XCS229ii Milestone 3: Experiment Protocol

1. Hypotheses: A statement of the project's core hypothesis or hypotheses.

The core hypothesis of this project is to determine whether a trained reinforcement learning (RL) agent is able to perform better in determining a set of hyperparameters for a given model on an underlying dataset as compared with adaptive sampling or grid based search methods. The RL agent would be trained to tune hyperparameters either of the same model, a subset of the full dataset or both. The baseline would be an off the shelf grid search based hyperparameter tuner (eg: SciKit learn) or sampling based one (eg: Optuna).

2. Data: A description of the dataset(s) that the project will use for evaluation.

The primary dataset we will use for evaluating our core hypothesis is the Yahoo Finance Dataset. This dataset provides daily OHLC (open, high, low, close) quotes for stock market, crypto currencies, and currency exchange. Additionally the API also handles real time low latency quotes on a per minute resolution. Yahoo Finance dataset can also be leveraged to extract financial news, summaries like earnings, balance sheet for various securities, finance and sector index values among other things. We plan to use a historical segment from the dataset to train the RL agent, a different one to tune the hyperparameters using a baseline and trained RL agent and finally the most recent segment to evaluate the performance of the various trained child models.

In addition to the Yahoo datasets, we have identified a few other datasets which we can acquire if additional data are required.

First, TradingView offers second-based stock and crypto data, as opposed to the minute-based sampling rates of Yahoo, thus representing a potential 60-fold increase in our dataset size.

Secondly, there is a measure called the Fear-and-Greed index, which provides a measure of sentiment in the market based on a number of factors, including text analysis (https://alternative.me/crypto/fear-and-greed-index/#api). Data can be downloaded as a CSV, sampled daily.

Finally, there are public stock- and crypto-focused chat groups. Unlike Twitter posts or news articles, these groups are strictly focused on trading and, thus, could provide a better measure of the sentiment of the average investor. This could be useful if we decide to include text-based sentiment analysis as a feature in our child model.

3. Metrics: A description of the metrics that will form the basis for evaluation. We require at least two frameworks from the Machine Learning Theory module in class which includes but is not limited to:
(a) Error Analysis Diagnostics
(b) Ablative Analysis
(c) Approximation and Estimation Errors
(d) Bias-Variance Diagnostic
(e) Optimization Diagnostics
Note: If you are doing a reinforcement learning project the above evaluative metrics will not apply. Instead find a way to illustrate the performance of your policy or policies.

For finance it is very common to use the Sharpe ratio, defined as: the expected return minus the risk free return all divided by the standard deviation. For this work, we will fix the risk free return at 0.1, as an approximate average from recent estimations (source: https://www.treasury.gov/resource-center/data-chart-center/interest-rates/pages/textview.aspx?data=yield)

$$\frac{R_a - R_f}{\sigma_a}$$

We are going to use the sharpe ratio as a reward signal. In RL, showing the average reward or the absolute reward over a period of time is not enough, given the variability of these models, due to, for example, different random seeds to start the parameters. Idding a confidence interval by bootstrapping the data and repeating the process with different random seeds can provide a more robust metric.

If the confidence intervals are too large, it might be necessary to test with more trials. This is because some problems are naturally more stochastic than others and more trials are needed to get more stable confidence interval values.

4. Models: A description of the model(s) that you will use as your baselines and a preliminary description of the model(s) that will be the focus of your investigation. At this early stage, some aspects might not yet be worked out, so preliminary descriptions are sufficient.

The goal of this work is to build a *hyperparameter optimization model*, or what we will call a *hyperparameter model* for short. The purpose of this model is to learn how to tune the hyperparameters of a second model, which we will call a *child model*, so as to maximize the child model's cross-validated performance on a time series dataset. We assume that the *child model* will always be of the same form (e.g., a LSTM), but the characteristics of the underlying dataset may vary.

As an example, consider we have an LSTM and are supplied with two datasets: a small, simple dataset, and a large, complex dataset. Our hyperparameter model should adjust the hyperparameters of our child model so that a simple LSTM is used to fit the small dataset and a complex LSTM is used to fit the large dataset.

To begin, we will first discuss how we will split our data into training and testing pools. Each child model is assumed to be solving a stock trading task. Therefore, it will be trained and tested on a single time series, using the beginning 70% for training and the ending 30% of the time series for cross validation testing the performance of the model.

For the hyperparameter model, we assume that it should be able to fit the child model to any new, unseen dataset. Assuming we have a total of N_{ts} stock time series, the hyperparameter model will use N_{train} time series for training and the remaining N_{test} time series for testing, such that $N_{train}=0.7N_{ts}$ and $N_{test}=0.3N_{ts}$.

