ORIGINAL RESEARCH PAPER

Volume-8 | Issue-6 | June-2019 | PRINT ISSN No. 2277 - 8179

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

SURVEY OF PREDICTION USING RECURRENT NEURAL NETWORK WITH LONG SHORT-TERM MEMORY

Engineering

Ms. Archana Gopnarayan — Computer Engineering, Vidyalankar Institute of Technology, Wadala, Mumbai

Prof. Sachin Deshpande* — Computer Engineering, Vidyalankar Institute of Technology, Wadala, Mumbai
*Corresponding Author

ABSTRACT

Sequence prediction problems are major problem from long time. From predicting sales price, movie plots, speech recognizing, predicting next word on the Phone's keyboard and match results. With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long Short Term Memory is most effective solution .LSTM have an edge over feed-forward neural networks and RNN in many ways. This is because of selectively remembering patterns for long durations of time. LSTM enable RNN to remember their inputs over a long period of time. This is because LSTM 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.

KEYWORDS

LSTM, RNN, prediction

1.1 Deep Neural Network

Deep learning is a technique that builds on deep neural networks (DNNs). A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it is a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculates the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks. Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on your Phone's keyboard.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, LSTMs have been observed as the most effective solution. LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time.

1.2 Neural Network

Artificial neural networks (ANNs) are a computational approach that is based on the way a biological brain solves problems. The neurons are connected by links, which imitate the biological axons, and they interact with each other. Each node takes input data, performs a simple operation, and passes the result to other nodes.

1.2.1Feed-Forward neural Networks

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 feed forward 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.

1.2.2 Recurrent Neural Networks

In 1983, Hopfield introduced feedback connections to the feedforward backpropagation networks. This was the introduction of Recurrent Neural Networks (RNNs). As Recurrent Neural Networks are having at least single feedback connection they have the ability to store correct pattern and they were based on more recent history than the past [5].
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.

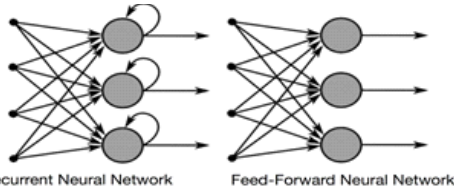
Recurrent Neural Networks are the state of the art algorithm for sequential data. This is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for Machine Learning problems that involve sequential data But it is also important that you understand what sequential data is. It basically is just ordered data, where related things follow each other. Examples are financial data or the DNA sequence. The most popular type of sequential data is perhaps Time series data, which is just a series of data points that are listed in time order.

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. Recurrent Neural Networks produce predictive results in sequential data that other algorithms can't.

The two images below illustrate the difference in the information flow between a RNN and a Feed-Forward Neural Network.



A usual RNN has a short-term memory. Consider normal feed-forward neural network and give it the word “computer” as an input and it processes the word character by character. At the time it reaches the character “p”, it has already forgotten about “c“, “o“ and “m“, 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.

Recurrent Neural Networks add the immediate past to the present. Therefore a Recurrent Neural Network has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about what is coming next, which is why a RNN can do things other algorithms can't.

In sequential data, which can be the stock market's data for a particular stock. A simple machine learning model or an Artificial Neural Network may learn to predict the stock prices based on a number of features: the volume of the stock, the opening value etc. While the price of the stock depends on these features, it is also largely dependent on the stock values in the previous days. In fact for a trader, these values in the previous days (or the trend) is one major deciding factor for predictions.

In the conventional feed-forward neural networks, all test cases are considered to be independent. That is when fitting the model for a particular day, there is no consideration for the stock prices on the previous days.This dependency on time is achieved via Recurrent Neural Networks. A typical RNN looks like:

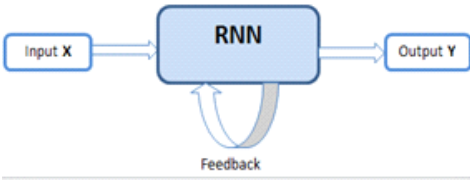


Fig: Recurrent Neural Network

RNNs can solve our purpose of sequence handling to a great extent but not entirely. Now RNNs are great when it comes to short contexts, but in order to be able to build a story and remember it, we need our models to be able to understand and remember the context behind the sequences, just like a human brain. This is not possible with a simple RNN.

1.2.3 Limitations of RNNs

Recurrent Neural Networks work just fine when we are dealing with short-term dependencies. That is when applied to problems like:
The color of sky is ------

RNNs turn out to be quite effective. This is because this problem has nothing to do with the context of the statement. The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue. Thus the prediction would be:
The color of sky is blue

However, RNNs fail to understand the context behind an input. Something that was said long before cannot be recalled when making predictions in the present. Let's understand this as an example:
I spent 10 years working for the kids in Spain. Then I moved to India.
I can speak well in --------

Here, we can understand that since the author has worked in Spain for 10 years, it is very likely that he may possess a good command over Spanish. But, to make a proper prediction, the RNN needs to remember this context. The relevant information may be separated from the point where it is needed, by a huge load of irrelevant data. This is where a Recurrent Neural Network fails!

The reason behind this is the problem of **Vanishing Gradient.** In order to understand this, consider feed-forward neural network learns. In conventional feed-forward neural network, the weight updating that is applied on a particular layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a particular layer is somewhere a product of all previous layers' errors. When dealing with activation functions like the sigmoid function, the small values of its derivatives (occurring in the error function) gets multiplied multiple times as we move towards the starting layers [7].

