Introduction

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory.

RNN’s are relatively old, like many other deep learning algorithms. They were initially created in the 1980’s, but can only show their real potential since a few years, because of the increase in available computational power, the massive amounts of data that we have nowadays and the invention of LSTM in the 1990’s.

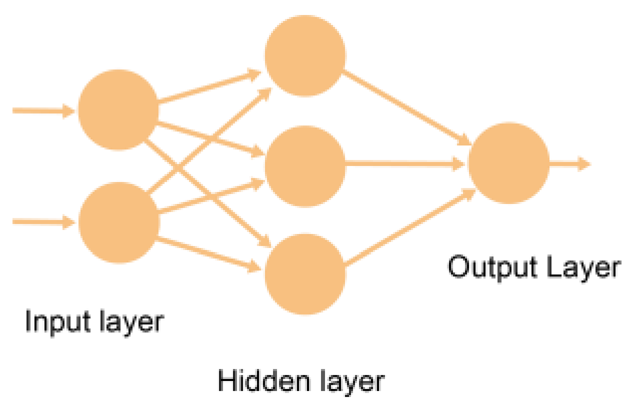
Because of their internal memory, RNN’s are able to remember important things about the input they received, which enables them to be very precise in predicting what’s coming next.

This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.

Why Recurrent neural network?

“Whenever there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame.”

Feed-Forward Neural Network

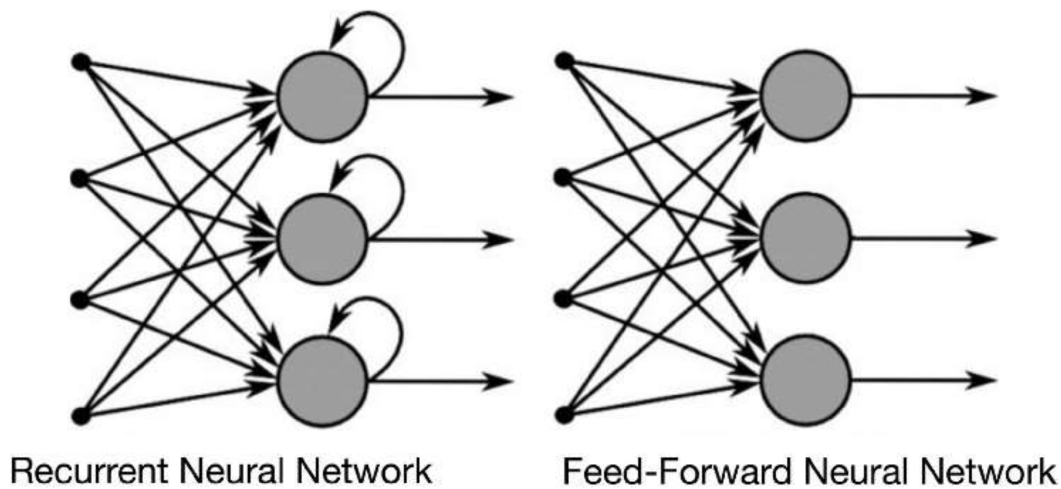


In a Feed-Forward neural network, the information only moves in one direction, from the input layer, through the hidden layers, to the output layer. The information moves straight through the network. Because of that, the information never touches a node twice.

Feed-Forward Neural Networks, have no memory of the input they received previously and are therefore bad in predicting what’s coming next. Because a feedforward network only considers the current input, it has no notion of order in time. They simply can’t remember anything about what happened in the past, except their training.

Recurrent Neural Networks

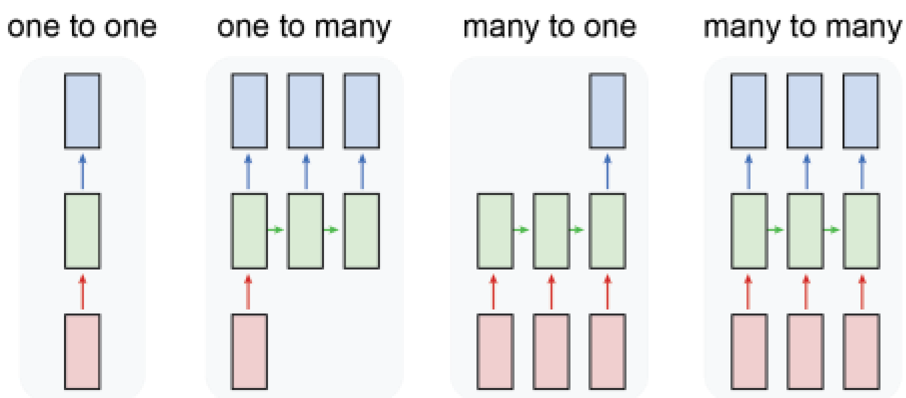
In a RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously.
The two images below illustrate the difference in the information flow between aRNN and a Feed-Forward Neural Network.



