Weather Features Prediction with Deep learning

Yiheng Ye
UCSD ECE department
4067 Miairmar Street
yiy291@ucsd.edu

Abstract

Weather Report has always been a critic issue in our current life. With the growth of modern industry, comes the demand of high accuracy and small granularity prediction on multiple aspect of weather. In this article, I will explore using simple architecture of LSTM models to find some useful methods to predict weather features like temperature and humidity in given areas. I will also discuss why sometimes LSTM models are not so reliable in making predictions.

1 Introduction

As the human society developing, our demand for weather report no longer stays at the level of "knowing whether today will be raining". We need, if possible, complete data of all features of weather. Humidity, temperature, wind speed, precipitation, and many other features, we want to know the exact data for each of them in the local or even in the adjacent areas. So many human activities are relied on those data from small things like making an outdoor event to big issues like maintaining basic infrastructure working. Therefore, it is obvious that many scholars have put their effort into making prediction for those data using deep learning. Currently, for weather data, our models are usually focus on a single feature like wind speed. For example, the common method for dealing with short term wind speed is using CEEMDAN to reconstruct the time-series data and making prediction on the specific features using neural network like LSTM[1] or even simple ANN [2]. The benefit of doing so is that you might get a better prediction on a single feature for a specific areas, however, as weather is a complex chaos system, the models we get this way is not robust enough if we want to apply them to different places. Therefore, In this article, I will try to challenge the method of putting different features at the same time into the LSTM models and make prediction for specific data in the next time range. In this way, I believe we can mimic the weather better using our models and the computation problem (constantly updating data for the next prediction) can be solved with the improvement of hardware and cloud computing.

2 Method

2.1 Basic Description and Discussion of Experiment Method

Generally, for dealing with weather data, we tend to use complete ensemble empirical model decomposition with adaptive noise (CEEMDAN) method to reconstruct the time-series data for one specific feature (rainfalll, etc.), and feed the reconstructed data to our deep learning model like LSTM model. As we have mentioned in the introduction, there are numerous way of integrating CEEMDAN with deep learning or machine learning method, such as LSTM[1], ANN[2], and DNN[3]. However, in this project, we want to incorporate multiple features in the same time so we do not need to make

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

decomposition on every feature we have. We will use a relatively brutal method which is considering the weather data in a specific time range and predict one of our selected feature in the next time range. For example, if we have a bunch of data including rainfall, wind speed, wind direction, temperature, humidity for Beijing area in past 4 years, what we want to do is to collect every day data and predict feature values for the next day one by one. Following this logic, we can construct our training and test data sets for every feature we have in our general weather data set and we can use mean squared error to evaluate our model performances on different features.

The main reason why I choose using multiple feature instead of sticking to single one is coming from the theory of how ensemble empirical model decomposition (EEMD) works. This kind of decomposition methods are capable of dealing with non-linear and non-stationary signals. They decompose the signal data into a set of components complete and nearly orthogonal basis for the original signal while keeping the same time range and time scale. Therefore, it is a great tool to analyze the time series data. The problem is that those kind of methods are considering the noise in our signals as white noise, and try to average out those noises in the computation process. It is an efficient way to dealing with the complex weather signal, but it is not an ideally proper assumption in geological theory. In a giving time range, the weather feature (for example, rain fall), is associated with other features in the same area and has a specific performance in those time range. For example, in a certain area, rain fall might be generally lower and fluctuation for the rain fall might not be huge while we have a series of heavy rains in the summer. If one conduct CEEMDAN on the whole year of rainfall data, it will easily break this important trait as the model is averaging out throughout those seasonal variations. Therefore, we have to capture the differences in a more specific time ranges. As we have discussed before, the better way to make the prediction here is try to mimic the weather changing process by observing all related features in one day, or in a past week/month, and try to generalize the chaotic system we have here to make a prediction for the next day.

However, choosing this method will also have some drawbacks. One of the problem of this method is that it can only be predicting single feature one by one, which makes the prediction time-consuming. Also, this means that we have to train multiple models on the same feature sets again and again which is also a redundant implementation of deep learning model. Although I believe this method will perform better prediction result over traditional methods, we should be noticed that this comes with a larger cost as well.

After we decide our general logic of experiment method, we want to find out the proper model to train on our time series data, and long short-term memory model (LSTM) is intuitively the viable option if we want to work on our time series data and we will work on this kind of model as well as its variances in the later of our project. Now I want to introduce the general concept of LSTM model and its advantage.

