Predicting Amazon Spot Prices with LSTM Networks

Conference Paper · June 2018

DOI: 10.1145/3217880.3217881

CITATIONS
6

READS
805

5 authors, including:

Matthew Baughman
University of Chicago
4 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Christian Haas
University of Nebraska at Omaha
24 PUBLICATIONS **167** CITATIONS
SEE PROFILE

Ian Foster
University of Chicago
973 PUBLICATIONS **79,454** CITATIONS
SEE PROFILE

Kyle Chard
University of Chicago
115 PUBLICATIONS **1,329** CITATIONS
SEE PROFILE

Predicting Amazon Spot Prices with LSTM Networks

Matt Baughman
Minerva Schools at KGI
San Francisco, California

Christian Haas
College of Information Science and Technology, University of Nebraska at Omaha
Omaha, Nebraska

Rich Wolski
Computer Science Department, University of California, Santa Barbara
Santa Barbara, California

Ian Foster
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois

Kyle Chard
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois

ABSTRACT

Amazon spot instances provide preemptable computing capacity at a cost that is often significantly lower than comparable on-demand or reserved instances. Spot instances are charged at the current spot price: a fluctuating market price based on supply and demand for spot instance capacity. However, spot instances are inherently volatile, the spot price changes over time, and instances can be revoked by Amazon with as little as two minutes' warning. Given the potential discount—up to 90% in some cases—there has been significant interest in the scientific cloud computing community to leverage spot instances for workloads that are either fault-tolerant or not time-sensitive. However, cost-effective use of spot instances requires accurate prediction of spot prices in the future. We explore here the use of long/short-term memory (LSTM) recurrent neural networks for spot price prediction. We describe our model and compare it against a baseline ARIMA model using historical spot pricing data. Our results show that our LSTM approach can reduce training error by as much as 95%.

CCS CONCEPTS

•Computer systems organization → Neural networks; *Cloud computing;* **•Software and its engineering → Cloud computing;**

KEYWORDS

Spot price prediction, Amazon spot instances, LSTM networks

ACM Reference format:
Matt Baughman, Christian Haas, Rich Wolski, Ian Foster, and Kyle Chard. 2018. Predicting Amazon Spot Prices with LSTM Networks. In *Proceedings of 9th Workshop on Scientific Cloud Computing, Tempe, AZ, USA, June 11, 2018 (ScienceCloud'18),* 7 pages.
DOI: 10.1145/3217880.3217881

ScienceCloud'18, Tempe, AZ, USA

978-1-4503-5863-7/18/06…$15.00
DOI: 10.1145/3217880.3217881

1 INTRODUCTION

In order to fill unused cloud compute capacity, Amazon Web Services and other cloud providers offer a tier of preemptable instance types. Amazon's preemptable tier, called spot instances, are allocated according to a dynamic market model in which instances are sold at a fluctuating "spot price." That is, running spot instances are charged at the current spot price throughout their lifetime. When provisioning spot instances, users may optionally specify the maximum price that they are willing to pay; however, this price serves only to limit the cost incurred. Unlike on-demand instances, spot instances may be terminated at Amazon's discretion or if the spot price exceeds the user's maximum price.

Due to the preemptable nature of spot instances, the spot price is often significantly lower than the on-demand or reserved price for the same instance type. However, this discount comes at the cost of reliability—an instance can be terminated with only two minutes notice—and thus spot instances are only suitable for applications that are fault-tolerant or are not time-sensitive. These unique characteristics have made spot instances particularly attractive in scientific computing scenarios as a way of obtaining elastic resources at low cost [4]. We ourselves have used spot instances for executing genomics [18], food security [20], brain imaging [3], and social science [2] analyses. In each case, the automated provisioners used to dynamically obtain and release spot instances [5] rely on methods to predict spot prices to select the cheapest instance types and schedule workload execution.

Researchers have explored a variety of methods to accurately predict spot prices, including parametric models [14, 15, 33], Markov models [28, 34], and cost minimization approaches [9, 35]. Furthermore, there are many prediction mechanisms used to predict prices in other markets (e.g., stock market, electricity market) such as autoregressive integrated moving average (ARIMA) [7], generalized autoregressive conditional heteroskedasticity (GARCH) [12], neural networks [16, 32], and many others. While many of these methods are able to accurately model spot prices in some scenarios they are not universally accurate, perhaps due to the fact that spot prices are not driven solely by demand but rather by Amazon's introduction of hidden externalities that affect pricing [1].

