Introduction to Sequence Modelling

Sequences are a data structure where each example could be seen as a series of data points. This sentence: “Ram is playing cricket. He is a very good batsman.” is an example that consists of multiple words and words depend on each other. The same applies to medical records. One single medical record consists in many measurements across time.

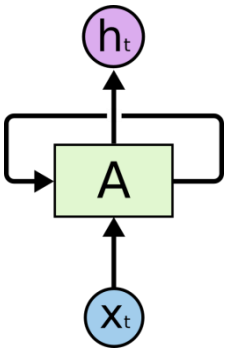
Sequence Modelling is the ability of a computer program to model, interpret, make predictions about or generate any type of sequential data, such as audio, text etc. Humans don’t start their thinking from scratch every second.

As you read this, you understand each word based on your understanding of previous words. You don’t throw everything away and start thinking from scratch again. Your thoughts have persistence. Classic ANNs do not have the mechanism of memory. An ANN neuron’s current state depends only on the current input as it discards information about the previous inputs to the cell. So, an AI algorithm called the Recurrent Neural Network is a specialised form of the classic Artificial Neural Network (Multi-Layer Perceptron) that is used to solve Sequence Modelling problems. Recurrent Neural Networks are like Artificial Neural Networks which has loops in them. This means that the activation of each neuron or cell depends not only on the current input to it but also its previous activation values.

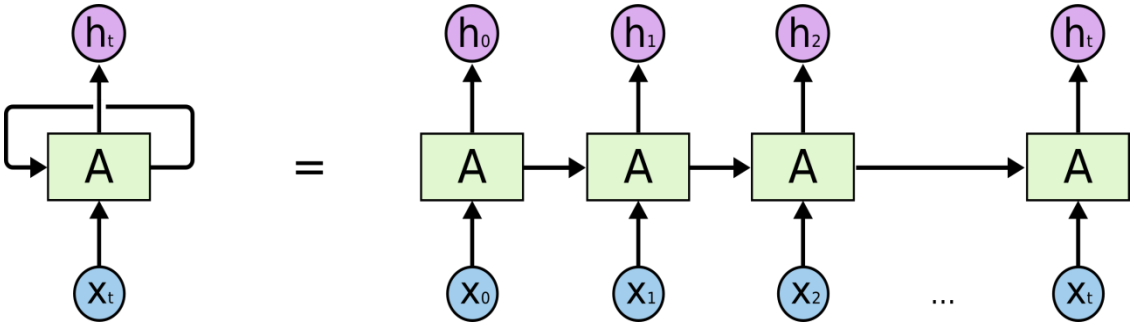
RNN (Recurrent Neural Networks)

Recurrent Neural Network (RNN) is a type of Neural Network where the outputs from previous step are fed as input to the current step. The architecture of an RNN is also inspired by the human brain. As we read any essay, we are able to interpret the sentence we are currently reading better because of the information we gained from previous sentences of the essay. On a basic level, interpreting a certain part of a sequence requires information gained from the previous parts of the sequence. Thus, in a human brain, information that persists in our memory while interpreting sequential data is vital in understanding each part of the sequence. Similarly, RNNs also try to incorporate this capacity of memory by updating something called the “state” of its cells each time we move from one part of a sequence to another. The state of a cell is basically the total information gained by it so far by reading the sequence. So, the current state or knowledge of a cell in an RNN is not only dependent on the current word or

sentence it is reading, but is also dependent on all the other words or sentences it has read before the current one. Thus, the name Recurrent Neural Network.



Recurrent Neural Networks have loops.



An unrolled recurrent neural network.

- The first image above illustrates a recurrent neuron or cell. It is a simple neuron that has a loop. It takes some input x and gives some output h.
- This neuron can be thought of as multiple copies of the same unit or cell chained together. This is illustrated by the second image, which shows an "unrolled" form of the recurrent neuron. Each copy or unit passes a message (some information) to the next copy.

Formula for calculating current state:

$$\mathbf{h_t = f(h_{t-1}, x_t)}$$

Where: h_t is current state,
h_{t-1} is previous state,
x_t is input state

Formula for applying Activation function (tanh):

$$h_t = \tanh (W_{hh} h_{t-1} + W_{xh} x_t)$$

Where:
W_{hh} is weight at recurrent neuron
W_{xh} is Weight at input neuron

Formula for calculating output:

$$y_t = w_{hy} h_t$$

Where:
y_t is output
W_{hy} is weight at output layer

Training through RNN

1. A single time step of the input is provided to the network.
2. Then calculate its current state using set of current input and the previous state.
3. The current h_t becomes h_{t-1} for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

Problems with RNN

Vanishing gradient problem

Capturing long term dependencies

Recurrent Neural Networks face the problem of long term dependencies very often. On many occasions, in sequence modelling problems we need information from long ago to make predictions about the next term/s in a sequence. For example, if we want to find the next word in the sentence “I grew up in Spain and I am very familiar with the traditions and customs of …..”. To predict the next word (which seems to be Spain), we need to have information about the word “Spain”, which is just the 5th word in the sentence. But we need to predict the 17th word in the sentence. This is a large time gap, and RNNs are prone to

losing information given to it many time steps back. RNNs are unable to capture these long term dependencies in practice.

Sequence Modelling Applications

Some applications of sequence modelling are:

- **Language models** (prediction of the next word given a seed string, as you see in keyboard apps on mobile phones)
- **Machine translation** (automatic translation between different languages)
- **Computational biology**, for example in the functional modeling of DNA and protein sequences (predicting which regions of biological sequences are functional and what that function could be)