COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

IEOR Deep Learning
Final Project Report

AutoML2

Jing Li
Jinghan Liu
Yufei Jin

Supervised by
Prof. Ali Hirsa, Department of Industrial Engineering and Operations Research, Columbia University

Jan 2022

Abstract

Our project focuses on hyperparameter tuning of the LSTM model. We mainly explored the Bayesian Optimization method for detecting the best hyperparameter set, which consists of three hyperparameters. Throughout this report, we will explain the work done by previous groups, the original dataset, our assumptions, our implementation, our results for the Bayesian Optimization method, and the future work that can be done to make the approach better.

Keywords: Bayesian Optimization, Auto ML, Hyper Parameter Tuning, LSTM

Content

1 Executive Summary

1.1 Goal of the Project

This project aims to implement and tune the hyperparameter set of the LSTM model on a fixed dataset using the Bayesian Optimization method. Our group, which consists of Jing Li, Jinghan Liu, and Yufei Jin, took over this project from the previous group and continued to implement and test the performance of the Bayesian Optimization method.

1.2 Previous Group Work

The initial group (Arun Varghese, Ananya Jalan) tested out the hyperparameter tuning method using Grid Search and Random Search. The second group (Hantao Liu) mainly used Tree-structured Parzen estimators (TPE) for tuning the learning rate parameter. We utilized the same LSTM structure as previous groups and tested the Bayesian Optimization performance using the same dataset.

1.3 Our Contribution and Findings

Our team was able to implement the Bayesian Optimization method mainly using the package BayesOpt. We used the inverse of Root Mean Square Error (RMSE) as the performance evaluation matrix and specifically tested the hyperparameter sets consisting of learning rate, batch size, and num epochs. We split the potential range of each hyperparameter into small intervals and tested Bayesian Optimization performance on all possible hyperparameter interval combinations. After getting the best hyperparameter set returned by BayesOpt, we zoomed in and verified the performance of Bayesian Optimization on a smaller surface surrounding the best hyperparameter sets. We conclude that for most cases, the Bayesian Optimization method returns the best hyperparameter combination.

We also find that Bayesian Optimization works well in robustness check. Through the plotting of the relationship between RMSE and learning rate, we could see that our result is robust. What's more, we also take a closer look at how different acquisition functions affect the performance on RMSE, through the efficiency search, we find that in our case, Gaussian Process Regressor works best on efficiency search and achieve the best result using least number of searches on learning rate.

1.4 Future Work

In our project scope, we mainly tested the hyperparameter combination using learning rate and batch size. The future groups can extend the test to other possible hyperparameter combinations and work on finding implementations relating to early stop methodology. A good early stop mechanism can help the future group to shorten the run time of each test and give them more potentials for researching the Bayesian Optimization method. In addition, the future groups can also test out the Bayesian Optimization method using other packages or work on exploring other AutoML methods such as DEEP-BO (Diversified, Early-termination-Enabled, and Parallel

Bayesian Optimization) for hyperparameter selection. Last but not least, future groups should utilize cloud resources to facilitate their model training and get rid of the limitation of computer power.

2 Data

Our group used the same dataset as previous groups. The dataset consisted of excess return data of large-cap funds. The LSTM model focuses on predicting the excess return of these large-cap funds. The data has already been preprocessed, and the preprocessing step includes using parameters such as feat window, pm window, and sample window. The tuning step for these parameters is not in the scope of our groups' work.

3 Bayesian Optimization

Bayesian Optimization is an approach for optimizing objective functions best-suited for continuous domains and tolerates stochastic noise in function evaluations. It builds an initial surrogate for the objective and then uses a Bayesian machine learning technique to improve the posterior distribution estimation. The Bayesian machine learning technique is mainly based on Gaussian process regression. An acquisition function defined from the surrogate is then used for sample generating. [1]

3.1 Bayesian Optimization Implementation

Our group is interested in testing the Bayesian Optimization performance on the three main hyperparameters defined in the fixed LSTM model structure: learning rate, batch size, and num epochs. Due to the limit of computational power, our group mainly tested learning rate on range 0.001 to 0.05 and Batch Size on range 64-256. After specifying the potential range for learning rate and batch size, our group split the range into smaller interval combinations:

Group 1 – Learning rate 0.02-0.03, Batch Size 64-128
Group 2 – Learning rate 0.02-0.03, Batch Size 128-256
Group 3 – Learning rate 0.03-0.04, Batch Size 64-128
Group 4 – Learning rate 0.03-0.04, Batch Size 128-256
Group 5 – Learning rate 0.04-0.05, Batch Size 64-128
Group 6 – Learning rate 0.04-0.05, Batch Size 128-256
Group 7 – Learning rate 0.01-0.02, Batch Size 64-128
Group 8 – Learning rate 0.01-0.02, Batch Size 128-256
Group 9 – Learning rate 0.001-0.01, Batch Size 64-128
Group 10 – Learning rate 0.001-0.01, Batch Size 128-256

These intervals are used as the search space boundaries for the hyperparameters.