In prior work we have shown that a time series-based approach is well-suited to predicting spot prices for the purpose of providing durability guarantees [33]. In this paper, we again treat the spot

price as a time series and use a hybrid long/short term memory (LSTM)-dense neural network architecture to predict spot prices in the future. Specifically, we describe our three-layer network composed of two LSTM layers and one dense layer to consolidate output from the LSTM layers. To minimize the computational overhead of training our model we explore the potential use of two preprocessing steps that regularize prices and account for seasonality. We evaluate our network architecture using historical spot pricing data and compare the accuracy of our predictions against an ARIMA model—which has been previously shown to perform well in spot prediction [1, 33]. Our results show that our LSTM architecture can accurately predict historical spot prices with average performance up to 95% better than our baseline ARIMA model.

2 RELATED WORK

A wide range of predictive models have been explored to predict spot prices. For example, researchers have used parametric models based on exponentials fitted using Expectation-maximization (EM) [14, 15]; Constrained Markov Decision Processes to minimize the expected cost of an instance, taking into account various costs, including checkpointing and restart delays [28]; and organized the problem as a cost minimization problem based on a Markovian state model that estimates transition probabilities from spot price histories [9]. Closely related prediction models have been developed to support application migration and checkpointing to avoid service downtime [11, 22, 27]. In these cases, schedulers capable of migrating services or applications use prediction models to predict when instances will be terminated.

Little prior work has investigated the prediction of spot prices using artificial intelligence. In fact, apart from our work, we have found few other publications that investigate the application of neural networks to spot price prediction [30, 32]. In the time since these papers were published, neural networks have been applied to playing retro video games [19], beating the masters of Go [24], identifying cancerous tumors [10], listening to and generating raw audio [31], and even beating Go without any previous knowledge of the game [26] and then transferring what was learned to other games such as chess and Japanese chess [25]. Recently, there has been significant interest in using neural networks for effective prediction of stock market prices [23, 29]. Moving away from neural networks and towards spot price prediction, we see different time series analysis methods being applied to stock market prediction [21, 23] and specifically to spot prices [6].

The neural networks used in prior work would be considered small by today's standards, using only a simple multilevel perceptron type neural network with five input nodes, 10 hidden nodes, and one output node [30]. In comparison, our network uses 64 more advanced units and we are exploring models with over 250 input nodes and nearly 3,000 total nodes. Furthermore, prior efforts have focused on using a recurrent neural network, where outputs from each node in a directed line feed to the next, whereas we explore a hybrid architecture that combines elements of the traditional recurrent neural network [13] with traditional fully connected dense layers (i.e., multilevel perceptrons). Advances in computational hardware and software have enabled us to explore such complex and extensive architectures, while seeing similar training times for our most complex models (~13 hrs. on 2x Nvidia Tesla P100s) as the much simpler models from several years ago.

3 BACKGROUND

Amazon EC2 offers computing capacity as predefined "instance types." The instance type defines the resources available to a particular instance in terms of CPU, memory, and storage. Instance types are organized into various families such as general purpose, compute optimized, memory optimized, storage optimized, and accelerated computing. A particular instance type is named according to its family and its relative resource capacity. For example, `c3.2xlarge` is a fifth generation, compute optimized instance type with 8 vCPUs, 15 GB of RAM, and user-configurable block storage.

Amazon organizes instances into independent *regions*, each with a number of independent *availability zones*. Regions represent geographically separated data centers and availability zones represent physically separated resources within those data centers. When provisioning resources, users must select a region, availability zone, and instance type. While all users see the same region names, availability zones are arbitrarily named for each user to avoid users overloading a single availability zone.

