Capstone Project

Samar Shaaban Abdelfattah Haytamy

Machine Learning Engineer Nanodegree

October 23, 2018

Definition

Project Overview

The cloud computing idea is based on reusability of IT capabilities. The enterprises as well as individuals can use these Cloud-based services as an partial solution to their operational and business problems. The leading cloud computing providers(Google, Microsoft, Amazon, E-Bay, IBM, etc) have built An online marketplaces to facilitate the publication and searching of different types of cloud services in a more suitable way. The marketplaces provide services on demand, paying per usage and managing automatic service elasticity to meet users' requirements (1).

The Cloud consumer usually needs to use Cloud services as a partial solution to his requirements. So, the appropriate Cloud services have been composed and provided as a single virtual service to the Cloud consumers.

I this project, I create a desktop application that capable of composing multiple services from different cloud providers (e.g. IBM, Google, Rackspace,). The application uses a predictor trained using QoS (quality of services) dataset[1] to predict future provision values to accurately select providers.

The project was inspired by this paper[2].

Problem Statement

The goal is to create a service composition application capable of recommending best providers to contract with; the tasks involved are the following:

1. Download and preprocess the QoS attributes data.

2. Train a predictor that can predict future provision QoS values

3. Develop the composer (broker) application

4. Rank or recommend the best cloud providers.

The final application is expected to be useful for cloud end users (consumers) because it help him to contract with the best providers appropriate to his requirements.

---

[1] https://github.com/SamarShabanCS/Math_for_ML/tree/master/time%20series%20data%20QoS

[2] https://ieeexplore.ieee.org/document/6964807

Metrics

Root Mean Square Error (RMSE): RMSE is used to evaluate the performance of a prediction model.

Let assume that the time series of an individual attribute in the QoS history fits by many prediction models. If the predicted time series are ($\widehat{Q_{1t}}, \widehat{Q_{2t}}, \dots, \widehat{Q_{nt}}$), the prediction error is calculated using the following equation. A lesser value of RMSE imposes a better prediction model.

$$\text{RMSE (i)} = \sqrt{\frac{\sum_{i=1}^{m}(\widehat{Q_{it}} - Q_{it})^2}{m}}$$

Analysis

Data Exploration

- A synthetic dataset is be used to represent the cloud consumer preferences or his requirements which are his preferred weights for response time, throughput, availability for the required service :
  It will be a historical time series data follow Gaussian distribution. It is generated using the TimeSynth open source library (https://github.com/TimeSynth/TimeSynth).

  - The end user preferences data will be 3 series of floating point values of length 1440
    Data Shape= 1440 rows × 3 columns

- The cloud providers' data set will be represented using a real cloud service data (2) which is updated in (3). It contains 5 historical time series for 100 cloud service providers collected through 6 months as 28 time slots as follows: 1. Availability 2. Max Response time 3. Min Response time 4. Avg. Response time 5. Throughput.
  - The values are floating point numbers

It is available here:

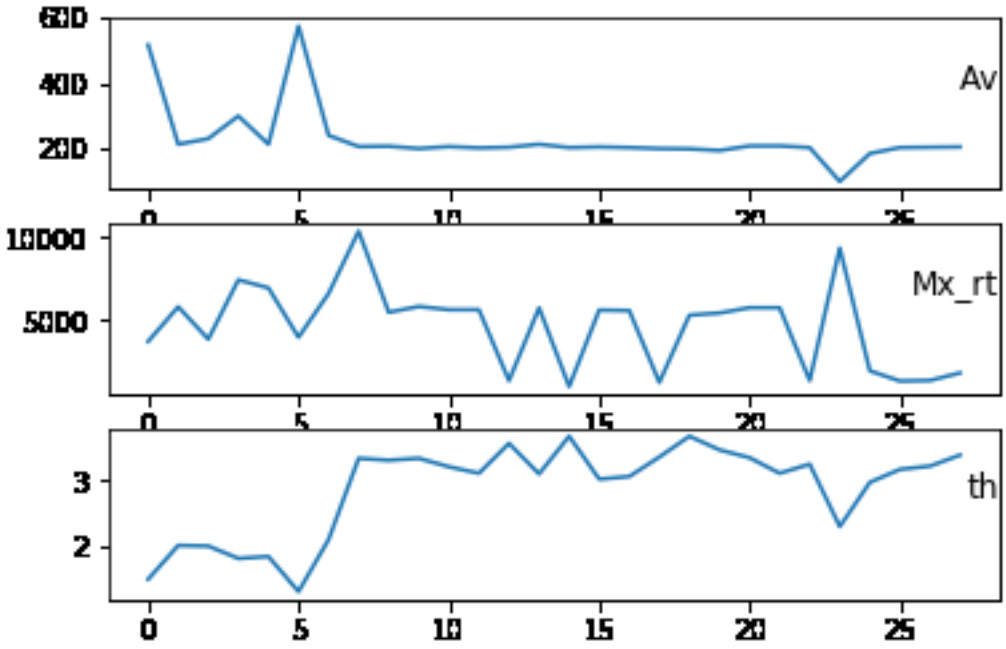
(https://github.com/SamarShabanCS/Math_for_ML/tree/master/time%20series%20data%20QoS)

