An Introduction to RNN for Beginners

BEGINNER DEEP LEARNING MATHS

Introduction

Have you ever used “Grammarly” or “text summarizer” and wondered how does Grammarly tells us grammatical mistakes or how “Apple’s Siri” and “Amazon Alexa” understand what you say? There is an interesting algorithm working behind all these applications named “Recurrent Neural Networks” or RNN in short.

The first time I came across RNNs, I was completely mixed up. How can a NN remember things? Why do we even need RNN while we had Feedforward and convolutional Neural networks? After a lot of research and hard work, I somehow managed to understand this algorithm and in this article, I want to convey all my knowledge regarding this algorithm so that you feel comfortable while reading the actual research paper.

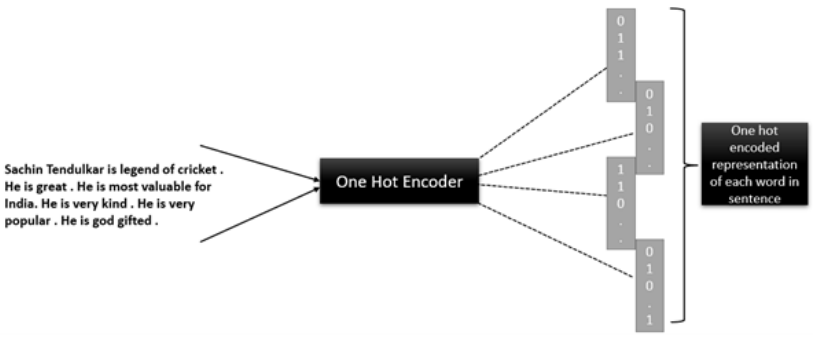
Recurrent Neural Network

Let’s start this algorithm with a question – “love I go to the gym to” did this make any sense to you? Not really. Now read this “I love to go to the gym”, this makes perfect sense. A little jumble in the words made the sentence incomprehensible. If a human brain can’t understand this can a computer encode this? It will have a really tough time dealing with such sentences.

From the above example, we can understand that sequence of a text is very important. Data, where the order or the sequence of data is important, can be called sequential data. Text, Speech, and time-series data are a few examples of sequential data.

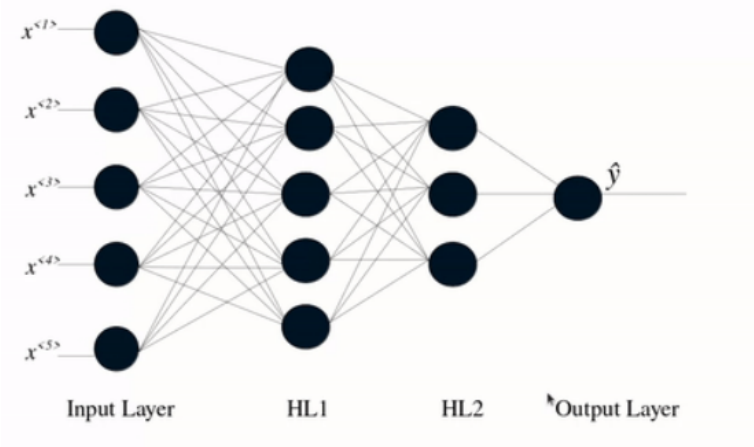
Why do we need RNN while we had Forward Neural Networks?

Suppose we have some reviews of cricket players, and our task is to predict if the review is positive or negative. The first step before making a model is to convert all the reviews (textual data) into machine-understandable form. We can use various techniques like One Hot Encoding or deep learning techniques like Word2vec.



Sentence-1 –> Sachin Tendulkar Legend of cricket

We see that the total number of words or tokens in this sentence is 5.



Similarly sentence-2 “He is great” has 3 words, but the input layer in the architecture is fixed which is 5. Hence there is no direct way to feed this data into the network. Now you must be thinking that why don’t we convert each sentence to be length equal to that of the sentence with maximum length by adding zeros. Yes, this might solve the problem but then think of the number of weights your network will have, it will definitely increase exponentially.

Suppose the maximum length of a sentence is 10 (which is not realistic, it will be much bigger in real-world applications).