For the objective function, our group used the inverse of Root Mean Square Error that was calculated on the testing set.

For the sake of computational power, 50 steps of iteration are used for each group.

After specifying these key parameters, our group ran the Bayesian Optimization function defined in the library BayesOpt on each testing group and found the best hyperparameter combination that maximizes the objective function for each testing group.
In order to verify the Bayesian Optimization methodology performance, our group first plotted the surface generated by the trajectory of each run. Our group used order of four polynomial to fit the surface. Next, our group used:

[best learning rate – 0.001, best learning rate + 0.001] as the new boundary for learning rate and
[best batch size – 20, best batch size + 20] as the new boundary

for batch size for each testing group and rerun the Bayesian Optimization in order to generate a closer look surrounding the optimized result point. The trajectory surface is also plotted for each run. An order of six polynomial is used to fit the surface.

3.2 Robustness Check

After using Bayesian optimization method to get the optimized result, we checked the robustness of our result and explored how the inverse of RMSE evolves with the change of learning rate. In this process, we foundd the specific region where we could get a robust optimized result of the objective function, i.e., the inverse of RMSE. And we also enhanced the findings in 3.1 section where we split the range into smaller interval combinations and checked the robustness in each small interval.

3.3 Embedded Acquisition Function

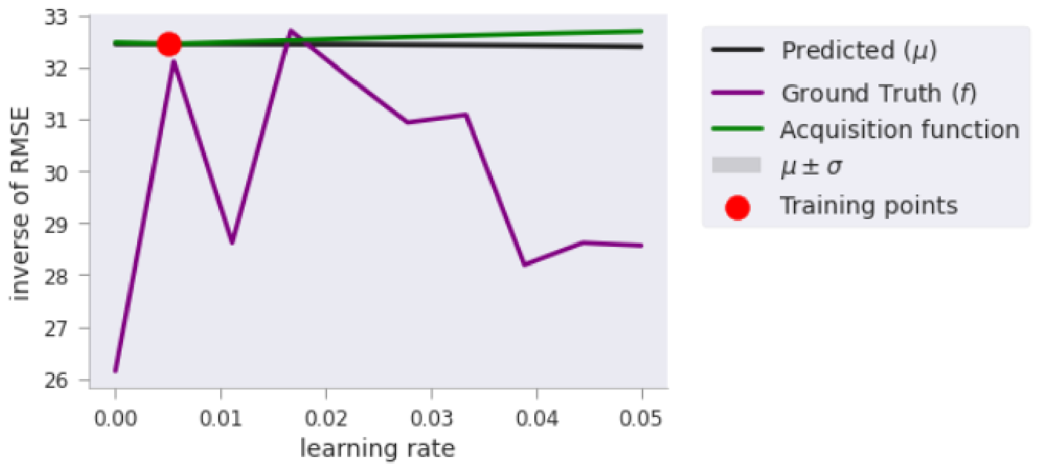
We are curious about how Bayesian optimization finds its optimal points.

Bayesian optimization works by constructing a posterior distribution of functions (gaussian process) that best describes the function we want to optimize. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not.

As we iterated over and over, the algorithm balances its needs of exploration and exploitation taking into consideration what it knows about the target function. At each step a Gaussian Process is fitted to the known samples (points previously explored), and the posterior distribution, combined with an exploration strategy (different acquisition functions), are used to determine the next point that should be explored.

Different acquisition functions lead to different paths that are toward the optimal point. We figured out how to apply different acquisition functions to get the better optimization result, i.e., which acquisition function will lead to the optimal result using the least times of search and thus will lead to the highest efficiency.

3.3.1 How Acquisition Function Works



The function value near the location of the just added point (red point) is high. But as we go far from the red point, we see that our uncertainty increases to a maximum. Then we are going to search the point whose uncertainty is high, which is also the maximum of the acquisition function (the point of 0.05 learning rate is the next searching point).

3.3.2 Different Acquisition Functions

(a) Standard Gaussian Process

We fit the Gaussian Regressor (gp) on some training data under the assumption of Gaussian distribution. Then we use the fitting gaussian regressor to define the acquisition function and let the point where the maximized acquisition function value lie in be the next query point.

(b) Probability of Improvement

The idea behind the algorithm is fairly simple - choose the next point as the one which has the highest probability of improvement over the current max μ^+.

- We have two points of similar means (of function values (RMSE in our case)). We now want to choose one of these to obtain the labels or values. We will choose the one with higher variance. This basically says that given same exploitability, we choose the one with higher exploration value.
- We have two points having same variance. We would now choose the point with the higher mean. This basically says that given same explorability, we will choose the one with higher exploitation value.

Detailed procedure is shown as follows:

1. Let μ^+ be the current highest value of the function
2. Let ε be close to zero
3. Choose $x^* = argmax(P(f(x)) > \mu^+ + \varepsilon)$

(c) Expected Improvement

The utility function is defined as $u(x) = \max(0, f' - f(x))$。 We want to find the x where f values of the next query point is less than the known f'.

