Recurrent Neural Networks For Weather Prediction

Bo Yang
Department of Computer Science
University of Texas ar Dallas
Richardson, United States
bxy180002@utdallas.edu

Chunhe Ni
Department of Computer Science
University of Texas ar Dallas
Richardson, United States
cxn170001@utdallas.edu

Hao Si
Department of Computer Science
University of Texas ar Dallas
Richardson, United States
hxs180013@utdallas.edu

***Abstract*—Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs [1]. In this report, we explore LSTM RNN architectures for weather prediction. Weather prediction provides convenience for people. People can know the weather changes in advance so as to plan things in time. Weather prediction can make climate-related industries do well in weather prevention measures in advanced. For example, it can reduce the impact of natural disasters in agriculture and reduce the occurrence of safety accidents in the transportation industry. The ultimate goal is to do our best to reduce the impact of weather on people. Therefore, we need a useful and efficient tool. LSTM is a reliable tool that we can achieve our goal.**

Keywords—RNN, LSTM, Keras, Elephas, Spark

I. INTRODUCTION AND BACKGROUND WORK

There are two forecasting methods that we know, one is the linear algorithms(AR, MA, ARIMA) and the non-linear algorithms (ARCH, GARCH, Neural Networks). We can use any of them to do our project but finally we find that they are not fit the dataset. Statistic method is according to the previous time series and sometimes is easier. But the prediction of this method is limited because this method is based on random time series or time duration. Actually, the feature of time series is unstable. As a branch of Artificial Intelligence, machine learning can learn from dataset so as to build linear or nonlinear map between input and output. Over the years the LSTM has been used in many fields like Language Modeling [2], Large Vocabulary Speech Recognition [3], 3D Human Action Recognition [4]. In this project, we use LSTM to predict weather.

Nowadays, weather prediction becomes more and more available on the internet. Most of us like to know the weather from some applications. The data on the second day is always accurate. In order to make data like this, we need to feed the model and let them learn some useful things. In order to achieve that, this model need to remember things it previously learned. But traditional neural network cannot do this. Therefore, we will use RNN to keep learning. We can use MSE to evaluate the performance of the model. Deep neural network structures can understand and predict hidden movement. In addition, RNN can predict information for a given time series, to identify the increase and decrease of the trend. The weather changes may not always according to the same cycle so RNN can make the best model.

In this project, we are going to implement our own LSTM unit and base on our custom unit to build an RNN which is able to forecast weather. We are using Keras python library to help us achieve this goal. Keras is a high-level, widely used neural networks library which is written in Python and is able to run on top of TensorFlow. It has two concepts: one is the layer, and another is the cell. The Keras library has implemented lots of layers and cells. Users can use these layers and cells to build their own neural networks. What's more, Keras offers support to users which allow users to build custom layers and cells by extending base layers and implements specific methods. With the help of this feature, we can implement our custom layer and custom LSTM cell easily using Keras.

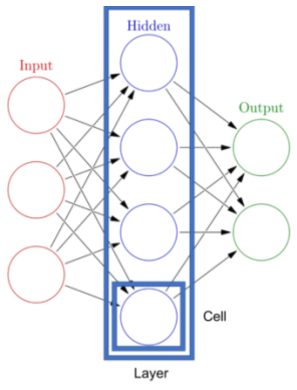


Figure 1 Layer and Cell in neural network

To training the Keras model with Spark, we are using Elephas. Elephas is a powerful library which is an extension of Keras. It allows user to training distributed deep learning models with Spark. It implements a data-parallel algorithm class on top of Keras. It accepts Spark RDDs or data frames as input. Keras model will first be initialized and serialized. Then it will be sent to workers with data and model parameters. Workers will deserialize the model, train the model with data that sent to it, and send updates back. Base on received updates, the "master" model will be updated synchronously or asynchronously.

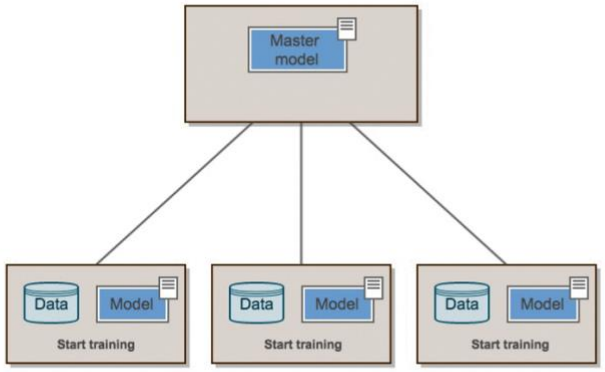


Figure 2 Elephas Structure

We are using Dark Sky API to collect weather data. Dark Sky API is an API service which offers detailed weather statistic data anywhere on the global. We use python to fetch data through the Dark Sky API and pick the weather data we need in the weather prediction.