Exploratory Visualization

- The following plot show the distribution or history of the QoS attributes (Av: availability, Mx_rt: maximum response time, Th: throughput) provisioned from one provider selected randomly

from the cloud providers' data set. It shows that there is strong negative correlation ship between response time and throughput, when the response time is slower; the throughput is higher and vice versa. When calculating the correlation operator between these two parameters, it is found it is equal to -0.67 which ensure the mentioned description. This information helps us to use machine learning model that is capable of capturing this relation between Th and Rt.

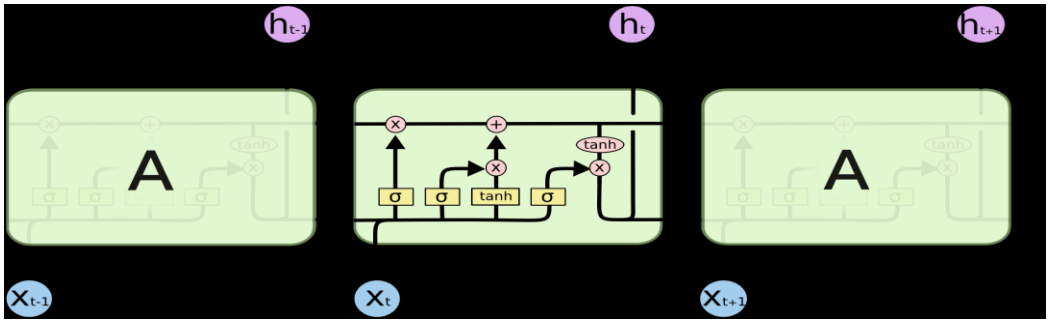
Also, the QoS values are varied, so they need to preprocessed (scaling).



Algorithms and Techniques

The predictor is the LSTM [3](long short term memory) which is a type of recurrent neural network capable of learning order dependence in sequence prediction problems (Time series in our problem). LSTMs are designed to avoid the long-term dependency problem which is in traditional RNN. Remembering information for long periods of time is practically their default behavior. LSTms are illustrated in the following figure:

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/



The following parameters can be tuned to optimize the predictor:

- Input Parameters

  Preprocessing and Normalization

- Neural Network Architecture

  Model Type (MLP or LSTM; mostly focused on LSTM)

  Number of Layers (how many layers of nodes in the model)

  Number of Nodes (how many nodes per layer)

- Training Parameters

  Training / Test Split (how much of dataset to train versus test model on

  Look back (how many prior days are included in the input sequence

  Batch Size (how many time steps to include during a single training step

  Optimizer Function (which function to optimize by minimizing error

  Epochs (how many times to run through the training process

Benchmark

According to my research, the nearest and closest paper to the mentioned problem is “Long-Term QoS-Aware Cloud Service Composition Using Multivariate Time Series Analysis” (4). So, it will be used as the benchmark model. It propose a multiple QoS prediction model (MQPM) by using the Arima model to predict the QoS values then compose the services that match cloud consumer requirements by using the Euclidian distance.