(d) Thompson Sampling

The idea is to sample functions within upper and lower probabilistic bounds of a regressor; one can then optimize on these functions and chose the next query point to be the argmax of the sampled_f function.

Thompson Sampling is general enough to be useful even when we have Bernoulli (the domain of x is spatially independent) distributions modeling the function F, instead of Gaussian Process.

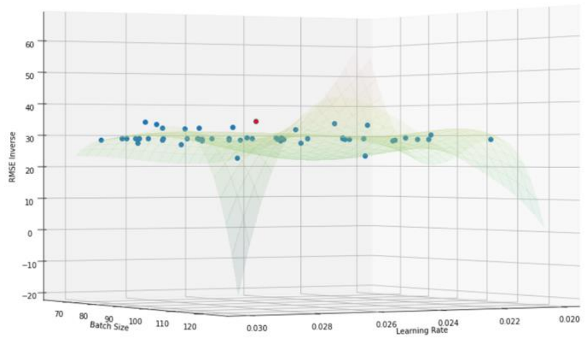
4 Results

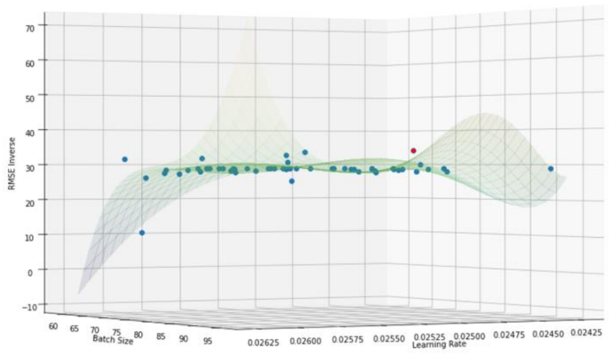
In this section, our group will display the trajectory surface generated from each original run and the zoomed-in run.

4.1 Bayesian Optimization Results

We colored the optimized point returned in red and all the other trajectory points in blue.

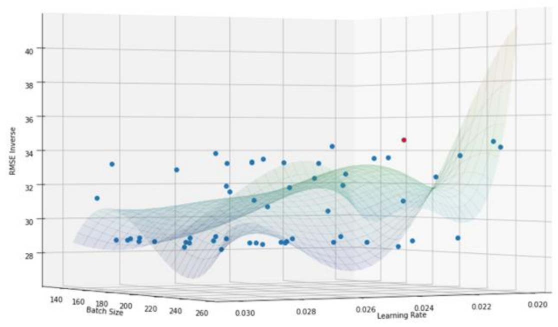
Group 1 Original run and zoomed-in run:

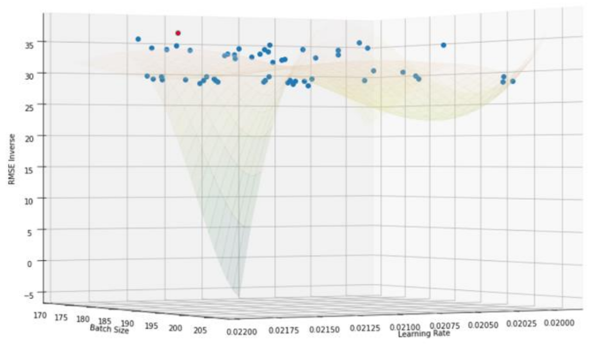




For Group 1, even though the optimized point is not on an apparent up surface, it still achieves a good RMSE inverse estimation.

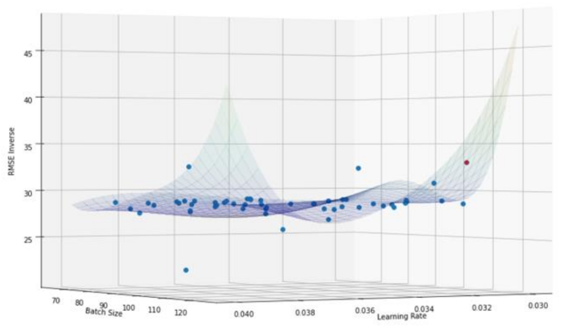
Group 2 Original run and zoomed-in run:

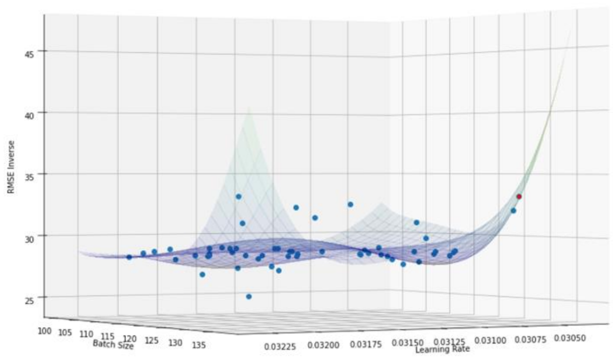




We have the same conclusion for Group 2 as Group 1.

