CMSC 35360: Assignment 3

Irene Madejski, Bayard Walsh

Spring 2024

Collab Notebook Link:

https://colab.research.google.com/drive/1ai-EmZG4VlBP-eQ_DJMCfFOjLB_66sEK#scrollTo=aMrz14CaUgau

Model Surrogate

Our surrogate model is constructed as a neural network using PyTorch, with three hidden layers and one output layer. This model serves as an approximation of the unknown function that determines profit given certain reaction conditions and is smaller for many quick predictions. As the surrogate model is exposed to more data points from reactor runs, it becomes increasingly precise in predicting outcomes. The acquisition function, explained in the following section, helps determine which to collect and use for further training the model. In running our model, one of the key tradeoffs to consider is the number of points between fine-tuning instances. With more points per batch, we can better fine-tune the model, however that requires more reactor calls.

Acquisition Function

We used an Upper Confidence Bound acquisition (UBC) function to determine the next data collection points by identifying points that maximize the acquisition function. Our UCB acquisition function computes an upper bound for potential points by adding a scaled version of the uncertainty (standard deviation) to the mean given a range of probable parameters and their predicated price per hour, as modeled by the surrogate function. The scaling factor, λ, allows the function to be more explorative in regions with higher uncertainty, while regions with lower uncertainty are exploited.

The UCB acquisition function can be expressed as follows:
$\text{UCB} = \mu(x) + \lambda\sigma(x)$

This acquisition function allows us to easily manage the exploration-exploitation tradeoff in the tunable parameter λ. A larger lamda will lean towards overall more exploration by providing a greater upper confidence bound, while a lower lambda will be more exploitative by keeping points within a smaller upper confidence bound.

Bayesian Optimization Process

We designed a Bayesian optimization process using the surrogate model and acquisition function described above. Before the optimization process begins, 100 reactor runs are completed using a grid search method across the search space. The results from these datapoints help to establish a baseline and determine the starting point for optimization within the larger state space. The parameter for stoicheometry is also held constant at 600 for these initial tests; this decision is based on prior knowledge that maintaining the stoichiometric ratio is preferable to maximize yield.

Then, the optimization process is initialized with the starting parameters established in the previous step, and the reactor is run. The results from this initial run are used to train the surrogate model.

After this initialization phase, an iterative process is used to improve the surrogate model and select new parameters. In each iteration, new reactor conditions are determined using the acquisition function on a range of probable nearby points. The reactor is then run under these new conditions, and the results are collected and used to further fine tune the surrogate model for more accurate predictions. Each set of results is compared to the previous best result; if the new result shows improvement, it becomes the new benchmark.

This cycle is repeated until the time limit is met. Throughout this process, the surrogate model is trained with the new data and adjusted to better reflect the underlying patterns and dependencies discovered through the successive reactor runs.

The last step in the optimization process indentifies the optimal reactor conditions that yield the best possible outcomes under the defined constraints and objectives. This can be compared to the brute force outcome in order to see how our active learning approach can improve the molecule production process using few reactor calls to achieve better price per hour.

Bayesian Optimization Performance

The improvement in profitability over batches due to our Bayesian optimization process is clear when looking at our graphs of profitability over rector runs. Profit consistently increases with more reactor runs as the model becomes more accurate at predicting conditions for maximum profitability.

The benchmark determined by the best Brute Force Performance is a profitability of $2710.458 per hour at reactor call 31169. Our bayesian optimization method performance is compared to this value.

Using our preliminary grid search method with 100 reactor runs, the model also starts with a profitability value of $2622 per hour, which is only $88 per hour less than the brute force best performance. This preliminary grid search method also incorporates the knowledge that maintaining the stoichiometric ratio of 3:2 of the two reactants is prefereable – which reduced the number of parameters explored in the grid search from 5 to 4. This preliminary search helped narrow the parameter space for the bayesian optimization.