II. Theoretical and Conceptual of RNN and LSTM

A. *Reccurent Neural Networks (RNN)*

In a traditional neural network, the layers are fully connected from the input layer to the hidden layer then to the output layer, but the nodes between each layer are disconnected. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. The specific form of expression is that the network memorizes the previous information and applies it to the calculation of the current output, that is RNNs are networks with loops in them, allowing information to persist.

The following picture shows Recurrent Neural Networks A. The input is X_t and output is h_t. The upper loop allows the information generated by each step then to the next step. There is a vector in position A, which represents the value of the hidden layer (a counter is hidden here, you can also imagine that this layer is actually multiple bytes, the number of counters is the same as the size of the vector) .There is weight matrix from the input layer to the hidden layer. And another weight matrix is from the hidden layer to the output layer. However, if you think more, RNNs are not so different from ordinary neural networks. An RNN can be viewed as multiple copies of the same network, each of which passes information to the next copy.

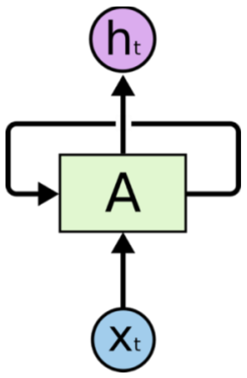


Figure 3 RNN has loops

B. *Long Short-Term Memory (LSTM)*

Among the RNNs, the most widely used and most successful model is the LSTM. It is a special type of RNN, capable of learning long-term dependencies and is a type of deep learning model that is mostly used for analysis of sequential data. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. Forget gate is to decide what information to get rid of from the cell state. Input gate is to decide what new information to store in the cell state. Output gate is to decide what part of the cell state to output. [5]

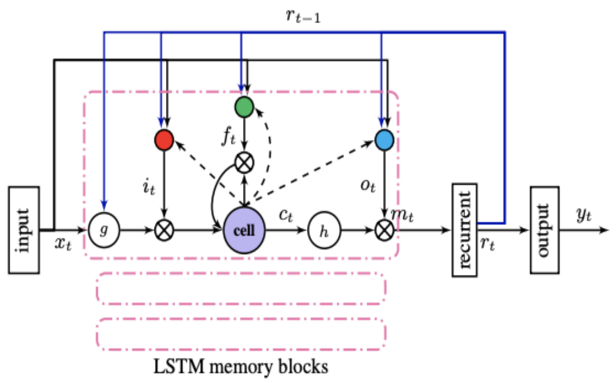


Figure 4 LSTMP RNN architecture. A single memory block is shown for clarity. [1]

Neural network models suitable for multiple input variables have always been a headache for developers, but recurrent neural networks based on (LSTM) can almost perfectly solve the problem of multiple input variables. The LSTM model is a very powerful time series model. They can predict any number of steps in the future. The LSTM module (or unit) has 5 basic components that can model long-term and short-term data.

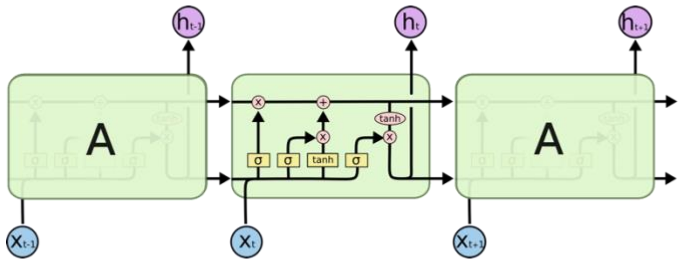


Figure 5 The repeating module in an LSTM contains four interacting layers

There are 4 interacting layers in the LSTM showing in the Figure 4. The first layer is called "forget gate layer". It determines how much information we are going to keep in the previous cell status. The layer input is h_{t-1} and x_t. The output is a number between 0 and 1 which indicates how much information will be kept from previous cell status. The formula is as follow:

$$f_t = \sigma(W_f\, x_t + U_f h_{t-1} + b_f)$$

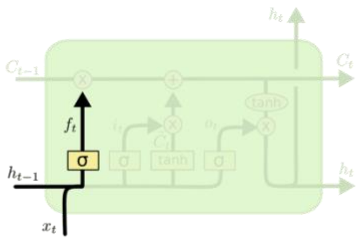


Figure 6 First layer in LSTM

The Second step is to determine how much new information that need to be added to the cell status. This contains two parts. One is a sigmoid gate and another is a tanh gate. The formula is as follow:

$$i_t = \sigma(W_i\, x_t + U_i h_{t-1} + b_i)$$

$$\widehat{C}_t = \tanh(W_C\, x_t + U_C h_{t-1} + b_C)$$

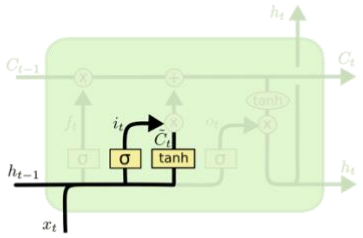


Figure 7 Second layer in LSTM

The third layer is to update the cell status. The information from previous cell status and the new information will be combined as the new cell status. The formula is as follow:

$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t$$

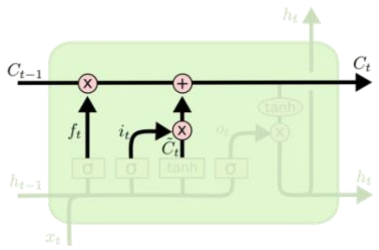


Figure 8 Third layer in LSTM

The final layer is the output layer. This layer determines what information we are going to output. We multiple output with the result of cell status go through tanh gate which means we combine the output with the cell status. The formula is as follow:

$$o_t = \sigma(W_o\, x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

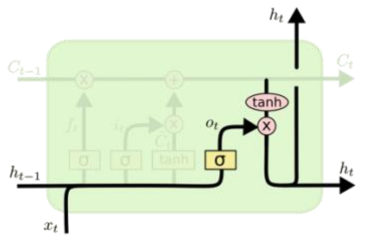
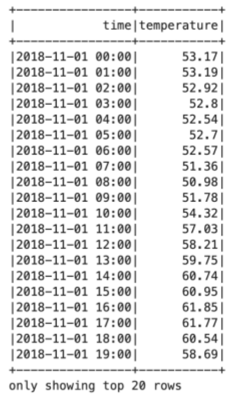


Figure 9 Forth layer in LSTM

III. Dataset Description

All weather data were collected from Dark Sky API. The API request called 'Time Machine Request' returns the observed weather information going back decades. It also gave you choice to look up Hour-by-hour which meet the project requirement. The training model need the dataset only contains Temperature value for every hour of half year in Dallas city. Attached the latitude and longitude of Dallas with request, The API returns the JSON format data. It was converted from JSON to CSV file and excluded useless data blocks. The original CSV dataset includes the time of the hour, weather summary, temperature, apparent temperature, humidity, UV index, wind speed and etc. Because the project only focusses on weather prediction, the temperature data was selected using Spark data frame from raw dataset. The first column shows timestamp and the second column includes the Temperature at that timestamp. The total number of temperature records is 3437. The following figure is the overview of Temperature dataset and data table.



```
+----------------+-----------+
|            time|temperature|
+----------------+-----------+
|2018-11-01 00:00|      53.17|
|2018-11-01 01:00|      53.19|
|2018-11-01 02:00|      52.92|
|2018-11-01 03:00|       52.8|
|2018-11-01 04:00|      52.54|
|2018-11-01 05:00|       52.7|
|2018-11-01 06:00|      52.57|
|2018-11-01 07:00|      51.36|
|2018-11-01 08:00|      50.98|
|2018-11-01 09:00|      51.78|
|2018-11-01 10:00|      54.32|
|2018-11-01 11:00|      57.03|
|2018-11-01 12:00|      58.21|
|2018-11-01 13:00|      59.75|
|2018-11-01 14:00|      60.74|
|2018-11-01 15:00|      60.95|
|2018-11-01 16:00|      61.85|
|2018-11-01 17:00|      61.77|
|2018-11-01 18:00|      60.54|
|2018-11-01 19:00|      58.69|
+----------------+-----------+
only showing top 20 rows
```