Amazon creates a spot market for each instance type, availability zone, and region. Prior to November 2017, the spot market required that users place a bid—the maximum value they were willing to pay for a particular instance type—to acquire spot instances. Periodically Amazon ordered current bids and recalculated the spot price based on available capacity. Bids above the spot price were allocated instances, while those below the spot price were not. Any running instances with bids below the current spot price could be terminated. Thus, the spot price was directly related to revocation and therefore users aimed to compute a bid price that was above the spot price (in order to avoid revocation) but not significantly so (to reduce financial risk). In November 2017, the spot market was updated such that users are no longer required to specify a bid. Instead, spot prices are set by Amazon and adjust gradually based on long-term trends in supply and demand. Users simply pay the spot price for an instance at the given time and instances may be revoked entirely at Amazon's discretion (rather than based on the user's maximum price). However, from a user's perspective, the existence of an optional maximum price is similar to the previous bidding model.

Amazon provides up to three months of price histories for each instance type, availability zone, and region. This information provides a basis from which one can try to forecast prices and derive patterns. An example of the price history for the `c3.2xlarge` instance type in the `us-east-1`, `us-west-1`, and `us-west-2` regions is shown in Figure 1. This figure highlights the significant savings (compared to on-demand instances) possible in the spot market. It also shows the variability of the spot price within and between availability zones.

In order to improve prediction accuracy and our ability to apply machine learning models, we have collected spot pricing data from multiple sources spanning 3 years, with data for all instance types and availability zones in the US regions for the past year. This data is stored as "tick" data: it only includes a point in the series when the price changes from the previous price.

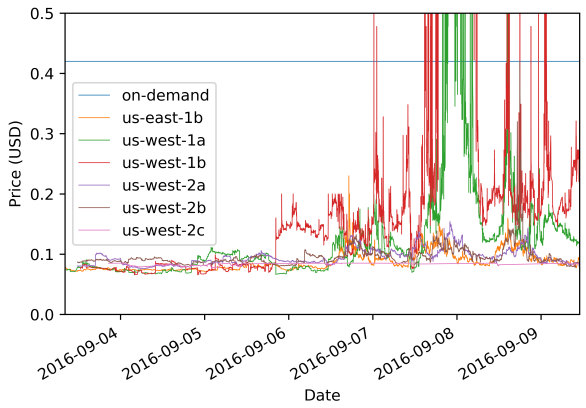

Figure 1: Spot price history for `c3.2xlarge` instances from September 3, 2016 to September 10, 2016 in three `US` regions. Each series represents a different availability zone.

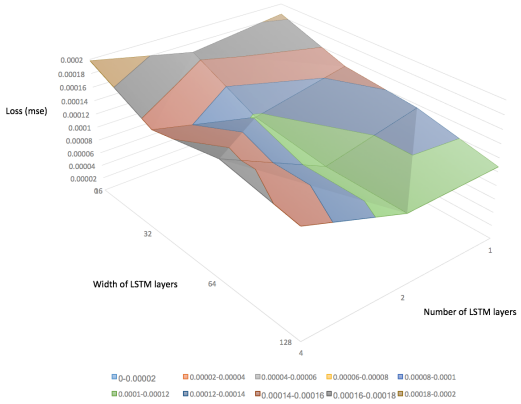

Figure 2: Training MSE (USD squared) after ten epochs of training each combination of LSTM width and depth.

4 SPOT PRICE PREDICTION

In this paper we aim to predict the spot price for a particular instance type, availability zone, and region for a specific time in the future. Specifically, we aim to predict the price using prior histories from all regions, availability zones, and instance types. Given the relative complexity of this task and the potential for many latent features to influence the spot price (e.g., seasonality, relationships between instance families, resource supply, etc.) we investigate if recurrent neural networks (RNNs) can provide better accuracy than standard statistical approaches. RNNs are becoming increasingly popular as a means of modeling time series data such as meteorological data and stock prices. For our task, we use long/short-term memory (LSTM) as the main building blocks of the RNN. LSTMs are able to identify and remember latent features over an unspecified number of time periods, making them a versatile tool in time series prediction.

RNNs, just like other neural networks, are specified as a combination of various parameters such as number and type of layers as well as activation functions. Hence, a necessary step in building RNNs is a systematic evaluation of these parameters to find the best combination for the given task. In addition to specifying the RNN parameters, the input data for RNNs (and neural networks in general) is often pre-processed into a format that makes it easier for RNNs to generate good predictions. Hence, in the remainder of this section we describe these model specification steps: our hyperparameter selection, the resulting network structure, and the preprocessing steps considered to transform our input data.