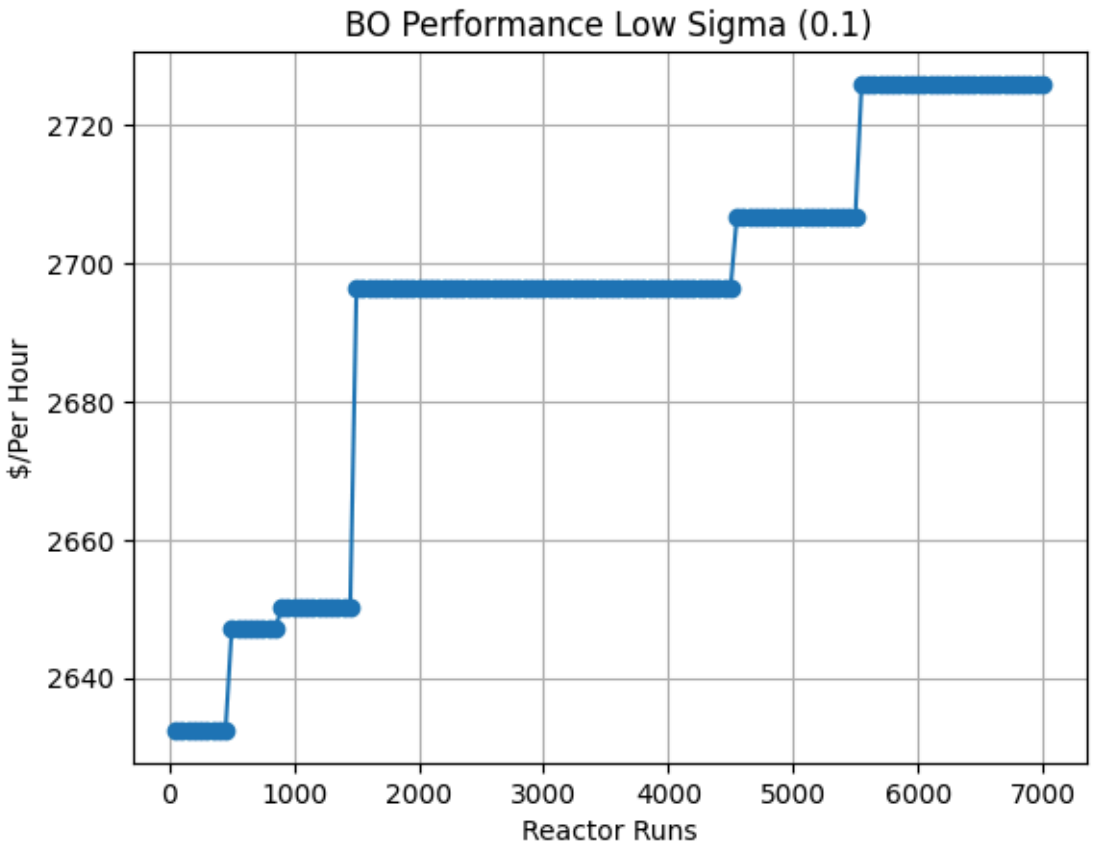
We ran our bayesian optimization process with three different sigma values – varying the exploration/exploitation trade off. For the best performing bayesian optimization process (very large sigma), the model performed very well and quickly achieved an improvement of $212/hour at only roughly 1/13 (8%) of the reactor calls as compared to the brute force method.

All of the other bayesian optimization runs (with different sigmas) also performed better than the brute force model, although by varying amounts. The low sigma bayesian optimization process also performed $10 per hour better than the brute force method after 5600 reactor calls (18% as many as brute force). The medium sigma bayesian optimization process performed $80 per hour better afer 2750 reactor calls (9% as many as the brute force method). The large sigma bayesian optimization performed $210 per hour better after 6450 reactor calls (20% as many as brute force). All of the bayesian optimization performed better than the brute force methods, by varying amounts.

Future quantification methods that would provide valuable information about the performance of the bayesian optimization models would be to calculate total profit over a set amount of time or to calculate the change in profit over time.