Figure 10 Weather data set after processing

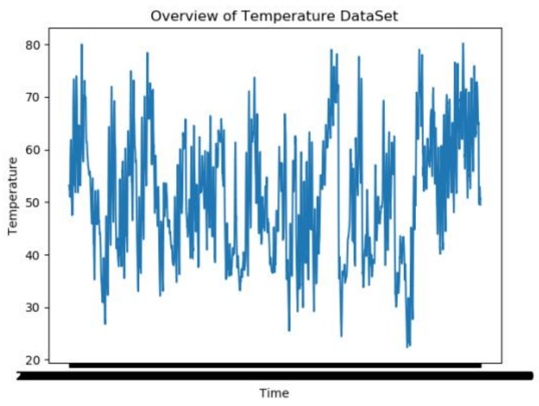


Figure 11 Overview of temperature data set

IV. Results and Analysis

We split weather data set into two parts. One part is the training part which is used to training RNN. The other part is the testing part which we used to test the performance of our RNN. For this project, the first 67% of weather data are assigned to training data set and the rest of weather data are assigned to testing data set.

Since we are training an RNN, the input data should include previous weather data. We need to preprocess weather data before we fit the data into RNN. For each time point, we combine its weather data with previous weather data in a fix length of time which is given by the user setting as the input data. The actual weather data at that time point is the output data. After data preprocessing is finished, we build the RNN, fit the training data and test the RNN performance with the testing data. We use root mean square error (RMSE) to measure the performance during training and testing.

We built an RNN model with one layer and in this LSTM layer there are 5 LSTM cells. The cell is also called as unit. Our LSTM Layer extends RNN Layer in Keras library. It uses our custom LSTM cell which extends base layer in Keras Library. We implement our custom LSTM cell based on the LSTM structure and formulas mentioned in Part II. Elephas is used for training Keras model with spark. The parameter prev_data_length defines the length of previous data which is used in data preprocessing. We set the prev_data_length to 4 which means each training data entry contains previous 4 hours weather data. The data set size is 3437. During the data set, the first 67% data is the training set and the last 33% data is the testing set. After 100 iterations of training, we use the trained model to predict testing set. Based on the training result and testing result, we draw the graph as follow:

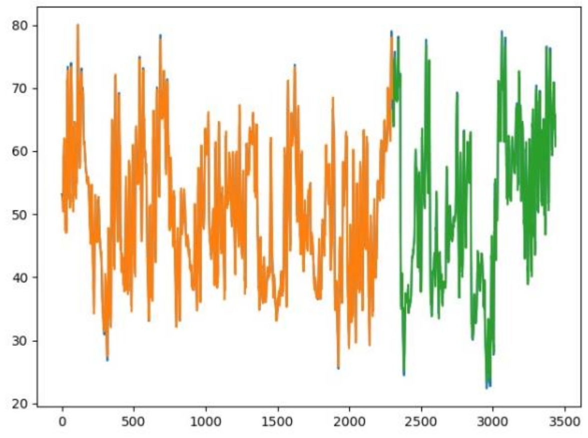


Figure 12 Model training result and testing prediction

In this graph, the blue line is the original data set, the orange line is the training result, the green line is the prediction based on the testing set. From the graph we find that the orange line and green line almost cover the entire blue line, which means the model makes prediction on weather perfectly. The training RMSE is 1.1389. the testing RMSE is 1.2617. These metrics shows that the RNN model with LSTM has a very high precision on predicting time series data.

V. Conclusion and Future Work

Based on the experiments and result showing before, we can find that RNN has a very great performance on predicting time series data. Especially predicting data which has internal relationship with previous data. This is because the structure of RNN and the LSTM unit inside the network. In this project, we implemented our own LSTM unit and built the RNN using our LSTM units. We trained the RNN and tested precision of prediction with the weather data that we collected from the internet. There are still lots of parts can be improved in this project. For example, we only make the prediction on the temperature. Various metrics can be appended to prediction list such as precipitation, humidity and so on. We can also add more options to our own LSTM units. User should be able to choose different activation functions and bias options. In conclusion, with the help of RNN and LSTM, a powerful, accurate and effective weather prediction system can be built and bring us a lot of benefits and convenience in our daily life.

REFERENCES

[1] Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling : https://www.isca-speech.org/archive/archive_papers/interspeech_2014/i14_0338.pdf.

[2] Sundermeyer, Martin, Ralf Schlüter, Hermann Ney. "LSTM neural networks for language modeling." Thirteenth annual conference of the international speech communication association. 2012.

[3] Haşim Sak, Andrew Senior, Françoise Beaufays "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition" arXiv:1402.1128

[4] Liu, Jun, et al., "Spatio-temporal lstm with trust gates for 3d human action recognition." European Conference on Computer Vision. Springer, Cham, 2016.

[5] Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.