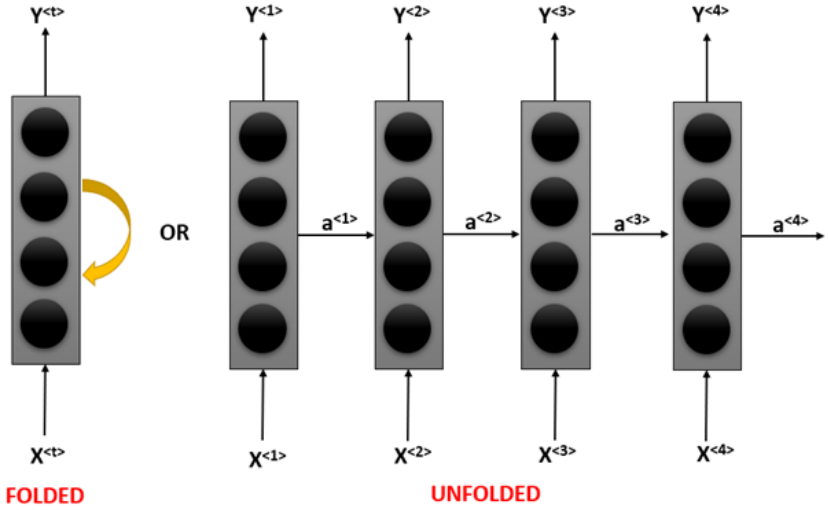
Let the number of words in corpus= 10k (considering a very small corpus)

Then each input will become 100k dimensional and with just 10 neurons in the hidden layer, our number of parameters becomes 1 million! To have a proper network means having billions of parameters. To overcome this we need to have a network with weight sharing capabilities.

This is where RNN comes into the picture. They use the same weights for each element of the sequence, decreasing the number of parameters and allowing the model to generalize to sequences of varying lengths.

In a standard neural network, all the inputs and outputs are independent of each other, however, in certain cases such as predicting the next word of the phrase, the preceding words are essential. Therefore, RNN was created which used a Hidden Layer to overcome this problem.

RNN Model Overview



In the diagram above the “Folded” part represents the neural network, we do not pass the entire sentence to it but rather we pass a word in it and this word will give us one output. It will also produce an activation function which will then be passed on to the next time step as you can see in the diagram. Both these diagrams represent the same thing.

Let’s understand this with the help of an example.

Suppose you are doing a “Named Entity Recognition” project, where you want to predict the name of an entity. The sentence you pass to your model is “Chirag worked at Microsoft in India”. Instead of passing this entire sentence at once, we will only pass the first word in our model.

This RNN will take an input activation which is nothing but a zero matrix initially (it will learn it gradually) and give an output activation function which is then passed on to the next timestep.

In t=2, it will take **worked** as an input and produce an activation as well as an output. Here 0 means it is not an entity. It will then take the third word at t=3, it will continue repeating the process until we reach the end of the statement.

A few things to note here is that this is the same block that is repeated through the time and here the order in which we pass the sentence does matter, for example in the diagram above the prediction “Organization” is made not just considering the word Microsoft but also considering all the words that have occurred into the sentence so far because we are passing the activation functions to next time stamps repeatedly.

Types of RNN and their application

IMAGE SOURCE: https://www.analyticsvidhya.com/blog/2021/06/a-visual-guide-to-recurrent-neural-networks/

Many to one: Suppose you want to make a model which predicts the rating of a movie based on its reviews (Movie rating prediction). There your output will always be one that is the rating of the movie and input may be of any size, size reviews can be of any length. Some more applications are Sentiment analysis, emotion identification, etc.

One to many: One example could be a model whose job is to generate a baby name based on a given input. For example, if we pass “male” as the input so our model should generate a male name or vice versa. Here the input size can be only one and the output size can be anything. One thing to note here is that the output from each timestamp is also passed as an input to the next time stamp and that is how the model knows which character has occurred first and which character to produce.

Many to many: In the case of language translation the input size can be of any length and the output can also be of any length. For example, English to French translation. Here when all the input words are passed only then the output will be generated (fig-4 in the above diagram). In such type of model, the first part which takes the input is known as “Encoder” and the second part which gives the output is called “Decoder”, we will understand more about these two later.

Conclusion

I hope this article provided you with the basic intuition you need. In the next article, we will understand what this model does under the hood, we will understand how does backpropagation works in RNN along with the detailed descriptions of LSTMs and GRUs. To summarize, we learned what is sequential data and how we can make use of textual data to make wonderful applications like speech recognition, image caption generator, etc.

Try playing with the architecture of these RNNs and be amazed by their performance and applications. Do share your findings and approach in the comments section.

Article Url - https://www.analyticsvidhya.com/blog/2022/04/an-introduction-to-rnn-for-beginners/

Anshul Saini

I am an undergraduate student currently in my last year majoring in Statistics (Bachelors of Statistics) and have a strong interest in the field of data science, machine learning, and artificial intelligence. I enjoy diving into data to discover trends and other valuable insights about the data. I am constantly learning and motivated to try new things.