As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

A similar case is observed in Recurrent Neural Networks remembers things for just small durations of time, i.e. if we need the information after a small time it may be reproducible, but once a lot of words are fed in, this information gets lost somewhere. This issue can be resolved by applying a version of RNNs – the Long Short-Term Memory Networks.

3.1 LSTM (Long Short-Term Memory) Networks

When we arrange our meetings for the day, we prioritize our appointments. If in case we need to make some space for anything important we know which meeting could be canceled to accommodate a possible meeting.

Turns out that an RNN doesn't do so. In order to add new information, it transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for *'important'* information and *'not so important'* information [3].

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

Let's take the example of predicting stock prices for a particular stock. The stock price of today will depend upon:

1. The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.
2. The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.
3. The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

These dependencies can be generalized to any problem as:

1. The previous cell state (i.e. the information that was present in the memory after the previous time step)
2. The previous hidden state (i.e. this is the same as the output of the previous cell)
3. The input at the current time step (i.e. the new information that is being fed in at that moment)

We may have some addition, modification or removal of information as it flows through the different layers. Just because of this property of LSTMs, where they do not manipulate the entire information but rather modify them slightly, they are able to forget and remember things selectively.

Fischer and Krauss [2] deployed Long Short-Term Memory (LSTM) networks, which are used to carry out the prediction of out-of-sample directional movements for S&P 500 stocks from 1992 to 2015.

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory [2]. 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.

3.2 Architecture of LSTM

The functioning of LSTM can be visualized by understanding the

functioning of a news channel's team covering a criminal story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take any of the three steps. Let's say, we were assuming that the crime was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? We immediately **forget** the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You **input** this information into your news feed.

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and **output** the relevant things to your audience. Maybe in the form of "*XYZ turns out to be the prime suspect.*"

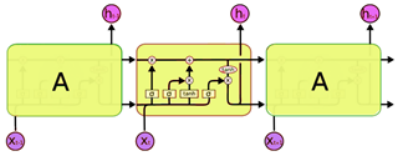


Fig 3.1.1: Architecture of LSTM

A typical LSTM network is comprised of different memory blocks called **cells** (the rectangles)**.** There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called **gates.** Each of them is being discussed below.

3.2.1 Forget Gate

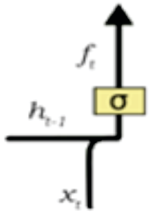


Fig 3.1.2: Forget gate

A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.

This gate takes in two inputs; h_t-1 and x_t.

h_t-1 is the hidden state from the previous cell or the output of the previous cell.

x_t is the input at that particular time step.

The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state. Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely. Similarly, a '1' means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

3.2.2 Input Gate

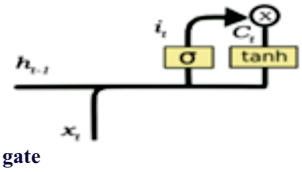


Fig 3.2.2: Input gate

The input gate is responsible for the addition of information to the cell state. This addition of information is basically three-step process as seen from the diagram above.

1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h_t-1 and x_t.
2. Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the **tanh** function, which outputs values from -1 to +1.
3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important* and is not *redundant.*

3.2.3 Output Gate

This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.

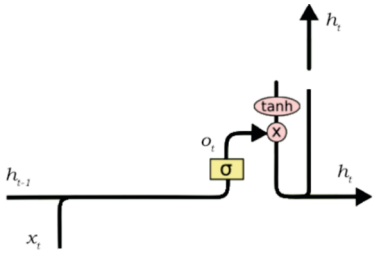


Fig 3.2.3: Output gate

The functioning of an output gate can again be broken down to three steps:

1. Creating a vector after applying **tanh** function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

CONCLUSION

Deep learning is the application of artificial neural networks which requires high computational resources. There are wide ranges of neural networks proposed by researchers with specific modifications of existing models. It is difficult to know what type of neural network is best suitable for prediction as it is governed by the problem we aim to solve. LSTN prediction model give better result than normal RNN because of internal memory. For sequence prediction problem LSTN gives more accuracy.

REFERENCES

[1] WILLIAM GRANT HATCHER AND WEI YU, "A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends" date May 24, 2018.
[2] T. Fischer and C. Krauss, ``Deep learning with long short-term memory networks for financial market predictions," Eur. J. Oper.Res., 2017.
[3] https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
[4] https://ieeexplore.ieee.org/document/8300345 "A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market"
[5] K. D. Gunawardana, An Introduction to Artificial Neural Networks for Accounting and Finance Modelling, Colombo: Author Publication, 2009.
[6] https://towardsdatascience.com/deep-learning-in-finance-9e088cb17c03
[7] DANIEL PETTERSSON ROBERT NYQUIST "Football Match Prediction using Deep Learning"
[8] Manuel R. Vargas, Beatriz S. L. P. de Lima and Alexandre G. Evsukoff "Deep learning for stock market prediction from financial news articles"
[9] Qu Xiaoyun, Kang Xiaoning, Zhang Chao, Jiang Shuai, Ma Xiuda Shaanxi Key Laboratory of Smart Grid, Xi'an Jiaotong University "Short-Term Prediction of Wind Power Based on Deep Long Short-Term Memory"
[10] "Deep learning for finance: deep portfolios" J. B. Heaton N. G. Polson J. H. Witte