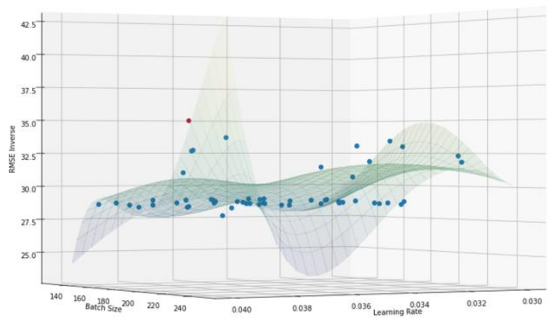
Group 3 Original run and zoomed-in run:

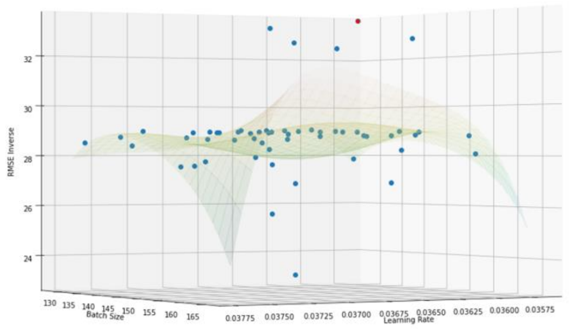




For Group 3, we can see that the optimized point is on an apparent up surface, and we conclude that Bayesian Optimization is well behaved under this case.

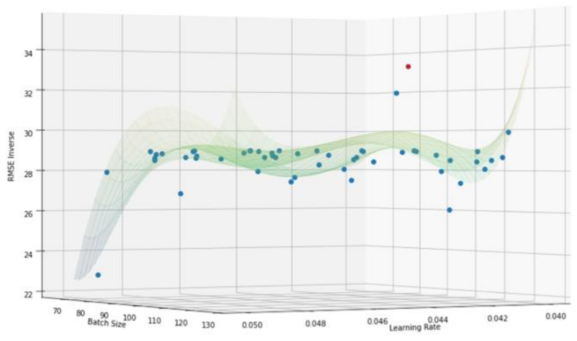
Group 4 Original run and zoomed-in run:

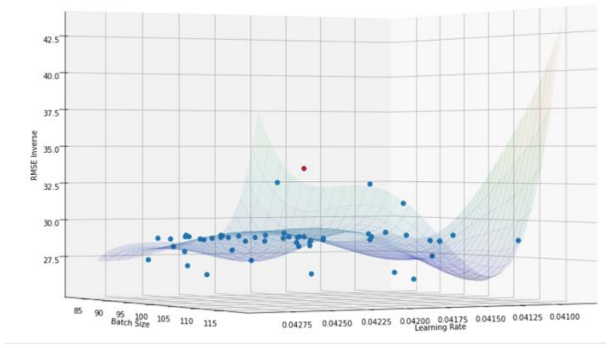




For Group 4, we can see that the optimized point is on an apparent up surface, and we conclude that Bayesian Optimization is well behaved under this case.

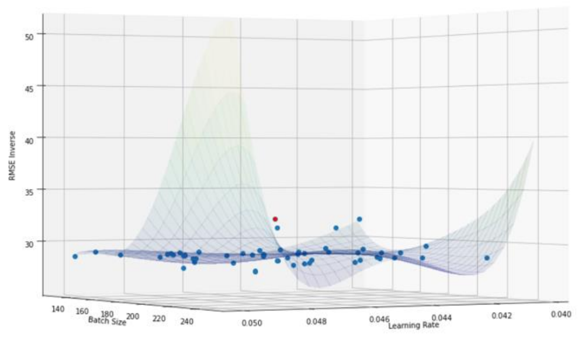
Group 5 Original run and zoomed-in run:

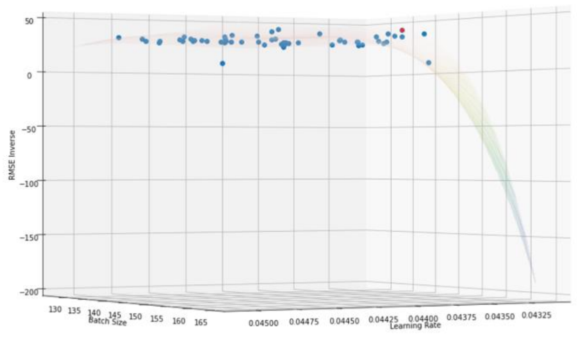




For Group 5, even though the optimized point is not on an apparent up surface, it achieves a decent high RMSE inverse estimation.