4.1 LSTM hyperparameter selection

The number and type of layers is a fundamental architecture decision for a neural network. To select our network architecture for the given task, we performed a simple grid search over a range of potential network architectures. Specifically, we systematically varied the number of LSTM layers in the network (from 1 to 4), and the width of each LSTM layer (from 16 to 128 nodes). Each neural network had an additional dense layer for consolidation after the given number of LSTM layers.

Figure 2 shows the loss function for these combinations of LSTM layers and widths of each LSTM layer. Based on this analysis we identified the relatively small, three-layer solution with two LSTM layers and a dense layer for consolidation to show the most promise. Note: for this analysis, we did not use a mini-batch training method such as to reduce variation between training and validation error, given the computational requirements to do so.

4.2 Network structure

Based on our previous hyperparameter selection, our final RNN incorporates elements from both LSTM and dense neural networks. As illustrated in Figure 3, we use a simple, three-layer network composed of two LSTM layers—each 32 units wide—and one dense node to consolidate input from the second LSTM layer to a final predicted value. We chose the LSTM unit to comprise the primary layers of our neural network model due to the temporal element of our data and regression problem, as well as the benefits of LSTM over other types of RNNs. Like other RNNs, the LSTM units can be assembled into layers of the neural network that are able to accept data sequentially (i.e., the data is processed through each unit such that the previous data point is able to be related to the last). The advantage held by LSTM over traditional RNNs is that it possesses three internal "gates" that allow it to explicitly forward or forget data, allowing it to relate data from point to point and "forget" earlier relations over a certain period within the sequence, if these earlier relations turn out to be irrelevant. This functionality provides a mechanism to use information from surrounding inputs to affect a given input while still remaining unaffected by inputs more distant.

To train our neural network, we used the ADAM optimization algorithm [17] with Nesterov momentum [8] and Mean Square Error as our loss function.

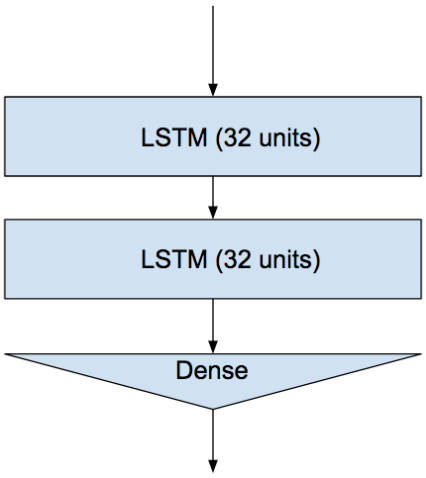

Figure 3: Our neural network architecture composed of two LSTM layers and a dense node to consolidate output.

4.3 LSTM data preprocessing

Our neural network is designed to process tick-level data (as given by the spot price history), rather than points for every time period. We choose to use tick-level data as it is less computationally intensive to train while still providing a strong pricing signal to our network. To further minimize the computational overhead of training we investigate two forms of preprocessing to our input data. One method regularizes pricing data relative to the on-demand price; the other applies exponential smoothing to regularize data for seasonal patterns.

In order to allow for greater generalizability of the model to different spot price series, we have attempted to regularize our data by transforming the raw data into a percentage of the on-demand price. This allows us to better expand one optimal solution from one instance type in one region to all instance types in all regions. While the price of a spot instance does not vary directly with on demand price (i.e., some instance types are in higher demand than others), this allows the network to work with data that are much more similar in terms of actual value. For example, imagine one instance type that is in low demand but very expensive such that the on demand price is $10/hr and the mean spot price is around 20% of this value ($2/hr). Imagine another less expensive instance type in high demand such that the on demand price is $0.50/hr and the mean spot price is 60% of this value ($0.30/hr). It is much less computationally expensive to retrain a network from values between 0.1 and 0.3 to values between 0.5 and 0.7 than it is to retrain a network from values around 10 to values around 0.5. Additionally, in the retraining process, we are much more likely to be marginally altering the specific weighting than we are to be completely reorganizing the function of the network.