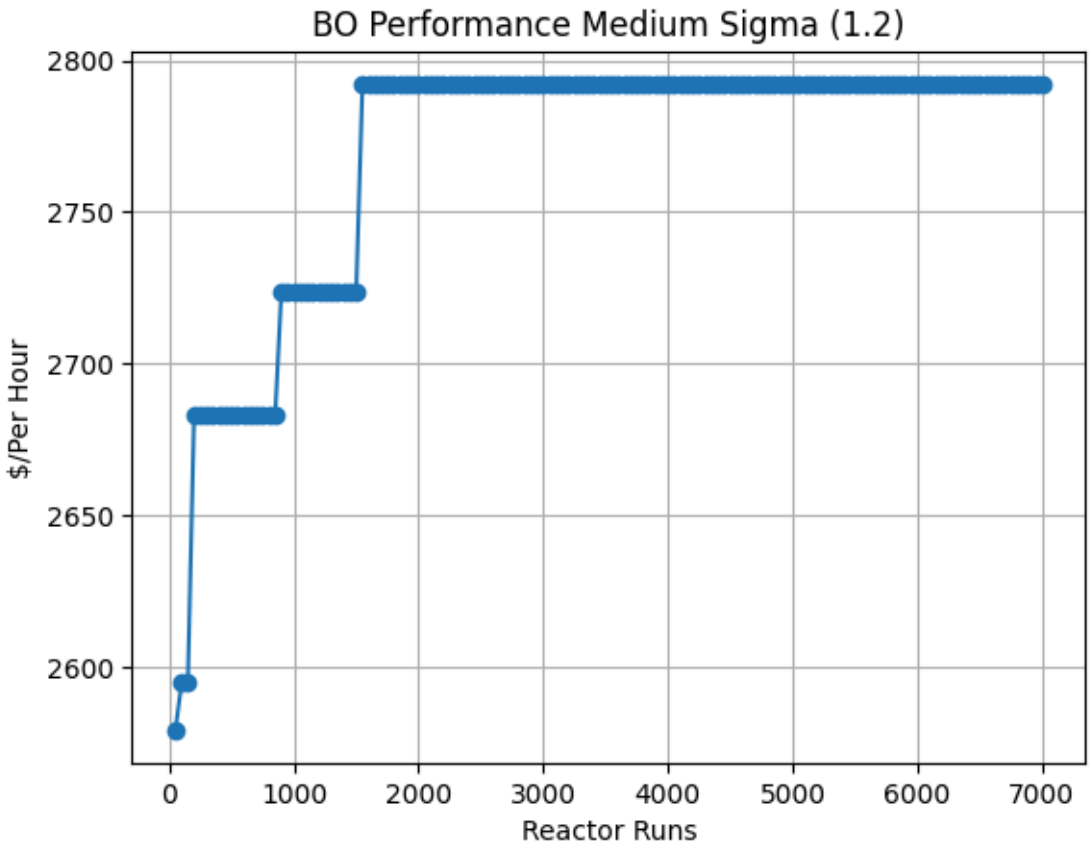
Explore and Exploit Tradeoff

To consider this tradeoff, we varied the sigma value in our UCB caluclations. A larger sigma correlates to a acquisition function that favors exploration, while a smaller sigma favors exploitation. The sigma value impacted the distance of the upper confidence bound from the model-determined mean and influenced which conditions were chosen as next best data points to collect. (Analysis continued after performance graphs)