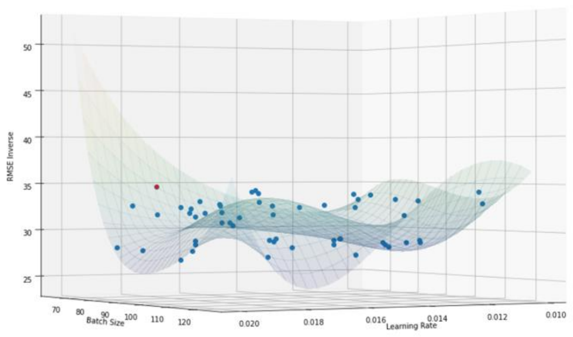
Group 6 Original run and zoomed-in run:

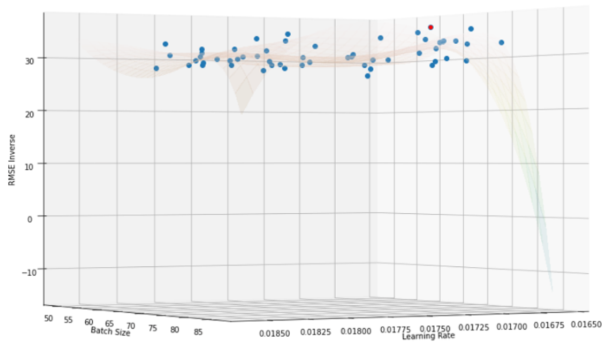




For Group 6, we can see that the optimized point is on an apparent up surface, and we conclude that Bayesian Optimization is well behaved under this case.

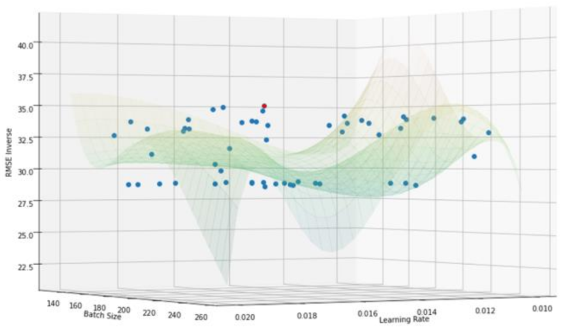
Group 7 Original run and zoomed-in run:

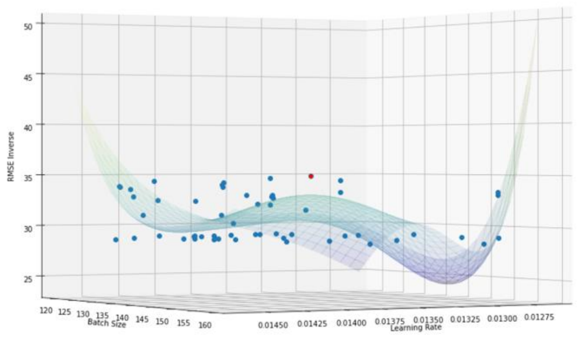




For Group 7, we can see that the optimized point is on an apparent up surface, and we conclude that Bayesian Optimization is well behaved under this case.

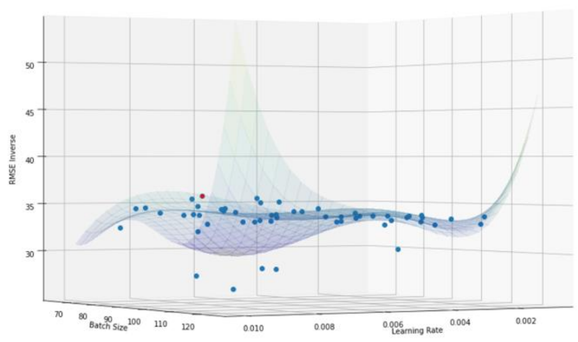
Group 8 Original run and zoomed-in run:

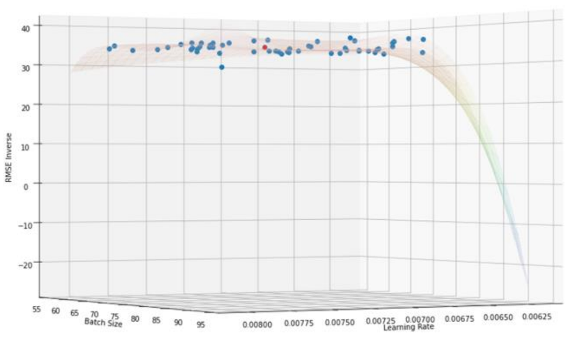




For Group 8, even though the optimized point is not on an apparent up surface, it still achieves a good RMSE inverse estimation.

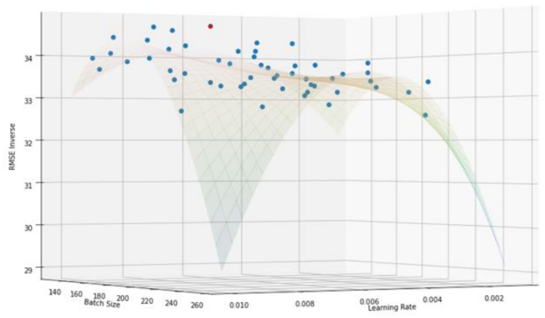
Group 9 Original run and zoomed-in run:

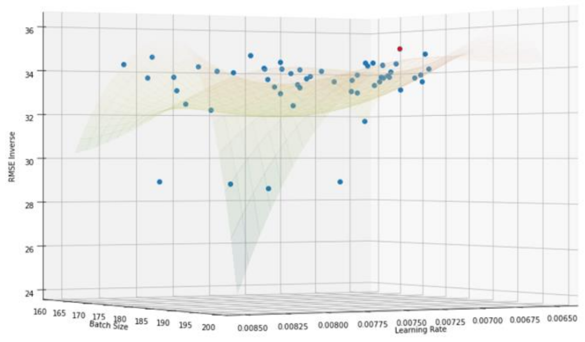




For Group 9, we cannot draw a clear conclusion about the optimality of the Bayesian Optimization algorithm.