The second preprocessing step we explore looks at seasonal regularization at different levels (day, week, year) using Holt-Winters exponential smoothing. This preprocessing step represents a well-tested approach for removing regular and easily quantifiable seasonal trends so that the network itself can focus on predicting more irregular trends of the time series. This seasonal regularization also reduces the computational requirements of the network as it does not have to learn these broad trends.

5 EVALUATION

We used historical Amazon spot pricing data to evaluate our neural network and to compare its performance against that of a baseline statistical model—Autoregressive Integrated Moving Average (ARIMA)—which has been shown to perform well for spot price prediction [1, 33]. We first describe our input dataset and then investigate the training performance of our network and the comparative accuracy of both ARIMA and our network on our historical, not preprocessed, spot pricing data. Finally, we explore the effect of data preprocessing on accuracy.

When reporting on forecast accuracy, we use the Mean Square Error (MSE), defined as:

$$mean\left(\sum_{t}(actualValue_t - predictedValue_t)^2\right)$$

We use MSE as our primary metric for evaluation as it is the loss metric we used to train our network. We also report on the Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE).

5.1 Dataset

In order to validate the use of neural networks for spot price prediction, we arbitrarily picked a single instance type from our historical pricing data: the `c3.2xlarge` Linux instance type from the `us-east-1b` region between September 3, 2016 and September 10, 2016 (as shown in Figure 1). Note: we used historical data as, at the time of this study, there was not sufficient pricing data available from the new spot market. To make the network computationally tractable, so that we could explore different network architectures, we used a subset of 10,000 points (roughly six days) for training and held back the following 2,000 points (roughly two days) for validation. While this data represents a relatively small sample, and only considers a single instance type, availability zone, and region, we believe that it is sufficient to provide initial validation of the benefits of LSTM models.

We define our prediction task as follows: given 50 input observations, forecast the spot price for the subsequent 50 periods (i.e., predict the next 50 observations). We chose 50 observations for this analysis as it represents a window of approximately one hour. For each of these 50 input points, the ARIMA method creates and trains a new model to generate the output of the next 50 predictions; for our neural network, we trained on the 10,000 points for 250 epochs and then applied the model to sequences of 50 input points in the validation dataset.

5.2 Baseline: ARIMA

We use the ARIMA method to produce a baseline prediction. An ARIMA model for forecasting univariate time series data can be parameterized through three attributes, (p, d, q), where p is the

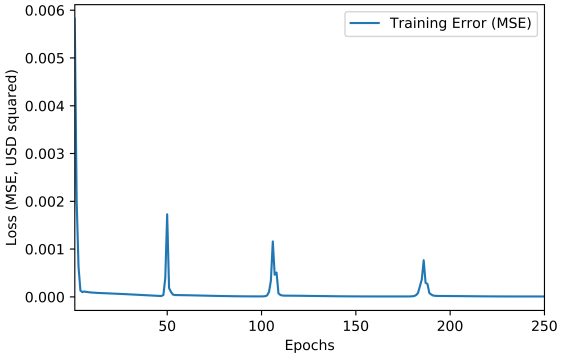

Figure 4: Training MSE (USD squared) over 250 epochs.

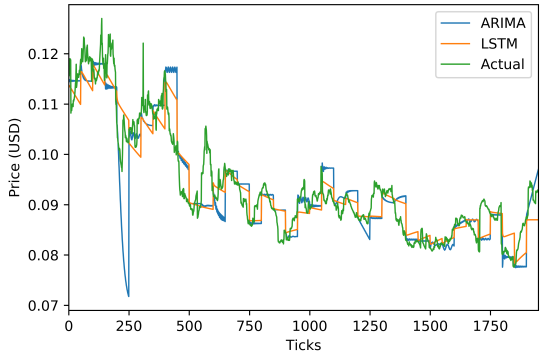

Figure 5: Prediction of the 2,000 point validation set using the ARIMA and LSTM models. Actual data included for reference.

number of autoregressive terms, d the number of differentiation, and q the order of the moving average. Selecting appropriate values for p, d, and q depends on the particular time series. For the subsequent evaluation, we used the `auto.arima` functionality of R to automatically compare several parameterizations and select the parameter combination with the lowest forecast error. We configured the ARIMA model to use a training window of historical data (here: 50 observations) and forecast the next time periods (here: 50 predictions). In general, longer training windows and shorter forecast windows lead to a more accurate forecast.