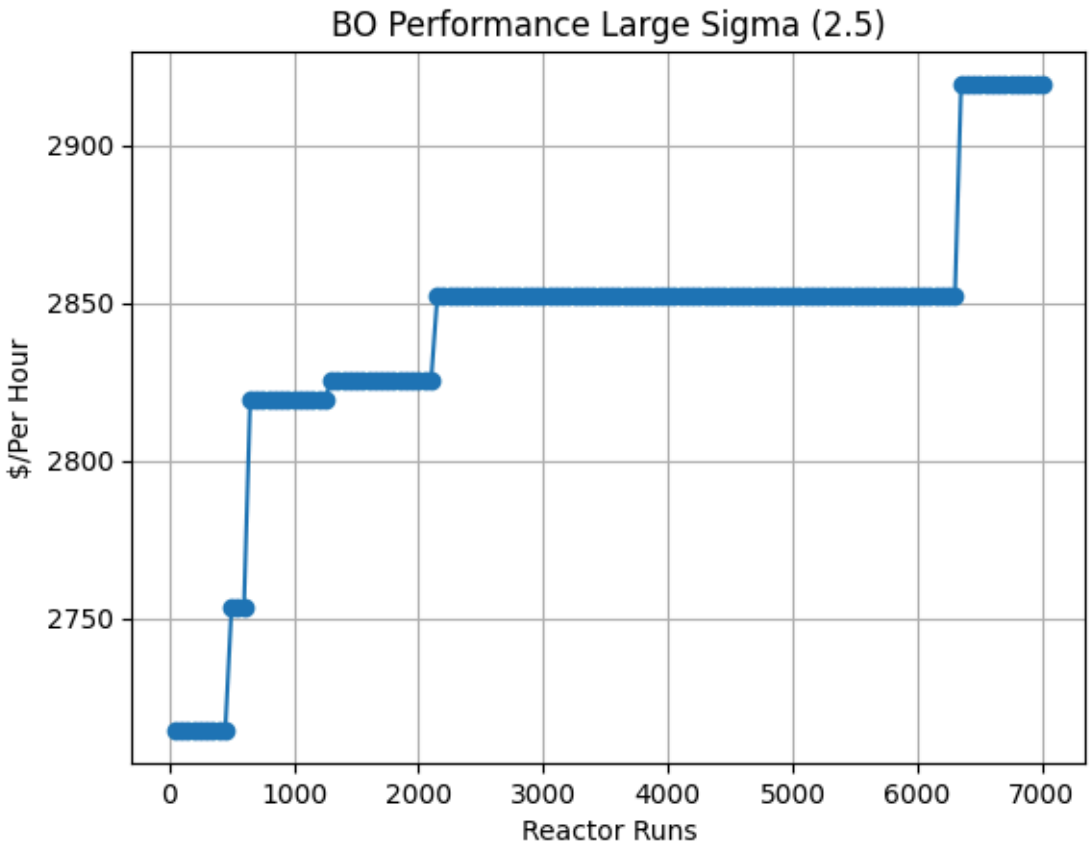
Small Sigma = .1 (reactor calls = 7100)