Group 10 Original run and zoomed-in run:



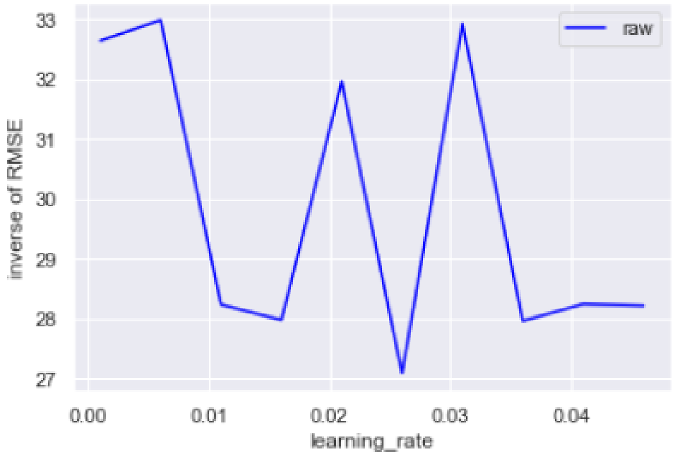


For Group 10, we cannot draw a clear conclusion about the optimality of the Bayesian Optimization algorithm.

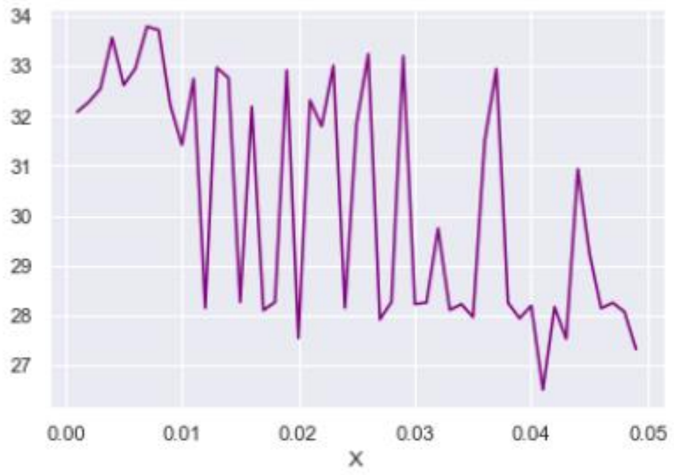
4.2 Robustness Check

When fixing epoch=30, we obtained the maximum of the inverse of the RMSE with batch size=86 and learning rate=0.01234 using Bayesian Optimization.

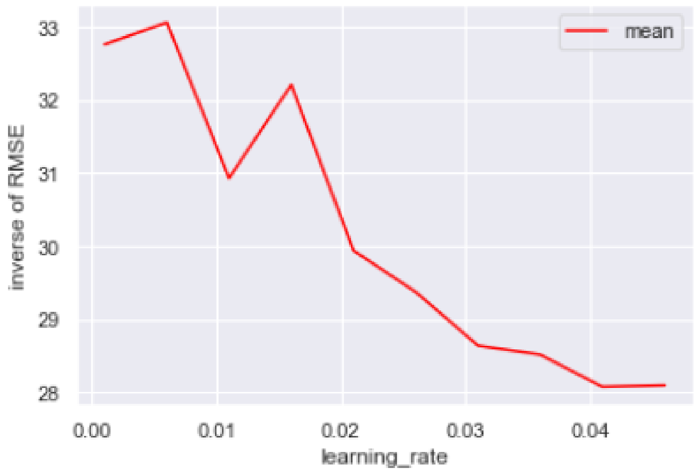
We ran one round of training using each different learning rate and we obtained quite a less robust result since the curve fluctuate up and down.



A closer look at the fluctuation when we reduced the interval and drew a denser plot:



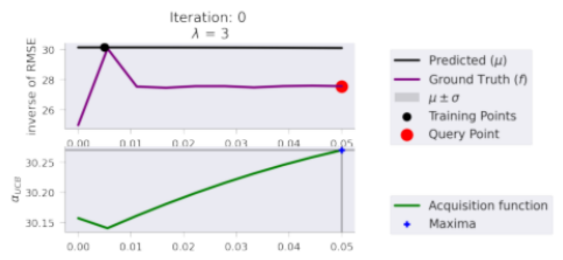
For each learning rate, we calculated its RMSE for each LSTM model 5 times and got the average of the RMSE on each learning rate to reduce the fluctuation.