The performance of MQPM of its predicted QoS attributes:

Prediction model	RMSE		
	Avg. throughput	Avg. response time	Avg. availability
MQPM	0.32	59	236

Methodology

Data Preprocessing

The following steps are done in the preprocessing phase before building and training the prediction model:

- Generate the user QoS preferences values using TimeSynth generator. This was done in a Jupyter notebooks (titled “END_user_preferences.ipynb”

- Read the providers’ QoS history 100 csv files.
- put the key data points into a Pandas DataFrame for ease of organization and visualization
- Define train test split ratio
- Create the training and test datasets
- Define Look back (how many prior days to include at each time step)
- convert from series format to supervised format(X,Y)
- Normalize data (normalize 0.0 to 1.0 for better performance)

The following graph show sample of preprocessed data

Out[8]:

	var1(t-12)	var2(t-12)	var3(t-12)	var4(t-12)	var5(t-12)	var1(t-11)	var2(t-11)	var3(t-11)	var4(t-11)	var5(t-11)	...	var1(t+2)	var2
12	0.045455	0.145173	0.248548	0.532258	0.547170	0.022727	0.155122	0.145394	0.483871	0.443397	...	0.204545	0.38
13	0.022727	0.155122	0.145394	0.483871	0.443397	0.022727	0.020265	0.150456	0.725806	0.688679	...	0.204545	0.25
14	0.022727	0.020265	0.150456	0.725806	0.688679	0.022727	0.379145	0.078506	0.435484	0.424529	...	0.204545	0.26
15	0.022727	0.379145	0.078506	0.435484	0.424529	0.000000	0.436625	0.103154	0.661290	0.518868	...	0.204545	0.41
16	0.000000	0.436625	0.103154	0.661290	0.518868	0.022727	0.119749	0.139419	0.838710	0.745283	...	0.272727	0.02
17	0.022727	0.119749	0.139419	0.838710	0.745283	0.045455	0.091747	0.253942	0.274194	0.283019	...	0.204545	0.13
18	0.045455	0.091747	0.253942	0.274194	0.283019	0.022727	0.109064	0.089295	0.661290	0.537736	...	0.181818	0.21
19	0.022727	0.109064	0.089295	0.661290	0.537736	0.022727	0.103906	0.096266	0.306452	0.330189	...	0.204545	0.47
20	0.022727	0.103906	0.096266	0.306452	0.330189	0.022727	0.000000	0.000000	0.129032	0.132076	...	1.000000	0.19
21	0.022727	0.000000	0.000000	0.129032	0.132076	0.000000	0.315033	0.226307	0.387097	0.283019	...	0.977273	0.23
22	0.000000	0.315033	0.226307	0.387097	0.283019	0.000000	0.029477	0.096929	0.887097	0.830189	...	1.000000	0.24
23	0.000000	0.029477	0.096929	0.887097	0.830189	0.250000	0.401990	0.427054	0.096774	0.084906	...	1.000000	0.61
24	0.250000	0.401990	0.427054	0.096774	0.084906	0.227273	0.073692	0.203154	1.000000	1.000000	...	1.000000	0.12

Implementation

The implementation process can be split into two main stages:

1. The predictor training stage
2. The application development stage

During the first stage, the predictor was trained on the preprocessed training data. This was done in a Jupyter notebooks (titled “prov_prediction model.ipynb”: for forecasting the cloud providers’ QoS attributes values and “end_user_prediction model .ipynb”: for forecasting the user preferences), and can be further divided into the following steps:

1. Load both the training and validation sequences into memory, preprocessing them as described in the previous section(convert series into (X,Y) or the supervised form, normalize data,..)
2. Define the network architecture and training parameters
3. Define the loss function, MSE
4. Train the network, logging the validation/training loss
5. Plot the logged values
6. If the MSE is not low enough, return to step 2
7. Save and predicted QoS Values