We propose two distinct formulations of the reinforcement learning problem. The first formulation we call the “RL-driven grid search.” This approach discretizes the hyperparameter search space into a grid. The RL agent then iteratively selects a location in this grid, which corresponds to a specific hyperparameter configuration of the child model. The child model is trained using this configuration and returns the cross-validation performance metric, which is used to populate the corresponding location in the grid. Therefore, over time, the grid is “filled in” with the performance metrics for the hyperparameter configurations that have been explored, and the RL agent should hone in on the best hyperparameter configuration for the dataset in question.

The RL-driven grid search approach is stated more formally below:

- Setup: We assume that the child model has N_p hyperparameters. This forms a N_p dimensional space, which we carve up into a N_1 x N_2 x N_3, … x N_p grid. Here, N_i is the number of grid slices allotted to the ith hyperparameter.
- State: The state vector, $S \in R^{N1\ x\ N2\ x\ N3,\ \dots\ x\ Np}$, corresponds to a flattened version of the hyperparameter grid. Each entry is populated with the cross validation performance associated with that particular hyperparameter configuration. Initially this grid is completely unexplored. Therefore, we will initialize the entire grid to the cross validation performance of a “naive classifier” (e.g., random guessing), which we presume to be the worst-case scenario.
- Action: An integer between 1 and N_p, denoting the chosen hyperparameter configuration.
- Transition: The transition populates the state vector S at index A with the cross validation loss associated with that particular configuration in the hyperparameter grid. That is, S[A] = P_t , where P_t is the child model’s cross validation performance for the current timestep. We note that this transition is probabilistic, since the cross validation loss is a random variable that depends on the characteristics of the underlying dataset.**
- Reward: We wish to reward the agent for making good guesses that result in improving the performance of the child model. Therefore, we define reward for the current timestep t as R=P_t - max(S). That is, the algorithm is rewarded by how much better the current

guess is compared to the best hyperparameter configuration in our grid. Note that we are assuming here that we wish to maximize our performance metric, rather than minimize it. Note also that Pt - max(s) is equivalent to P_t - max(P_{t-1}, … P_2, P_1, P_{null}), where P_{null} is the performance of our null model.

** For example, increasing the number of nodes in a given layer of a LSTM may result in an increase in performance for one dataset, and a decrease in performance for another.

An alternate formulation for the RL agent would be:

- State $s \in R^{p+1}$ where p are the set of hyperparameters to tune and the last state element is a trinary indicator that states whether the cross validation loss is lower, same or higher ($\{-1, 0, 1\}$) than before.
- Action $a \in R^{p}$ where the action vector indicates whether each of the p hyperparameters should be increased or decreased.
- Reward is the minimum cross validation loss from all the states that have been explored by the agent.

In both of these cases, the RL agent would run for a set number of timesteps and its goal would be to discover the best set of hyperparameters for the underlying child model before it times out.

5. General reasoning: An explanation of how the data and models come together to inform your core hypothesis or hypotheses.

Our core hypothesis is that hyperparameter tuning is a problem that can be tackled effectively by reinforcement learning. If our hypothesis is true, we expect to see our reinforcement learning algorithm identify the optimal hyperparameter configuration for our child model in fewer time steps, on average, than grid search and random guessing. In contrast, if our RL agent takes as many time steps as grid search to find the optimum, then we will fail to prove our hypothesis.

Time permitting, in addition to testing our RL agent against “brute force” optimization methods, we will also see how it compares with sophisticated, purpose-built hyperparameter optimizers like Optuna. Because we do not have the resources for intensive tuning, we do not necessarily expect to outperform these packages, but it will nonetheless provide a useful benchmark.

6. Summary of progress so far: A summary of what you have been doing, what you still need to do, and any obstacles or concerns that might prevent your project from coming to fruition. For all model(s) that have been finished please report their metrics of success.

We have implemented a sandbox system that consists of a child model with its appropriate hyperparameters exposed, the requisite dataset, and a baseline algorithm using the Optuna hyperparameter optimization framework that we will compare our approach against.

The child model, for which the hyperparameters will be tuned, is an implementation of DDPG reinforcement learning agent from the OpenAI StableBaselines3 library. We use the FinRL framework for data preprocessing and setting up the stock trading environment for both training and evaluating the child model.
The current baseline exposes 3 DDPG hyperparameters for tuning: buffer_size, learning_rate and batch_size. The dataset consists of OHCLV (open, high, low, close) quotes for 30 tickers on the Dow Jones Index with the training period between Jan 1st, 2020 to Dec 31st, 2020 and the evaluation data from Jan 1st 2021 through Oct 31st, 2021
The Optuna hyperparameter search discovers the following as baseline

```
{'buffer_size': 10000, 'learning_rate': 0.005414413211338522,
'batch_size': 64}
```

The model performance using the above set of hyperparameters is as follows:

```
begin_total_asset: 1000000.00
end_total_asset: 1113647.31
total_trades: 3012
Sharpe: 0.475
```

7. References: In the same format as Milestone 2.

Sharpe, William F. "The sharpe ratio." *Journal of portfolio management* 21, no. 1 (1994): 49-58.

Henderson, Peter, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters." In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1. 2018.

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).