LSTM model to forecast time series for EC2 cloud price

Alkharif Sarah
Computer Science
Kookmin University
Seoul, South Korea
sara.alabdulaziz@gmail.com

Kyungyong Lee
Computer Science
Kookmin University
Seoul, South Korea
leeky@kookmin.ac.kr

Hyeokman Kim
Computer Science
Kookmin University
Seoul, South Korea
hmkim@kookmin.ac.kr

***Abstract*—With the widespread use of spot instances in Amazon EC2, which utilize unused capacity with unfixed price, predicting price is important for users. In this paper, we try to forecast spot instance price by using long short-term memory (LSTM) algorithm to predict time series of the prices. We apply cross validation technique to our data set, and extract some features; this help our model to learn deeply. We make the prediction for 10 regions, and measure the performance using root-mean-square error (RMSE). We apply our data to other statistical models to compare their performance; we found that our model works more efficiently, also the error is decreasing more with cross validation and the result is much faster. Based on our result we try to find the availability zone that less become over on-demand price and less changing price over time, which help users to choose the most stable availability zone.**

***Index Terms*—LSTM, AWS, spot instance, time series**

I. Introduction

Cloud Computing becomes more popular with simple and inexpensive way to access servers, run applications and store any size of data, at anytime and anywhere with a fixed price. Amazon EC2, which is the most famous example of cloud computing, where they provide many instance types such as General, Compute Optimized, GPU, Memory Optimized, and Storage Optimized. With many types of instances, there are different operating systems and more than 10 regions around the world, ever region is independent but may one region has isolated locations which know as availability zones.

In August, 2006, AWS opened up a new service called spot instances sometime known as spot market, which used unused capacity from on-demand with unfixed price. The user can set the maximum amount of hourly pay and if the spot market price is less than or equal to the user price, it can be run. On the other hand, if at any time the cost is increasing, the user then has just one minute to save his work before instances shut down. Often, the spot instances price is lower than the on-demand price. However, at some periods, spot instance can grow higher than the on-demand price, which is uncomfortable for users. In this project, we forecast the spot instance price using long-short-term memory (LSTM) to predict the time series of the prices. We apply cross validation technique, and use root-mean-square error RMSE to evaluate the performance.

II. Related Work

Time series is a collection of data which is in a temporal order where the data is being collected through the time. As an example of time series: spot instance price, temperature and the stock market, all the observations are changed over time. Many researchers try to forecast time series using variety of deep learning algorithms and statistical models. With the statistical models, we have already examined to forecast time series using seasonal ARIMA as the best algorithm [5]. In this work, we try to avoid the delay time during prediction. We can solve the delay time problem by using Synthetic Gradients technique [8], which saves the time when we train a massive number of networks. Also, in time series prediction when we make the model simple as moving average and autoregressive, it works more efficiently [9]. In addition, our dataset shows high correlation between observation, but has not a volume in size enough to cover a whole seasonal pattern such as a year [10]. Thus, we think neural network may work much better than the classical time series models.

We build application work in real time with short period data [11] like ours, where we import daily data for three months to apply LSTM. From another research [12], we notice that k-fold cross validation technique works more practically, but the experiments should be independent. Based on the result from other studies [13] [10] [14] [15], we try to fix our forecast by using the best solution that we can get. Furthermore, based on our analysis, we compared the result with some other statistical algorithms.

III. Model

Deep learning models Artificial neural networks (ANN) by Rosenblatt [7], though the basic concept of the ANN was introduced by McCulloch and Pitts [6], were working well with almost all type of datasets to simulate human brain where the model learns to make prediction or classifier the data. LSTM model one of neural networks family for sequence prediction problems.

The benefit of LSTM is cell unit which tries to remember the most crucial point in data. LSTM developed to fix the exploding and vanishing gradient when training RNN; by using a memory unit known as a cell unit or cell memory c_t for a network. Cell unit is allowed to remember the output,

which helps the model to learn from all output of the sequence data.