5.3 Training and validation of LSTM

After exploring various network architectures, we trained our three-layer LSTM network on the 10,000 input data points for 250 epochs. Figure 4 shows the training loss function for this dataset. The disconcerting spikes are caused by the NADAM optimization function we used. Spikes are common with ADAM and are potentially exaggerated via NADAM's Nesterov (N) momentum component. We achieve an in-data training MSE of 7×10^{-6}. (A validation error of 9×10^{-6} was achieved when training on the same data split by Tensorflow with one-third of the data used for validation.) These results were achieved without preprocessing.

5.4 Forecast accuracy

We now compare the accuracy of our neural network with the ARIMA model. Figure 5 shows the predictions of both models as well as the actual `c3.2xlarge` series. Both models track the actual data fairly accurately. Figure 6 shows the MSE for each model on each point. While the errors look similar, the neural network is clearly more accurate than the ARIMA model. The MSE over our 2,000 point validation set was 1.7×10^{-5} for the neural network and 4.2×10^{-5} for ARIMA. Additionally, the Mean Absolute Percentage Error (MAPE) for the ARIMA and LSTM models were 3.76% and 3.35%, respectively. In Table 1 we report Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). In both cases the LSTM model is more accurate than the ARIMA model.

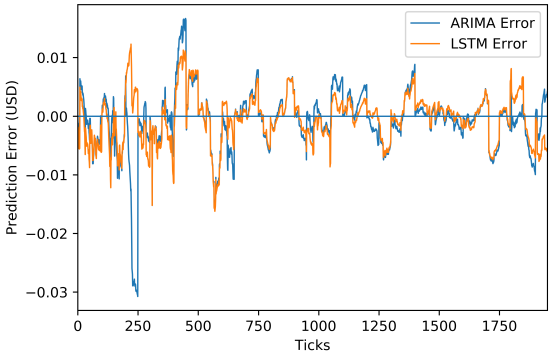

Figure 6: MSE of both ARIMA and LSTM models where a value of "0" would represent a perfect prediction.

Table 1: Root Mean Square Error and Mean Absolute Error for the ARIMA and LSTM models in cents (USD).

	ARIMA	LSTM
Root Mean Square Error (RMSE)	0.557	0.423
Mean Absolute Error (MAE)	0.362	0.320

5.5 LSTM with preprocessing

Our preprocessing steps aim primarily to reduce the computational requirements for training; however, they might also affect accuracy. To evaluate their effect on accuracy we retrained the neural network twice, each time using one preprocessing method. The price regularization method provided no increase (and no decrease) in accuracy. This was expected as this method was intended to reduce training cost by allowing generalizability across many instance types. The seasonal regulation, however, did affect accuracy. In this case, using Holt-Winters exponential smoothing, we were able to

achieve a training MSE of 6.98×10^{-6}—a small improvement over the same network without this preprocessing. This result indicates that the neural network is able to detect a small seasonal trend in the data.

6 SUMMARY

Accurate prediction of spot prices can reduce the cost of using spot instances. Our preliminary investigation of a neural network-based approach shows significant promise for accurately predicting spot prices. Using a three-layer LSTM-based network trained on a 10,000 point dataset of historical spot prices, we observed an average reduction in Mean Square Error of 60% and up to 95% in some cases, when compared with the baseline ARIMA model. This significant reduction suggests that the LSTM model is well suited for predicting spot prices. Such a reduction in error equates to roughly one-half cent per tick for this data and instance type. In other words, our LSTM predictions could allow a user to accurately estimate one-half cent closer to the actual bid price on average. For example, a user with a 12 hour allocation of the `c3.2xlarge` instance type could reduce their expected cost by approximately $5.