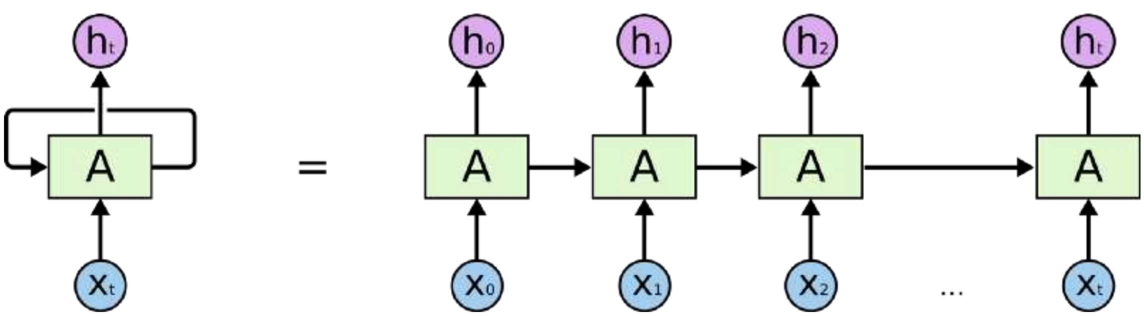
Imagine you have a normal feed-forward neural network and give it the word "neuron" as input and it processes the word character by character, At the time its reaches the character "r", it has already forgetten about "n", "e" and "u". Which makes it almost impossible for this type of neural network to predict what character would come next.

A Recurrent Neural Network is able to remember exactly that, because of it's internal memory. It produces output, copies that output and loops it back into the network.

Also note that while Feed-Forward Neural Networks map one input to one output, RNN's can map one to many, many to many (translation) and many to one (classifying a voice).



The image below illustrates an unrolled RNN. On the left, you can see the RNN, which is unrolled after the equal sign. Note that there is no cycle after the equal sign since the different timesteps are visualized and information gets's passed from one timestep to the next. This illustration also shows why a RNN can be seen as a sequence of Neural Networks.



If you do Backpropagation Through Time, it is required to do the conceptualization of unrolling, since the error of a given timestep depends on the previous timestep.

Within BPTT the error is back-propagated from the last to the first timestep, while unrolling all the timesteps. This allows calculating the error for each timestep, which allows updating the weights. Note that BPTT can be computationally expensive when you have a high number of timesteps.

Long-Short Term Memory

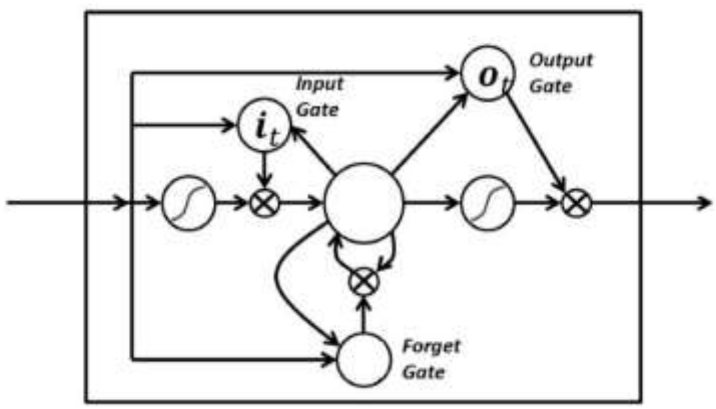
Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. Therefore it is well suited to learn from important experiences that have very long time lags in between.

The units of an LSTM are used as building units for the layers of a RNN, which is then often called an LSTM network.

LSTM's enable RNN's to remember their inputs over a long period of time. This is because LSTM's contain their information in a memory, that is much like the memory of a computer because the LSTM can read, write and delete information from its memory.

This memory can be seen as a gated cell, where gated means that the cell decides whether or not to store or delete information (e.g if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which not.

In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate) or to let it impact the output at the current time step (output gate). You can see an illustration of a RNN with its three gates below:



The gates in a LSTM are analog, in the form of sigmoids, meaning that they range from 0 to 1. The fact that they are analog, enables them to do backpropagation with it.

The problematic issues of vanishing gradients is solved through LSTM because it keeps the gradients steep enough and therefore the training relatively short and the accuracy high.

References:

For more details on RNN can refer below:

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/