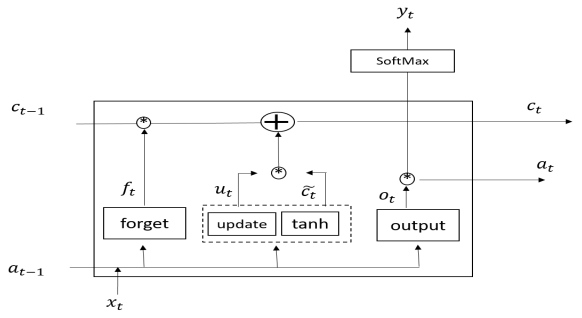

Fig. 1. The diagram of a LSTM building block.

The diagram in Figure 1 shows the most essential three gates of LSTM model: update, forget and output gates. LSTM can be described by Equations 1∼6. Equation 1 computes a candidate $\hat{c}_t$. The candidate $\hat{c}_t$ has own parameters which are the weight w_c and the bias b_c. The update gate u_t, which is computed in Equation 2, has parameters which are the weight w_u and the bias b_u. They help the model to decide when it updates the memory cell by using a sigmoid function; if an output of the sigmoid function is close to one, it is updated, and it is close to zero, it is ignored. The result from candidate $\hat{c}_t$ and update gate u_t will be multiplied together to update the parameters to the memory cell.

$$\hat{c}_t = \tanh\left(W_c[a_{t-1}, x_t] + b_c\right) \tag{1}$$

$$u_t = \sigma(W_u[a_{t-1}, x_t] + b_u) \tag{2}$$

$$f_t = \sigma(W_f[a_{t-1}, x_t] + b_f) \tag{3}$$

$$o_t = \sigma(W_o[a_{t-1}, x_t] + b_o) \tag{4}$$

$$c_t = u_t * \hat{c}_t + f_t * c_{t-1} \tag{5}$$

$$a_t = o_t * \tanh(c_t) \tag{6}$$

Operations of the forget gate f_t is similar to those of the update gate. The forget gate allows the model to choose when to forget the information in memory cell, as shown in Equation 3. The forget gate f_t has parameters which are the weight w_f and the bias b_f. The output gate o_t is computed in Equation 4 by multiplying current input with weight w_o and add bias b_o. Using Equation 5 to update the parameters to cell state, a new cell state c_t is computed and transferred to the next layer by multiplying the update gate with the candidate $\hat{c}_t$ and adding it to the forget gate that is multiplied with the previous cell state c_{t-1}. Finally, the model takes the result a_t to send it to the next layer to keep tracking the parameters using Equation 6.

Cross validation [19] is a widely used technique for evaluating training data in machine learning models. The benefit of the cross validation can be explained as follows: the data is used to train the model often fits the model well. However,

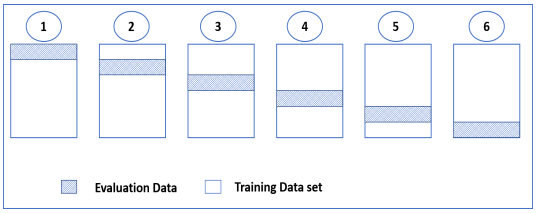

Fig. 2. Cross validation.

when the model is presented with a new set of training data that we did not see before or the data comes from a different distribution, the model may end up with problem of bias/variance; bias occurs when the function underfit the data, whereas variance is when the function overfit the data. As an example, when we get 10% error in a train set and 11% in a validation set, we end up with high variance problem. Also, if we get 15% error in a train set and 16% in a validation set, we will get high bias problem. Usually, we should be somewhere in between. To avoid this problem, we employed k-fold cross validation [20], where the training data is split into k subsets to train them individually. As shown in the Figure 2, in our work, we set k=6 and run the model 6 times. Every time our model runs, the model takes the blue section in Figure 2 as a training dataset, as shown in algorithm 1 line from 4 to 6.