2.2 LSTM model

Long short-term memory (LSTM) model is a type of recurrent neural network (RNN) that is good at dealing with sequence prediction by remembering the order dependence and thus it makes the model an ideal choice for predicting time series data. A recurrent neural network, compared with other deep learning model, can "memorize" past iterations information for further training, and thus making them suitable of training sequential data. LSTM model, furthermore, is an improved version of RNN whose structure enable its ability to learn to bridge large number of time steps. That is to say, LSTM model overcomes the vanishing/explode gradients problem by selectively remember or forget information so that this model is resilient even towards a large time lag. A vanishing/explode gradients problem is a problem with back-propagation when we are actually training RNN models. When calculating the long-term gradients (gradients with large time lag), due to the algorithm of back-propagation, the gradients terms to be zero or infinite, which will makes training over large time steps impossible. For example, a traditional RNN might be performing well in our project if we just doing a prediction using past 3 days of weather and try to predict the weather next day, but if we want to use past month's data, the model will be performing much worse on this task. However, if we use LSTM model, those kind of problems are no longer disturbing due to how LSTM model designs.

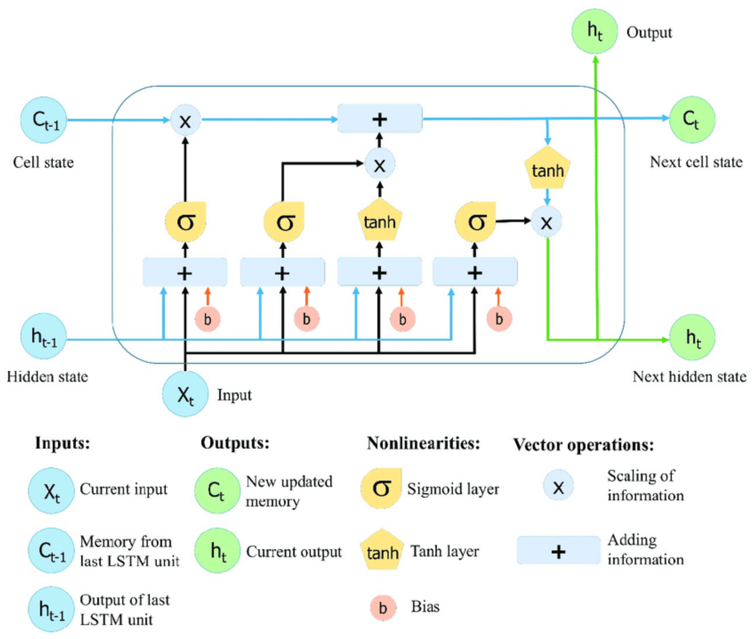


Figure 1: LSTM model structure made by Le et al[4].

To explain how LSTM works, we can look at the figure 1 we have above. One can find that there are 2 states involved in each step of LSTM model, which are cell state and hidden state. And also, new input is added in each step of LSTM model. We can clearly find that cell state is about information we memorize while the hidden state is just the output we have after each step of LSTM model. We look at the line from left to right one by one. The leftmost one that takes in input and hidden state to Sigmoid layer is the forget gate, the output of this gate is used to evaluate how much we want to "forget" the information we store in the cell state by multiplying the result matrix to cell state. Next two lines combined is input gate, where similarly, we memorize the information in this input to our cell state. At last on the right we have the output gate, which generate our final result for this step. Intuitively, we use current input to decide which information should be kept/discarded and this helps the model over come the problems we have in RNN models by giving an extra weight to the information we remember throughout each step.

2.3 other possible models/model combinations

To perform our task, there are also other possible implementation and I am going to discuss why they are viable though I am not going to use them here.

BiLSTM is also a pick for this kind of sequential prediction. Basically it just adds another layer to the model by reading the input in a backward direction. It will probably reduce the MSE we have here which makes our model looks better when making validation. However, in the real life the weather features can and can only be going in one direction and applying this model to our data set can probably resulting in a overfitting problem so that our model is not optimal in the actual scenario. CNN-LSTM is also a possible choice by applying convolution neural network to our features before using LSTM model. The CNN model here is used for featuer extraction, which is unnecessary here since we want to use as much as possible weather features we have and discarding some of them will only be resulting in a loss in MSE. Therefore, although we have some fancy options for our prediction task here, I still believe applying LSTM model is enough to solving the problem we got. Furthermore, as we have discussed before, our method is already consuming more computation cost than usual methods and we'd better keep our model simple.