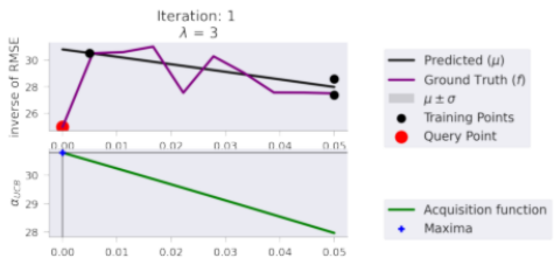


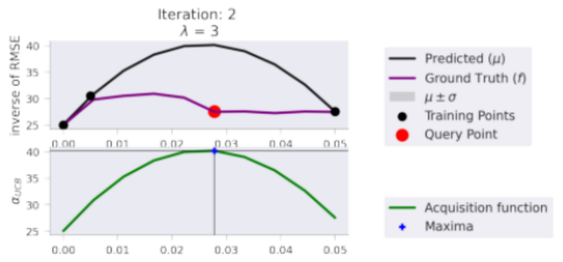
We could see that our result using Bayesian Optimization is robust for the training variable learning rate. Roughly speaking, a smaller learning rate will lead to a higher inverse of RMSE, which integrate closely with the Bayesian Optimization result described above.

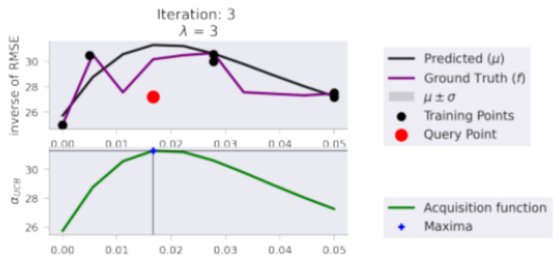
4.3 Acquisition Function’s Impact on Optimality Search Efficiency

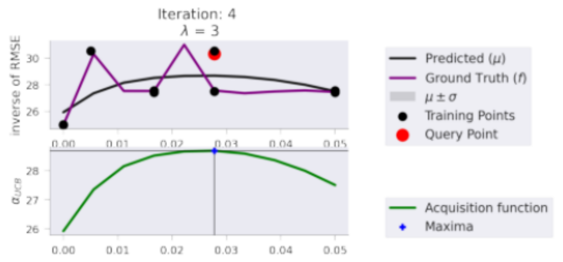
The graph below shows how the acquisition function behaves in each iteration. It is clear to see how Bayesian Optimization works. The next query point is the place where acquisition function achieves its largest value. Trough iteration, Bayesian Optimization gets its largest objective function value.