Algorithm 1 LSTM to predict AWS spot price

Input: x
Require: Select features
1: **if** $x >= on - demand$ **then**
2: x =1
3: **end if**
4: $R = subset(x)$
5: $R = array[R_1, R_2, ..., R_i]$
6: **for** `(i in R)` **do**
7: $\hat{y} = LSTM(i)$
8: $rmse = (\hat{y}, y)$
9: **end for**
10: $error.result = avg(rmse)$

IV. EVALUATION

In this work, we fetch 3 months (from Dec. 2017 to Feb. 2018) historical data of spot instance prices from Amazon Web Server. The data contains price, timestamp, instance type, and instance region; we collect this as a small database, for 10 regions. We use another dataset with much longer duration of 8 months (from Mar. to Sep. 2016). In both datasets, we get an hourly price that can make an hourly forecast; and set our target $\hat{y}$ as next price, and select few features. We decided to utilize 7 features as follows: mean, max, min, number of hours, number of days, current price (x_t) minus next price $(x_{(t+1)})$, and normalize value. The seven features improve model skill by learning how the price changes over time, we choose the numbers of features randomly, and not high numbers because

we try to avoid expensive model.

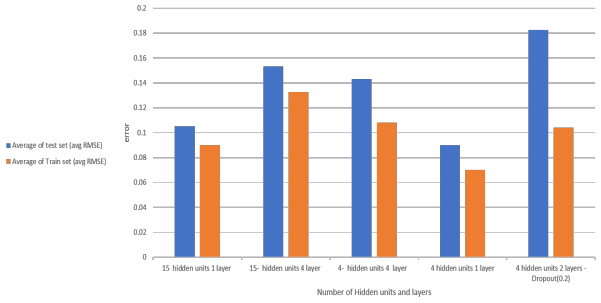

Fig. 3. Number of hidden units and layers.

We used 80% and 20% of the dataset for training and testing, respectively. We set the price that is equal or over an on-demand price to one. We split the training set into k subsets. In this work, we set $k = 6$ and run LSTM model individually to get output $\hat{y}$. We then compute a root-mean-square error (RMSE), with an actual price y to get an average error for every availability zone, using Equation 7, where y is an actual price and $\hat{y}$ is a prediction value.

$$RMSE = \sqrt{(y - \hat{y})^2} \tag{7}$$

We implemented the work using python 3.6.2 and Anaconda open source distribution 4.4.7. To run the model, we used Keras, which is a high-level neural networks API, running on top of TensorFlow. We set our model parameters with four hidden units and one layer which shows the smallest RMSE in Figure 3, where we run our model to find the best number of hidden units and layers. Figure 4 shows our model behavior when we use a small value of iterations or higher, so we start from 1 to 1000 iterations where the best value of iteration for our model for the most of availability zones is between 9 and 14. For this reason, we set number of instance to 10. Figures

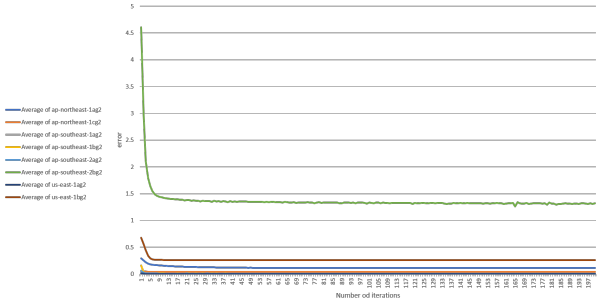

Fig. 4. Number of iterations.

5 and 6, we compare our model result with other time series algorithms using the 3- and 8-month datasets, respectively. Autoregressive, ARIMA and Moving Average performs better than the conventional LSTM. However, LSTM with k-fold cross validation shows the smallest RMSEs in both datasets. We obtain the best results with 0.07 error for training set and 0.09 for test set error using the 3-month dataset, and 0.09 error for training set and 0.12 for test set error using 8-month dataset. After our model is fitted to a training set, it would be

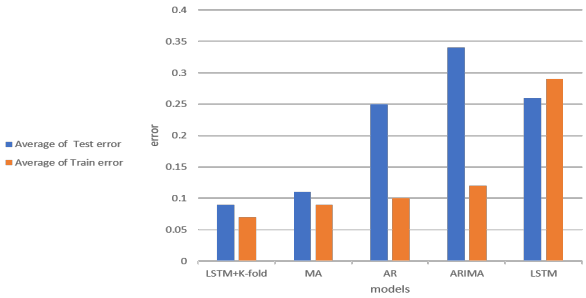

Fig. 5. Average result for 3 months data set.