The recent changes to the spot market have modified not only the market behavior but also the motivation for predicting spot prices. Users no longer need to calculate a bid for optimal provisioning and the prices are less volatile due to artificial smoothing. These changes are perhaps due to the increasing accuracy of spot prediction methods as highlighted by this paper. We believe that LSTM-based approaches (like that presented here) will also be applicable in the new spot market as a way of predicting prices further into the future. Thus, in future work we first aim to explore the application of LSTM networks to the new spot market. In particular, we are interested in exploring more complex network architectures and using larger training sets across many instance types, availability zones, and regions. Our initial experimentation, albeit with what is now old data, using larger datasets indicates that more complex network architectures might well improve performance. For example, we observed the best performance in our experiments on a larger dataset using a network composed of five layers: a first LSTM layer, two dense layers, a second LSTM layer, and a final dense layer. Finally, predicting spot prices is only the first step in making decisions about how to act in a spot price market in general. Thus, we intend to integrate our LSTM-based prediction models into a decision framework that can help users make decisions in the spot market.

ACKNOWLEDGMENTS

This work was supported in part by NSF award ACI-1550588 and DOE contract DE-AC02-06CH1135.

REFERENCES

[1] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. 2013. Deconstructing Amazon EC2 Spot Instance Pricing. *ACM Trans. Econ. Comput.* 1, 3, Article 16 (Sept. 2013), 20 pages. https://doi.org/10.1145/2509413.2509416

[2] Y. N. Babuji, K. Chard, A. Gerow, and E. Duede. 2016. A secure data enclave and analytics platform for social scientists. In *12th IEEE International Conference on e-Science (e-Science)*. 337–342. https://doi.org/10.1109/eScience.2016.7870917

[3] Kyle Chard, Ravi Madduri, Xia Jiang, Farid Dahi, Michael W. Vannier, and Ian Foster. 2014. A Cloud-based Image Analysis Gateway for Traumatic Brain Injury Research. In *9th Gateway Computing Environments Workshop*. IEEE, 13–16. https://doi.org/10.1109/GCE.2014.8

[4] Ryan Chard, Kyle Chard, Kris Bubendorfer, Lukasz Lacinski, Ravi Madduri, and Ian Foster. 2015. Cost-Aware Cloud Provisioning. In *11th International Conference on e-Science*. IEEE, 136–144. https://doi.org/10.1109/eScience.2015.67

[5] R. Chard, K. Chard, R. Wolski, R. Madduri, B. Ng, K. Bubendorfer, and I. Foster. 2017. Cost-Aware Cloud Profiling, Prediction, and Provisioning as a Service. *IEEE Cloud Computing* 4, 4 (July 2017), 48–59. https://doi.org/10.1109/MCC.2017.3791025

[6] M. B. Chhetri, M. Lumpe, Q. B. Vo, and R. Kowalczyk. 2017. On Estimating Bids for Amazon EC2 Spot Instances Using Time Series Forecasting. In *IEEE International Conference on Services Computing (SCC)*. 44–51. https://doi.org/10.1109/SCC.2017.14

[7] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo. 2003. ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems* 18, 3 (Aug 2003), 1014–1020. https://doi.org/10.1109/TPWRS.2002.804943

[8] Timothy Dozat. 2016. *Incorporating nesterov momentum into adam*. Technical Report.

[9] Weichao Guo, Kang Chen, Yongwei Wu, and Weimin Zheng. 2015. Bidding for Highly Available Services with Low Price in Spot Instance Market. In *24th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 191–202.

[10] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. 2017. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 35 (2017), 18–31.

[11] Xin He, Prashant Shenoy, Ramesh Sitaraman, and David Irwin. 2015. Cutting the Cost of Hosting Online Services Using Cloud Spot Markets. In *24th International Symposium on High-Performance Parallel and Distributed Computing*. ACM.

[12] Ludger Hentschel. 1995. All in the family: Nesting linear and nonlinear GARCH models. *Journal of Financial Economics* 39 (1995), 71–104.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.