Best Performance Achieved = 2725.893401698795 at reactor call 5600



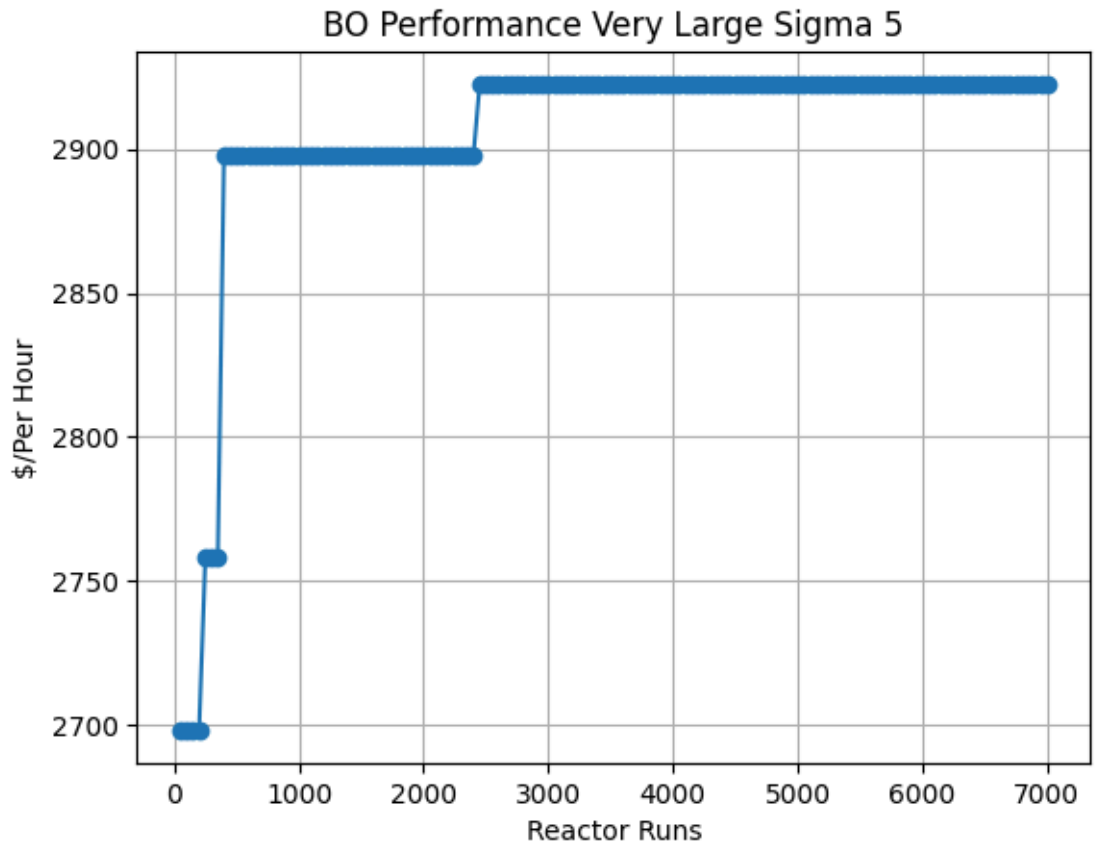
Medium Sigma = 1.2 (reactor calls = 7100)

Best Performance Achieved = 2792.1993665387295 at reactor call 2750



High Sigma = 2.5 (reactor calls = 7100)

Best Performance Achieved = 2919.2853762510003 at reactor call 6450



Very High Sigma = 5 (reactor calls = 7100)