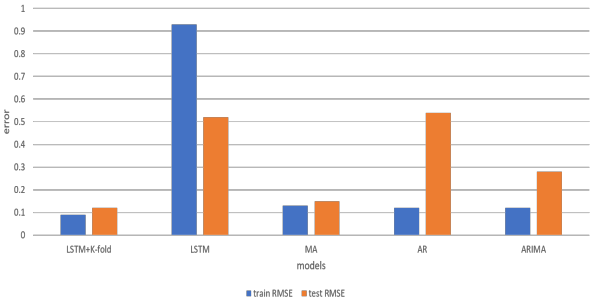

Fig. 6. Average result for 8 months data set.

easy to forecast the next 24 hours, by using many approaches to generate a test set. As an example, the training set can be used to make the prediction for the next 24 hours, by setting the last 24 hours price to forecast the next 24 hours, or it can be generated by using random values from the training set. In our experiment, we used the latter approach to forecast the next 24 hours, by fitting our model to random set and get the forecast.

V. ACPP APPLICATION

With our LSTM model, we try to find the most stable spot instances in real time, whose prices are under the preset on-demand price. This application is called as Amazon Cloud Price Prediction (ACPP). The ACPP application works in real time by importing Amazon spot instance prices daily. With the data, the ACPP application runs our model to forecast the next 24 hours, the first step prepare our data starting from splitting data based on availability zones; then we normalize the data based on on-demand price, then run our model. After we get the predictions, we run a python code to gives users the availability zones whose prices do not become over a preset on-demand price and less changing over time, which helps the users to choose the most stable availability zones. All results for 20 availability zones can be accessed via http://167.99.77.9. Figure 7 shows one of our result graphs, the graph has two y-axes that represent actual prices and its converted values.

The actual spot prices (green line) and the preset on-demand price (blue line) belong to the right axis which represents the prices in US dollar. Similarly, the normalized value (purple line) and predicted value (red line) belong to the left axis which represents the values between 0 and 1.

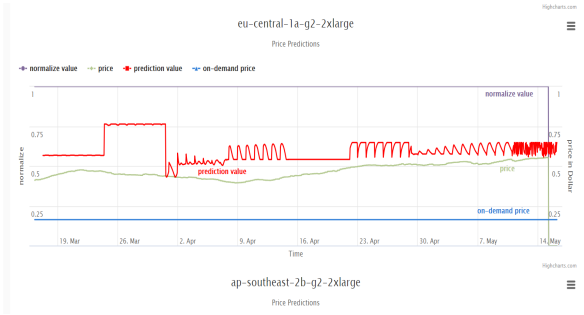

Fig. 7. Chart that show prediction with actual price.

Figure 8 shows the flowchart of ACPP application starting from importing data from AWS server, cleaning the data, applying k-fold cross validation, then running the LSTM model. After we get the results, we find the most stable availability zones.

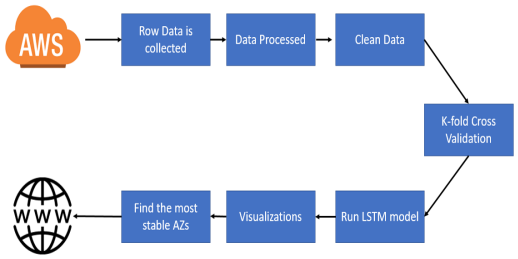

Fig. 8. ACPP chart.

VI. CONCLUSION

We applied LSTM model with k-fold cross validation to 3 months of data from AWS for 10 regions (20 availability zones) as a small data. We also used 8 months data separately. We then use the RMSE to evaluate the performance. we find the error is decreasing more with k-fold cross validation and the result is obtained much faster. We applied our data to other statistical mode to compare their performances. We found that our model works more accurately. Also, we implement our real-time application ACPP using our model to give users the prediction value for the next 24 hours. Based on our result, we try to find the availability zone whose price less becomes over on-demand price and less changes over time.