The following graph is the summary of the network architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_17 (LSTM)               (None, 60)                15840
_________________________________________________________________
dropout_15 (Dropout)         (None, 60)                0
_________________________________________________________________
dense_15 (Dense)             (None, 20)                1220
=================================================================
Total params: 17,060
Trainable params: 17,060
Non-trainable params: 0
_________________________________________________________________
```

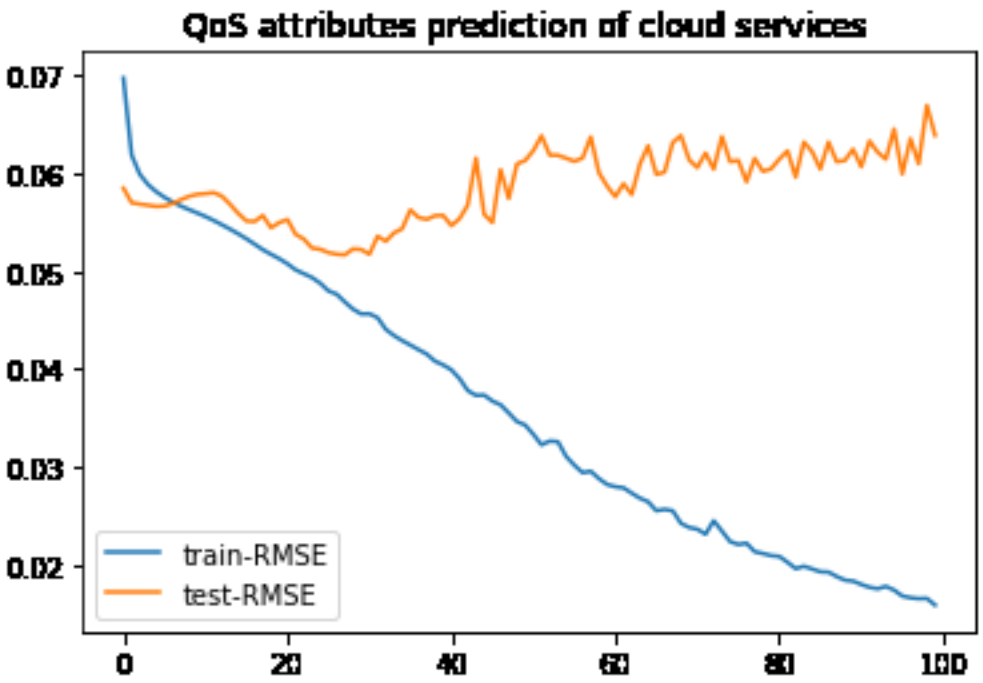
The application development stage which was done in a Jupyter notebooks titled “serviceComposition.ipynb” can be split into the following steps:

1. Load future provisioned QoS attributes values of the 100 providers.
2. Load future preferences QoS attributes values of the user.
3. Split 100 providers services into three classes
4. Define composition function, particle class, particle swarm class (PSO)
5. Run PSO class to obtain the recommended services.

Refinement

The initial solution has RMSE=1939.078. After adding a drop out layer the RMSE reduced to 1939.078.

The following plot of the training/validation losses has a divergence indicates overfitting, which can be addressed by adding dropout layer, or reducing the model complexity (e.g. reducing the number of layers), among other techniques.



Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model. The final architecture and hyper parameters were chosen because they performed the best among the tried combinations.

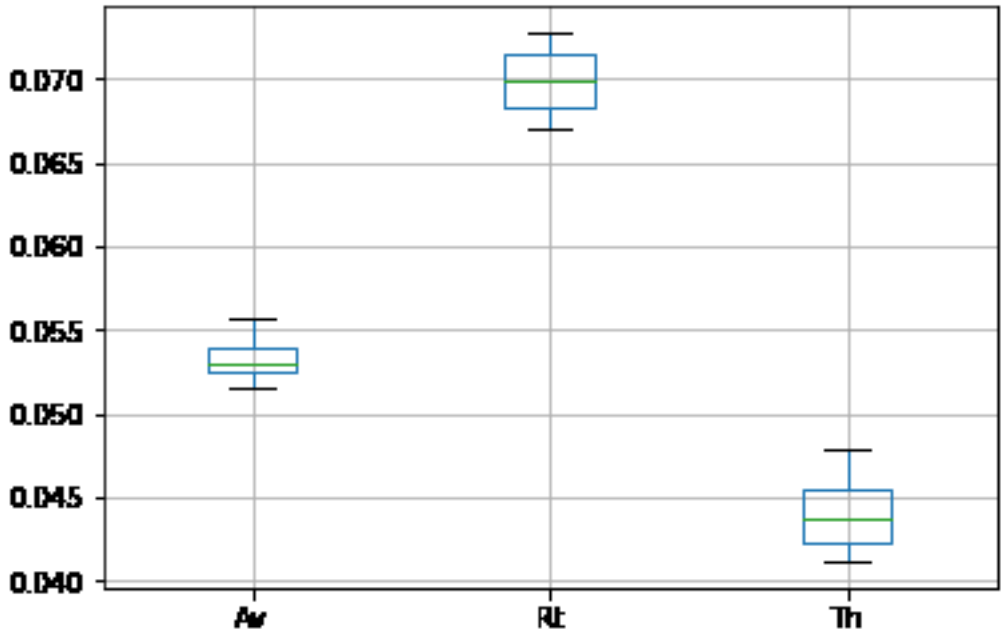
For a complete description of the final model and the training process, the following graph shows the network architecture:

It trained for 150 iterations and each has batch size of 2.

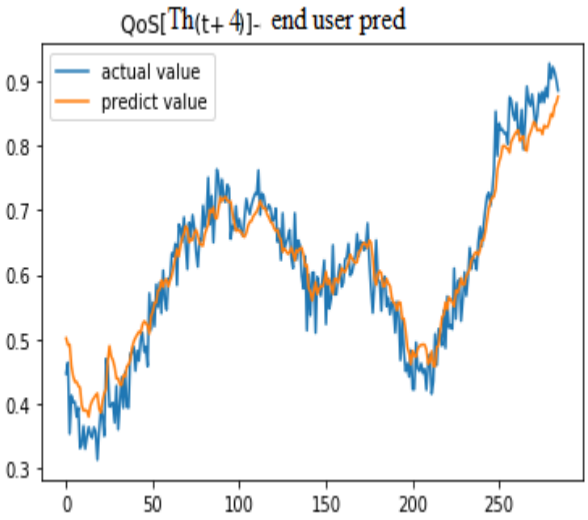
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_17 (LSTM)               (None, 60)                15840
_________________________________________________________________
dropout_15 (Dropout)         (None, 60)                0
_________________________________________________________________
dense_15 (Dense)             (None, 20)                1220
=================================================================
Total params: 17,060
Trainable params: 17,060
Non-trainable params: 0
_________________________________________________________________
```