Best Performance Achieved = price: 2922.6043023654142 at reactor call 2550

Explore and Exploit Tradeoff Analysis

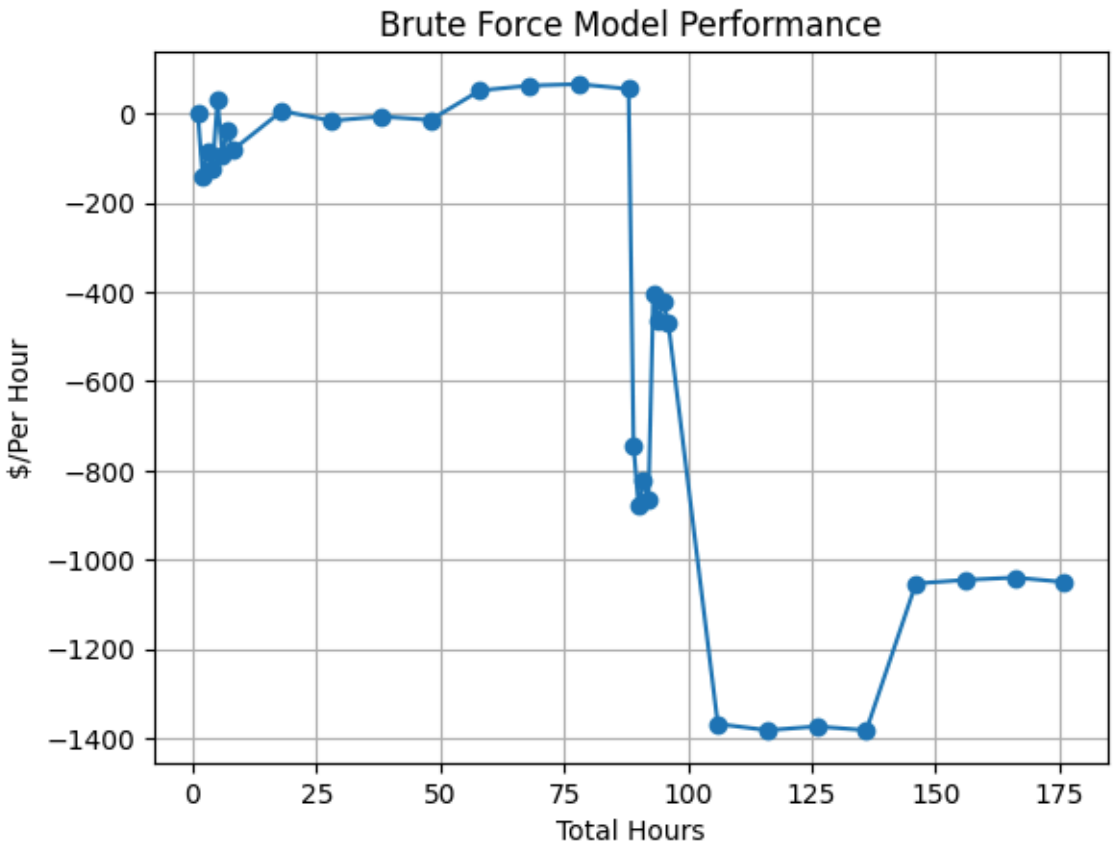
All Sigmas were able to beat the benchmark time, and with less compute. Note that 7100 total reactor calls was selected as model performance generally platued afterwards and it was well witin the brute force range. A larger sigma correlated with better performance across all the models, which is likely due to the scale of the experiment; given the size of the paramter space, having a larger preferance towards exploring new data points in the distribution leads to the potential for new optimal values, and with 7100 tests, there are more than enough instances to exploit data points. Additionally the two smallest sigmas platued in the end of the training instance, indicating that the acquisiton function was not exploring enough for the model to benefit from the training time.

In future work, it would be interesting to model performance increase as a function sigma. More specifically, it would be interesting to note at what maximum sigma value the performance suffers as a result of the UCB being too large.

Additional Exploration

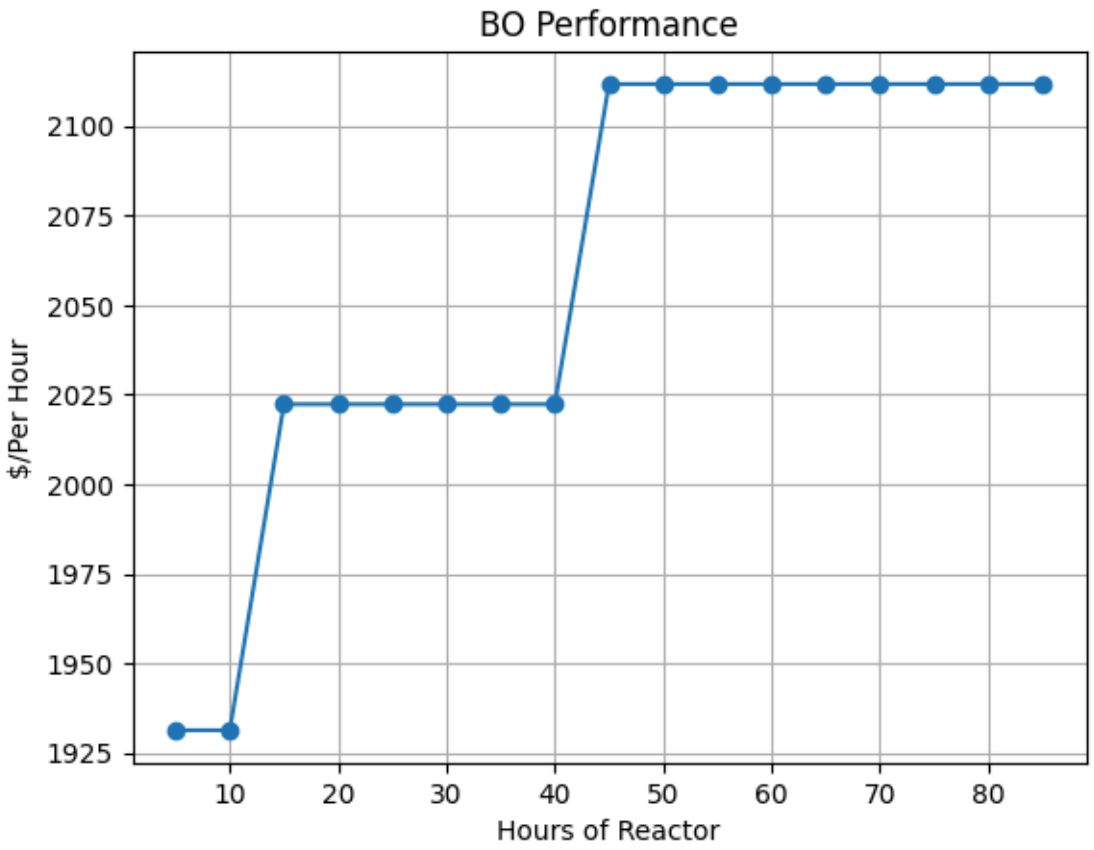
For the additional feature aspect within the model, we opted to focus on the finite time horizion, as we thought this would be an important consideration when transfering a similar model to a lab with constraints.