VII. ACKNOWLEDGMENT

This work is supported by the ICT R&D program of IITP (2017-0-00396) and the Scholarship Program of the Saudi Government.

REFERENCES

[1] *G. U. Yule, On The Time-Correlation Problem With Special Reference To The Variate-Difference Correlation Method, Journal Of The Royal Statistical Society, Vol.84, pp.497-537, July 1921.*

[2] *T. W. Anderson, C. Hsiao, Estimation of dynamic models with error components. Journal of the American Statistical Association, Vol.76, pp.589-606, Feb 1981.*

[3] *G. E. P. Box, G. M. Jenkins, G. C. Reinsel, Time series analysis: forecasting and control, 3rd. Ed, Englewood Cliffs, NJ: Prentice-Hall, 1994.*

[4] *J. Faraway, C. Chatfield, Time series forecasting with neural networks: a comparative. applied statistics, The Journal of the Royal Statistical Society, applied statistics series C, Vol 47, pp.231250, Feb1998.*

[5] *S. A. Alkharif, K. Lee, H. Kim, Time-series analysis for price prediction of opportunistic cloud computing resources. In proceedings of the 7th International Conference on Emerging Databases, Lecture Notes in Electrical Engineering, Springer, Vol.461, pp.221-229, Busan, Korea, Aug 2017.*

[6] *S. Warren, McCulloch, W. A. Pitts, logical calculus of the ideas immanent in nervous activity. In proceedings of the bulletin of mathematical biophysics, Vol.5, pp.115-133, Dec 1943.*

[7] *Rosenblatt, Cornell aeronautical laboratory, Inc. No. 85-460-1,Jan 1957.*

[8] *M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, K. Kavukcuoglu, Decoupled neural interfaces using synthetic gradients, arXiv:1608.05343v2, Jul 2017.*

[9] *S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: results of a forecasting competition. journal of forecasting, Vol.1, pp.111-153, Apr/Jun 1982.*

[10] *N. Laptev, J. Yosinski, E. Li.L, S. Smyl, Time-series extreme event forecasting with neural networks at uber. In proceedings of the IEEE International conference on data mining workshops, pp.18-21, New Orleans, LA, USA, Jun 2017.*

[11] *S. Ravuri, A. Stolcke, Recurrent neural network and lstm models for lexical utterance classification. In proceedings of the ISCA - International speech communication association, proc, interspeech, pp.135-139, Sep 2015.*

[12] *R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection. In proceedings of the 14th international joint conference on Artificial intelligence, Vol 2, pp.1137-1143, San Francisco, Aug 1995.*

[13] *K. Bandara, C. Bergmeir, S. Smyl, Forecasting across time series databases using long short-term memory networks on groups of similar series, arXiv:1710.03222v1, 2017.*

[14] *A. Thakur, S. Kumar, A. Tiwari, Hybrid model of gas price prediction using moving average and neural network. In proceedings of the 1st International Conference on Next Generation Computing Technologies, pp.735-737, Dehradun, India, Sep 2015.*

[15] *E. Coutinho, F. Weninger, B. Schuller, K. R. Scherer, The munich LSTM-RNN approach to the mediaeval, "Emotion in Music" Task. In: UNSPECIFIED, Oct 2014.*

[16] *D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Neurocomputing: foundations of research, pp.696-699, Cambridge, MA, USA, 1986.*

[17] *K. Cho, B. V. Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv:1409.1259v2, Association for Computational Linguistics, pp.103-111, Doha, Qatar, Oct 2014.*

[18] *J. Chung, C. Gulcehre, K. H. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv:1412.3555v1, presented at the deep learning workshop at NIPS2014, 2014.*

[19] *M. Stone, Cross-validatory choice and assessment of statistical predictions, Journal of the Royal Statistical Society B, Vol 63, pp.111-147, 1974.*

[20] *S. Larson, The shrinkage of the coefficient of multiple correlation, Journal of Educational Psychology Vol 22, pp. 4555,1931.*