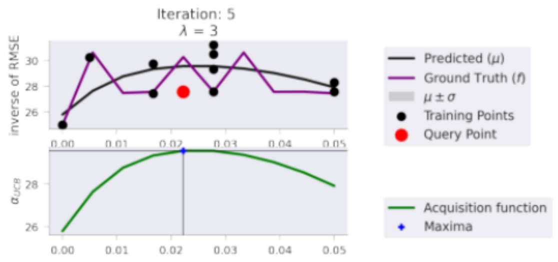


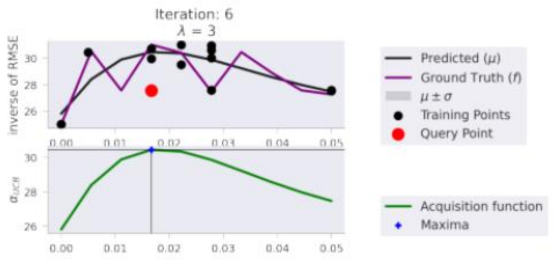


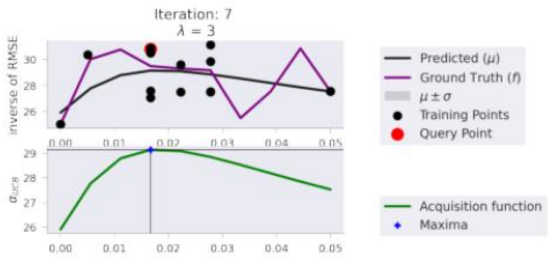


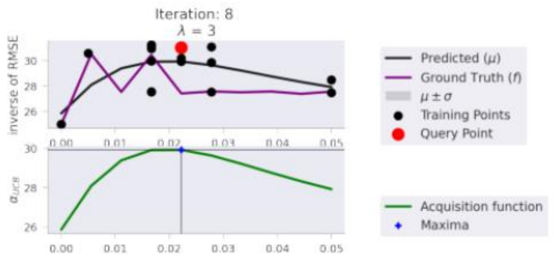


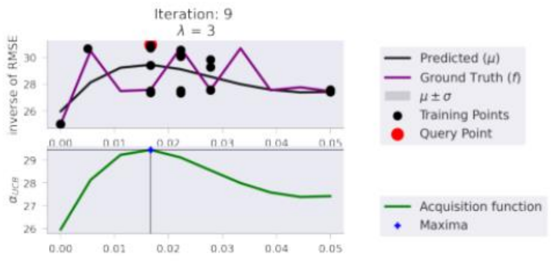




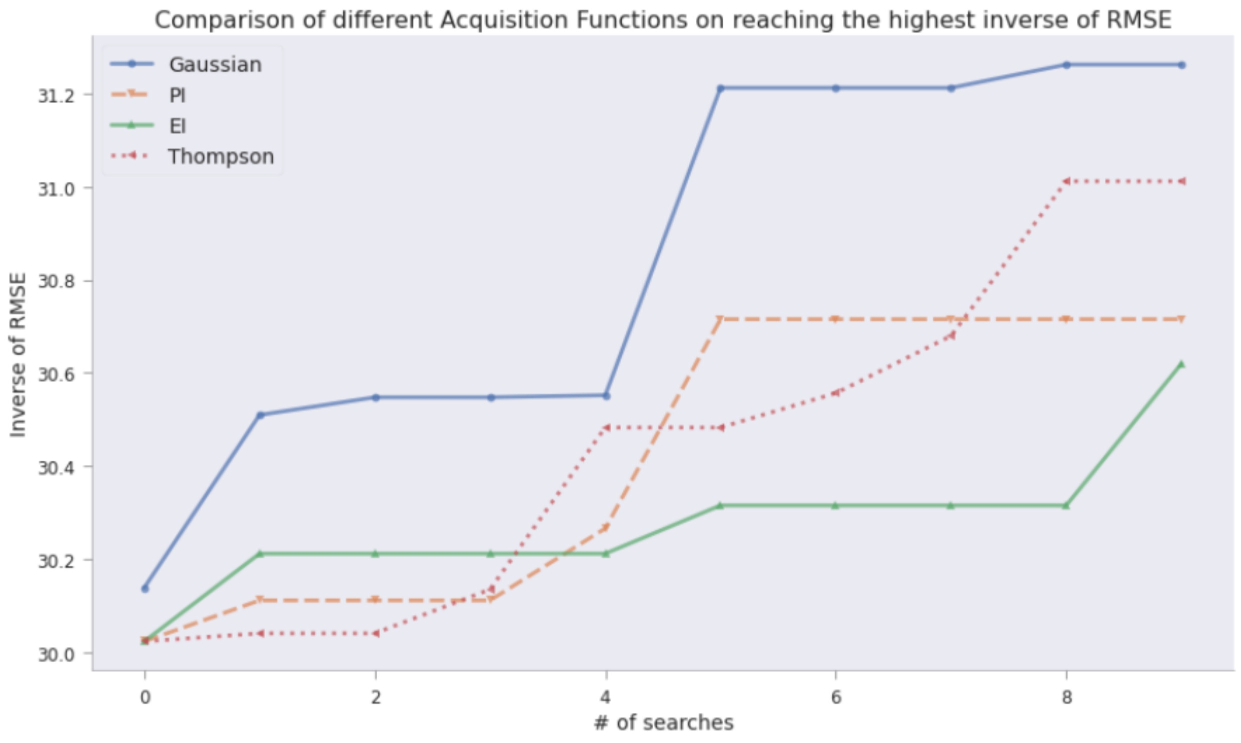








We compared different acquisition functions on reaching the highest inverse of RMSE. We could see that the plots above show the maximum inverse of RMSE reached versus the number of searches on learning rate. Looking at the graph below we could see that for our problem the standard Gaussian Process performed the best among all the variants of acquisition functions.



5 Conclusion

Based on the previous ten group testing for the combination of learning rate and batch size, we conclude that for most cases Bayesian Optimization algorithm is performing well and have the ability to find the optimal hyperparameter set on this LSTM model and this fixed dataset. The zoomed-in plot gives our group a better view and more evidence for the optimality of the returned hyperparameter set. Also, from robustness check, Bayesian Optimization verifies the robustness. From the exploration of embedded acquisition functions, we explored the research on how Bayesian Optimization works and finds its largest value in depth and found that in our case, Gaussian Process works best.

It will be beneficial to extend these kinds of test to more models and datasets to verify the conclusion. We have only illustrated the testing results for such a single case in this report.

6 Future Work

1. Our group mainly tested the combination of learning rate and batch size. Future tests can be conducted to include more possible hyperparameter combinations.

2. It will be helpful if an early stop mechanism can be specified to save computational power.

3. Future groups can research more libraries that implement Bayesian Optimization and compare the difference in between.

4. Other Auto ML methodologies such as DEEP-BO (Diversified, Early-termination-Enabled, and Parallel Bayesian Optimization) can be tested out for hyperparameter selection.

5. To deal with the limitation of computer power, future groups should utilize other more advanced cloud resources to facilitate their model training.

7 References

[1] Frazier, P. I., “A Tutorial on Bayesian Optimization”, 2018.
[2] Taking the Human Out of the Loop: A Review of Bayesian Optimization. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P. and Freitas, N.d., 2016. Proceedings of the IEEE, Vol 104(1), pp. 148-175. DOI: 10.1109/JPROC.2015.2494218
[3] Active learning: theory and applications. Tong, S., 2001.
[4] Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design Srinivas, N., Krause, A., Kakade, S.M. and Seeger, M., 2009. arXiv e-prints, pp. arXiv:0912.3995.
[5] Safe Exploration for Optimization with Gaussian Processes. Sui, Y., Gotovos, A., Burdick, J.W. and Krause, A., 2015. Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, pp. 997--1005. JMLR.org.