for usage in backpropagation. Putting all these notations together yields Equation 1 as the hidden variable and Equation 2 as the output variable.

$$\mathbf{H}_t = \phi_h(\mathbf{X}_t \, \mathbf{W}_{xh} + \mathbf{H}_{t-1} \, \mathbf{W}_{hh} + \mathbf{b}_h) \tag{1}$$

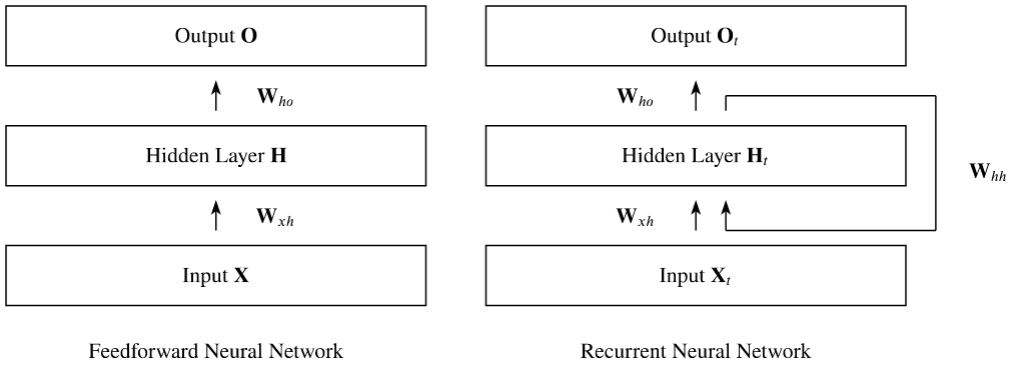

Figure 1: Visualisation of differences between Feedfoward NNs and Recurrent NNs

$$\mathbf{O}_t = \phi_o(\mathbf{H}_t \, \mathbf{W}_{ho} + \mathbf{b}_o) \tag{2}$$

Since $\mathbf{H}_t$ recursively includes $\mathbf{H}_{t-1}$ and this process occurs for every time step the RNN includes traces of all hidden states that preceded $\mathbf{H}_{t-1}$ as well as $\mathbf{H}_{t-1}$ itself.

If we compare that notation for RNNs with similar notation for Feedforward Neural Networks we can clearly see the difference we described earlier. In Equation 3 we can see the computation for the hidden variable while Equation 4 shows the output variable.

$$\mathbf{H} = \phi_o(\mathbf{X} \, \mathbf{W}_{xh} + \mathbf{b}_h) \tag{3}$$

$$\mathbf{O} = \phi_o(\mathbf{H} \, \mathbf{W}_{ho} + \mathbf{b}_o) \tag{4}$$

A RNN uses a simple nonlinear activation function in every unit. However, such simple structure is capable of modelling rich dynamics, if it is well trained through timesteps.

1.2 Activation Function

For linear networks, multiple linear hidden layers act as a single linear hidden layer [10]. Nonlinear functions are more powerful than linear functions as they can draw nonlinear boundaries. The nonlinearity in one or successive hidden layers in a RNN is the reason for learning input-target re- lationships. Some of the most popular activation functions are presented in Figure 2. The “sigmoid”, “tanh”, and rectified linear unit (ReLU) have received more attention than the other activation functions recently. The “sigmoid” is a common choice, which takes a real-value and squashes it to the range [0, 1]. This activation function is normally used in the output layer, where a cross-entropy loss function is used for training a classifica- tion model. The “tanh” and “sigmoid” activation functions are defined as

2 Conclusion

...

Acknowledgments

.....