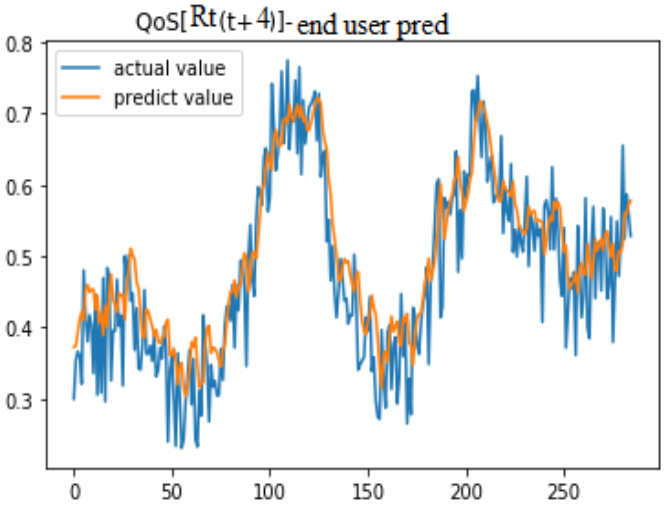
Justification

- The average test RMSE of the user preferences for all attributes is shown through the following box graph:

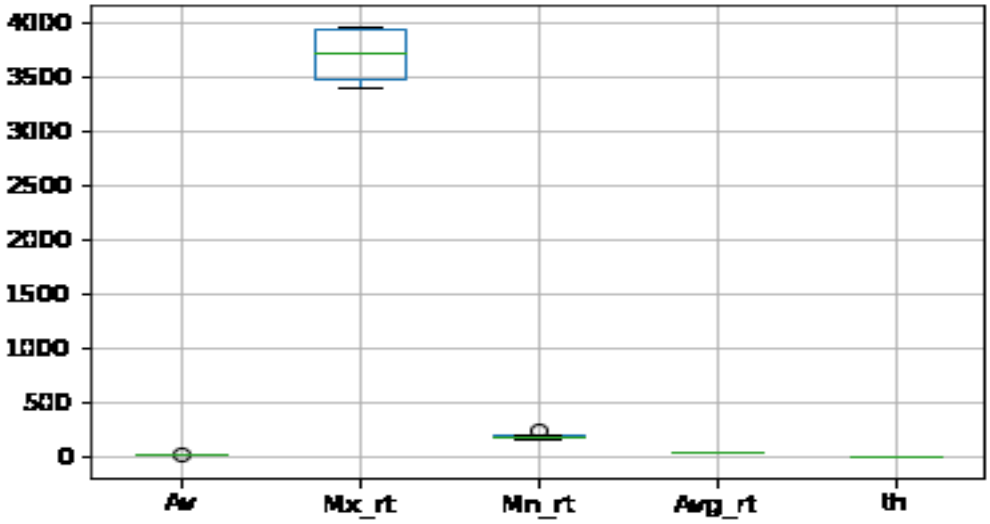


This is significantly less than the RMSE of the benchmark in all the attributes. The following graphs show the predicted throughput and response time at time t+4.