[14] Bahman Javadi, Ruppa K. Thulasiram, and Rajkumar Buyya. 2013. Characterizing Spot Price Dynamics in Public Cloud Environments. *Future Gener. Comput. Syst.* 29, 4 (June 2013), 988–999. https://doi.org/10.1016/j.future.2012.06.012

[15] B. Javadi, R. K. Thulasiramy, and R. Buyya. 2011. Statistical Modeling of Spot Instance Prices in Public Cloud Environments. In *4th IEEE International Conference on Utility and Cloud Computing (UCC)*. 219–228. https://doi.org/10.1109/UCC.2011.37

[16] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka. 1990. Stock market prediction system with modular neural networks. In *International Joint Conference on Neural Networks (IJCNN)*. 1–6 vol.1. https://doi.org/10.1109/IJCNN.1990.137535

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Ravi K Madduri, Dinanath Sulakhe, Lukasz Lacinski, Bo Liu, Alex Rodriguez, Kyle Chard, Utpal J Dave, and Ian T Foster. 2014. Experiences building Globus Genomics: a next-generation sequencing analysis service using Galaxy, Globus, and Amazon Web Services. *Concurrency and Computation: Practice and Experience* 26, 13 (2014), 2266–2279.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

[20] Raffaele Montella, David Kelly, Wei Xiong, Alison Brizius, Joshua Elliott, Ravi Madduri, Ketan Maheshwari, Cheryl Porter, Peter Vilter, Michael Wilde, Meng Zhang, and Ian Foster. 2015. FACE-IT: A science gateway for food security research. *Concurrency and Computation: Practice and Experience* 27, 16 (2015), 4423–4436.

[21] T. M. Sands, D. Tayal, M. E. Morris, and S. T. Monteiro. 2015. Robust stock value prediction using support vector machines with particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC)*. 3327–3331. https://doi.org/10.1109/CEC.2015.7257306

[22] Prateek Sharma, Stephen Lee, Tian Guo, David Irwin, and Prashant Shenoy. 2015. Spotcheck: Designing a derivative iaas cloud on the spot market. In *10th European Conference on Computer Systems*. ACM, 16:1–16:15.

[23] A. Sheta, S. Ahmed, and H. Faris. 2015. A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index. *International Journal of Advanced Research in Artificial Intelligence* 4, 7 (2015), 55–63.

[24] David Silver, , Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.

[25] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering

Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *CoRR* abs/1712.01815 (2017). http://arxiv.org/abs/1712.01815

[26] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Landrent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (2017), 354–359.

[27] Supreeth Subramanya, Tian Guo, Prateek Sharma, David Irwin, and Prashant Shenoy. 2015. SpotOn: A batch computing service for the spot market. In *6th ACM Symposium on Cloud Computing*. ACM, 329–341.

[28] S. Tang, J. Yuan, and X. Y. Li. 2012. Towards Optimal Bidding Strategy for Amazon EC2 Cloud Spot Instance. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*. 91–98. https://doi.org/10.1109/CLOUD.2012.134

[29] Jonathan L. Ticknor. 2013. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications* 40, 14 (2013), 5501–5506. https://doi.org/10.1016/j.eswa.2013.04.013

[30] Volodymyr Turchenko, Vladyslav Shults, Iryna Turchenko, Richard Wallace, Mehdi Sheikhalishahi, Jose Vazquez-Poletti, and Lucio Grandinetti. 2014. Spot Price Prediction for Cloud Computing Using Neural Networks. *International Journal of Computing* 12, 4 (2014), 348–359.

[31] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *CoRR* abs/1609.03499 (2016). arXiv:1609.03499 http://arxiv.org/abs/1609.03499

[32] R. M. Wallace, V. Turchenko, M. Sheikhalishahi, I. Turchenko, V. Shults, J. L. Vazquez-Poletti, and L. Grandinetti. 2013. Applications of neural-based spot market prediction for cloud computing. In *7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, Vol. 02. 710–716. https://doi.org/10.1109/IDAACS.2013.6663017

[33] Rich Wolski, John Brevik, Ryan Chard, and Kyle Chard. 2017. Probabilistic Guarantees of Execution Duration for Amazon Spot Instances. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 18, 11 pages. https://doi.org/10.1145/3126908.3126953

[34] Murtaza Zafer, Yang Song, and Kang-Won Lee. 2012. Optimal bids for spot VMs in a cloud for deadline constrained jobs. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*. IEEE, 75–82.

[35] Liang Zheng, Carlee Joe-Wong, Chee Wei Tan, Mung Chiang, and Xinyu Wang. 2015. How to Bid the Cloud. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 71–84. https://doi.org/10.1145/2829988.2787473