Total Week Bound On Runtime



Currently brute force uses 176000 hours. We measured reducing to roughly a week runtime (176 hours). Note that a lot of the power in the grid search reduced with cap on time, because if each reaction is within 1-10 hours, running 32000 reactions is no longer possible. To measure brute force performance we reduced the previous script to one that ran for a total of 176 hours, which is roughly a week.

Best Performance Achieved = 66.25587265252894 at hour 78



In comparison, capping the hours at 176 for the Bayesian model caused a slight reduction in performance, however the model far outperformed the brute force approach. With a firmer restriction on time there was even more value for initial conditions (we set the stoich to 600 based on the known variables in the experiment), and we were still able to achieve a strong performance within a 176 hours by calling the surrogate model more often with fewer batches. In part this was done because we knew we had less data to test with, so we increased the amount of model predictions instead of using those data instances to fine-tune the model for later optimization as we did with our longer training runs. Additionally, we reduced the amount of grid search to 88 total hours to get our inital starting point. Our Bayesian Optimization process used the same UBC acquisition function with a sigma of 5, which previously perfomed best by both time and profit in previous testing. Therefore our initial configuration was created by a brief grid search with Bayesian Optimization from the best observed point afterwards. We can see that with a realistic bound on time the value of Bayesian Optimization is more and more clear.

Best Performance Achieved = 2111.681664068077 at hour 136

Reflection

If given more time on the experiment, there were several other features we were considering implimenting. First, we wanted to come up with an efficent way of checking for for duplicate data instances. Because our model will save the parameters from the previous best run for the next training iterations, coming up with some way of removing paramaters that have already been checked would save a lot of wasted reactor runs. Second, varying sigma and the rate of random points for the surrogate model based on the amount of overall runs or platued model behavior could also improve performance. Similar to gradiant descent, we would want to explore more distant spaces initially, and then only explore local spaces as we approach optimization. Additionally, we could come up with a better heuristic for our initalization data and use grid search data to train our

Neural Network better. This is another consideration with using Neural Networks as the surrogate for Bayesian Optimization because if the black box function is too expensive to run enough times to generate sufficient training data for the model, another surrogate model would perform better. Finally, it would be interesting to incorporate more chemical knowledge into the computation process. For instance, the model could be used to calculate the activation energy of the reaction, which would help to choose an appropriate catalyst.