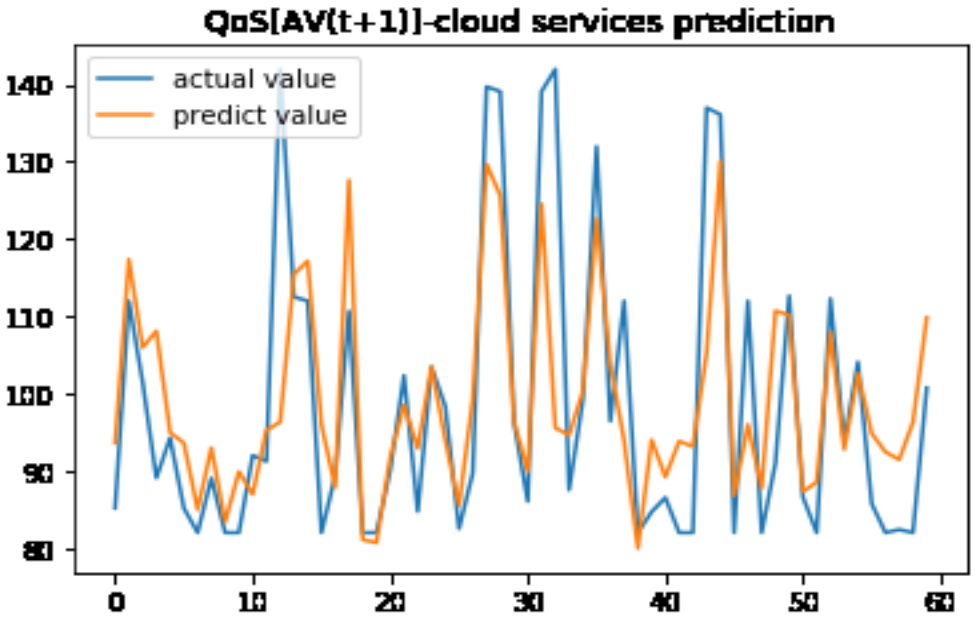


- According to the providers provisioned QoS attributes, the average test RMSE of all attributes is shown through the following box graph:

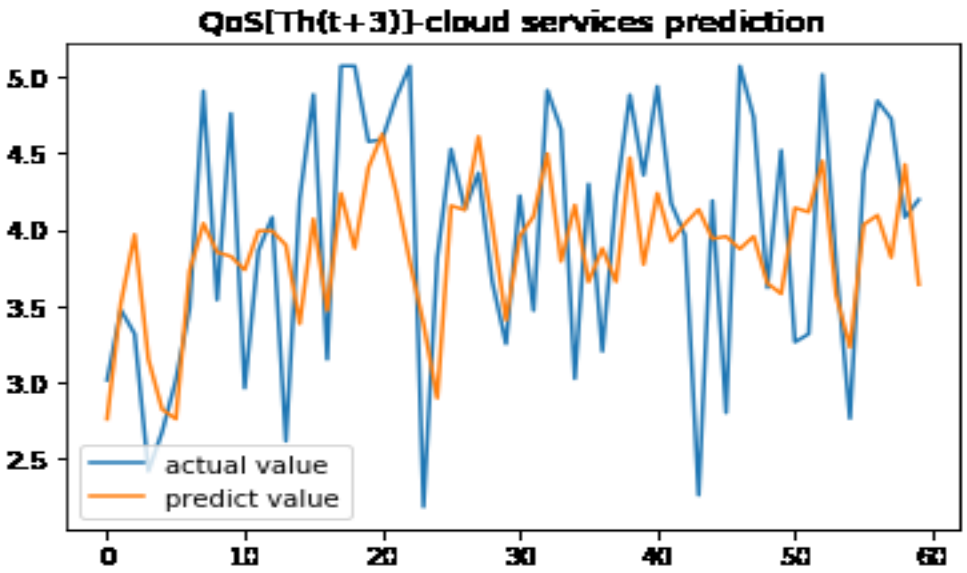


It is seen that the best attributes to use in the composition application is Av, th, and Avg_rt. They have also less RMSE of the benchmark results.