3 Experiment

3.1 Data set and set up

In this experiment, we will use 4166 weather records for the city ChangSha, HuNan, China, from 2018 to 2020. The data set contains those weather data including temperature, rainfall, pressure, wind speed, wind direction, and humidity. The Data is recorded every 3 hour from November 18th, 2018 to July 16th, 2020, except every 11pm of that day. Actually, the 11pm's record is recorded in the 2am's record next day, which cause some duplicated record time. Therefore, we can sort the data set by the time and keep the weather features only since they are strictly recorded by every 3 hours. For example, the temperature curve for our data set is shown below:

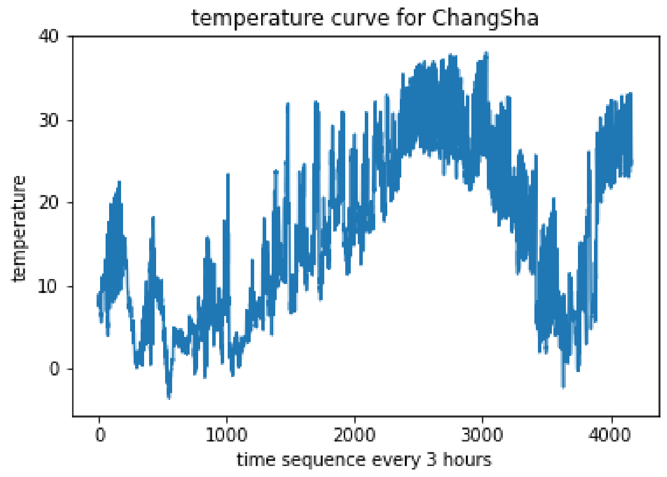


Figure 2: Temperature Curve for Changsha, HuNan from 2018 Nov. to 2020 Jul..

We can identify the two winter and one (and a half) summer from the curve. We will train on all features and make prediction on one of them. Next we want to determine the train-test split of our data set, which is 0.7:0.3. We will make a normalization of our data set since some of them are over 1000 while the other is around 0 40. According to the definition of train-test split, we will use the mean and standard deviation of our training set only to normalize our data set. We will evaluate our model from the training loss and the test loss of our data set. The loss function we are using is mean squared error, which can be stated below:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{1}$$

Where the y_i is the true value and $\hat{y_i}$ is the predicted value. And for the hyperparameter, we are using one-layer LSTM. We have 10 epochs to train and is training on batch size of 32. The learning rate is 0.001.

3.2 Result and analysis

Let's look at the temperature model first:

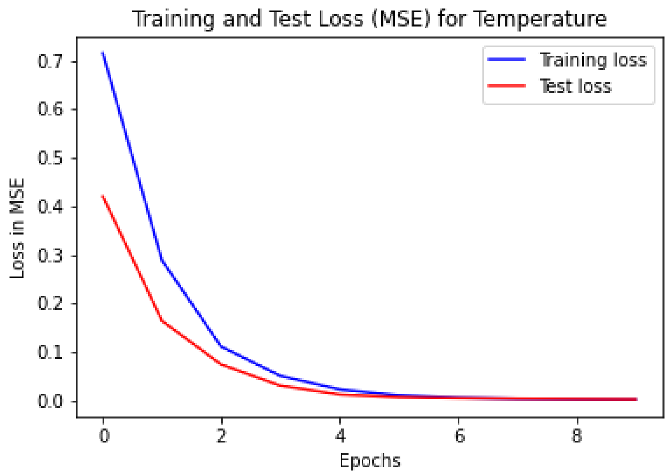


Figure 3: Train and test loss (MSE) of temperature prediction

We can find from the graph that our prediction converges successfully after 5 epochs and the model is performing very well on both training and test data set. Our prediction plot for orignal data set after normalization is:

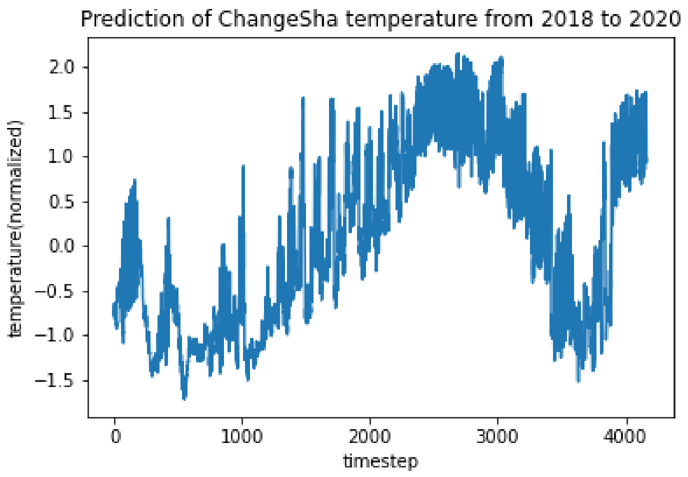


Figure 4: Prediction of normalized temperature

Clearly this is a very ideal output. Then, we can look at another feature to be predicted, which is pressure:

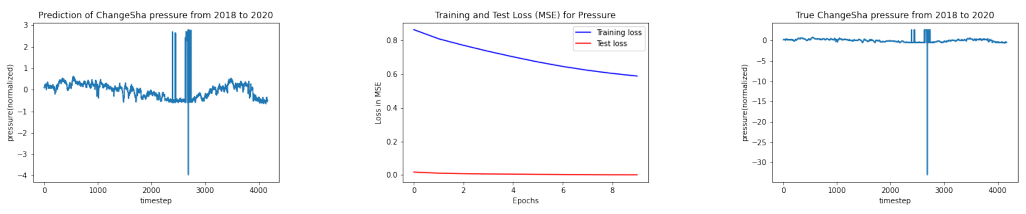
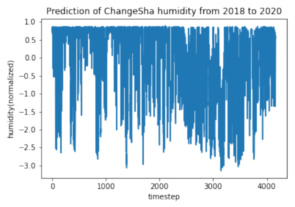
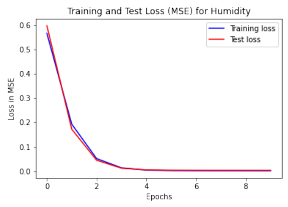


Figure 5,6,7: predicted pressure (left), train and test loss for pressure (middle), and true pressure (right)

Our model is not performing well on the pressure prediction test, especially for the training data set. We can find that there are some irregular outliers in the pressure dataset, which makes the model works not well. However, I believe this is more like a problem with the original dataset instead of our model. Our model is still doing a good job and it evens out the influences of some outliers.

At last, let us check the prediction for humidity:





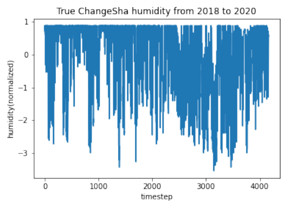


Figure 8,9,10: predicted humidity (left), train and test loss for humidity(middle), and true humidity(right)

It looks like even the humidity is making random variation, our model still successfully captured those changes and reaches very low training loss as well as test loss before 4 epochs. This illustrate that as far as there are no extreme outliers in our data set, the LSTM model can make pretty great prediction results with simple structure even on seemingly random data. Notable, all my training process actually went very fast (around 7ms for each epoch) and I believe the computation burden is not a very big issue for our solution.

3.3 Conclusion

In this paper, I explore the possibility of applying series of weather feature data to predict the weather of future time steps using simple LSTM model. The result we conduct turns out to be very good. As far as there are not some individual outliers, the LSTM model will always gives a good fit even the training data set seems to be random. The overall computation time and resources we use is also surprisingly acceptable and I think my solution a viable solution for making better weather prediction.

References

[1] Chen, Gonggui; Li, Lijun; Zhang, Zhizhong;Li, Shuaiyong. (2020). Short-Term Wind Speed Forecasting With Principle-Subordinate Predictor Based on Conv-LSTM and Improved BPNN. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2982839.

[2] Y. Ren, P. N. Suganthan and N. Srikanth, "A Comparative Study of Empirical Mode Decomposition-Based Short-Term Wind Speed Forecasting Methods," in IEEE Transactions on Sustainable Energy, vol. 6, no. 1, pp. 236-244, Jan. 2015, doi: 10.1109/TSTE.2014.2365580.

[3] Q. Zhang, Z. Tang, S. Cao and G. Wang, "Wind farm wind power prediction method based on CEEMDAN and DE optimized DNN neural network," 2019 Chinese Automation Congress (CAC), 2019, pp. 1626-1630, doi: 10.1109/CAC48633.2019.8996744.

[4] Le, Xuan Hien ; Ho, Hung ; Lee, Giha ; Jung, Sungho. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. Water. 11. 1387. 10.3390/w11071387.