- The following graph show sample of the availability prediction at time step t+1. It is acceptable results according to the benchmark.



- The following graph show sample of the throughput prediction at time step t+4.



Conclusion

Free-Form Visualization

- The following graph shows the output of the composition application:

```
FINAL:
best global position:
 [[15 10  6]
 [ 0  7 24]
 [21  0 10]
 [ 0  0 33]]

 best global target value:    5.965020856760001

 position[ [7, 10, 0, 0, 0, 24, 21, 0, 0, 0, 0, 33]
local_best_fitness_value[ 5.965020856760001

 position[ [7, 10, 3, 0, 14, 33, 21, 0, 4, 0, 0, 33]
local_best_fitness_value[ 5.78743868536

 position[ [19, 10, 0, 0, 10, 33, 21, 0, 9, 0, 0, 33]
local_best_fitness_value[ 5.6324023489399995

 position[ [0, 10, 13, 0, 7, 22, 21, 0, 19, 0, 0, 33]
local_best_fitness_value[ 5.71154732852

 position[ [15, 20, 27, 0, 0, 0, 22, 0, 33, 0, 0, 33]
local_best_fitness_value[ 5.6582854231
Out[28]: <__main__.PSO at 0x7fc658af5eb8>
```

It shows that according to the user preferences: the user should contract

In the first time period with

the provider number 15 at class 1

Provider number 10 at class 2

Provider number 6 at class 3 and so on,

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found
2. The data was downloaded and preprocessed
3. A benchmark was determined for the predictor
4. The predictor was trained using the data (multiple times, until a good set of parameters were found)
5. The composition application was adapted to use the results of the predictor
6. Use the composition application to remark the cloud providers and contract with the best ones.

I found steps 4 the most difficult, as I had to familiarize myself with LSTM model which was a model that I was not familiar with before the project. Also, the data set for both the user and the cloud providers.

Also, according to the cloud end user preferences, I use the synthetic generator Timesynth, which is the first time to use it.

Also, according the cloud providers QoS values, the used data set has a small length which is only 28 time slot. This hampers to get high performance (lesser RMSE).

Honestly, I tried to extend the length of the dataset, but I failed.(if you have a practical method that extend the time series dataset to keep the correlation ship between the parameters without using randomization)

As for the most interesting aspects of the project, I'm also happy about getting to develop LSTM model using multi variables with multiple lags to predict multiple time step. Also, I'm happy about using Keras which using tensorflow backend, as I believe it will be the deep learning library in the future.

Improvement

To achieve high performance we can:

1. Use more capable hardware and use the tensorflow GPU version instead of CPU one.
2. Get more dataset, this will significantly help in getting lesser RMSE.

References:

1. Cloud services. [Online] , http://www.webopedia.com/TERM/C/cloud_services.html.

2. *Large-scale longitudinal analysis of soap-based and restful web services.* **W. Jiang, D. Lee and S. Hu.** Honolulu, HI, USA : the 2012 IEEE 19th International Conference on Web Services, 2012. Proc. Web Services (ICWS), 2012 IEEE 19th International Conference on. pp. 218–225.

3. **Haytamy S.S., Kholidy H.A., Omara F.A.** (2018) ICSD: Integrated Cloud Services Dataset. In: Yang A. et al. (eds) Services – SERVICES 2018. SERVICES 2018. Lecture Notes in Computer Science, vol 10975. Springer, Cham.

4. **Z. Ye, S. Mistry, A. Bouguettaya and H. Dong.** Long-Term QoS-Aware Cloud Service Composition Using Multivariate Time Series Analysis. *IEEE Transactions on Services Computing.* june 2016, Vol. 9, 3, pp. 382-393. doi: 10.1109/TSC.2014.2373366.