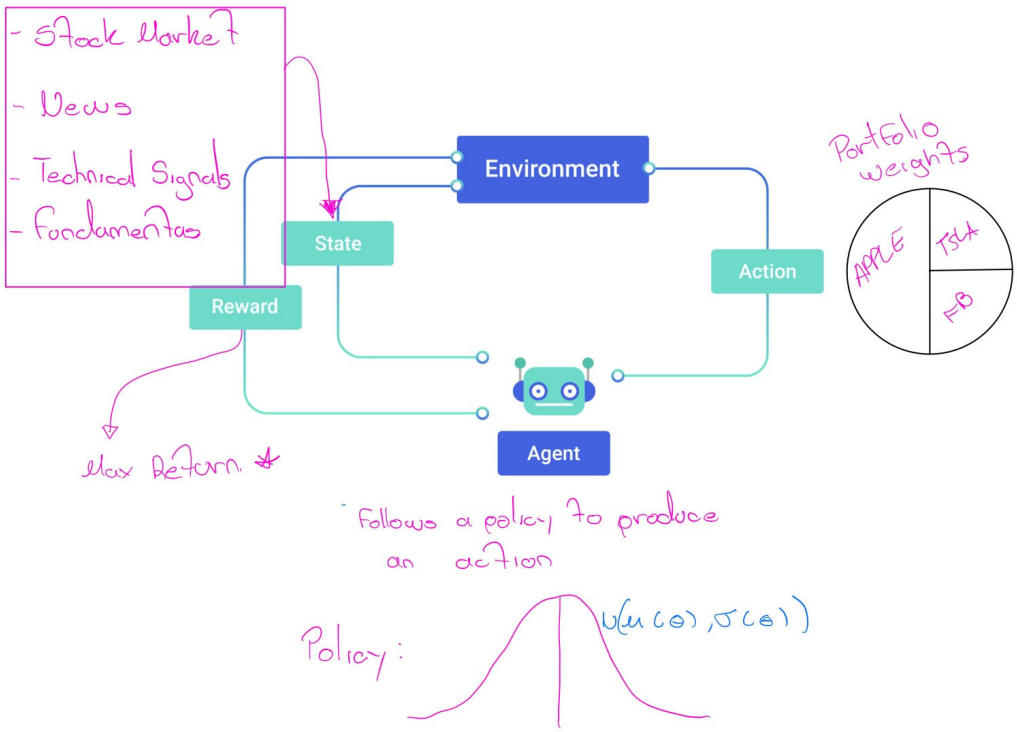
- Stock Market
- News
- Technical Signals
- Fundamentals
Environment
State
Reward
Action
Agent
Portfolio weights
APPLE
TSLA
FB
Max Return *
Follows a policy to produce an action
Policy:
$\mathcal{N}(\mu(\theta), \sigma(\theta))$

Capstone Project Demonstration

In this capstone we explored the usage of Reinforcement Learningfor portfolio construction and enhancement of quantitative investmentstrategies (QIS). Particularly we explored the usage of Policy GradientMethods (PGM) due to their ability to handle continuous action spaces.We used PGM to create a model free agent that selects portfolio weights according to a diverse set of features that we considered as space. The PGM explored are: REINFORCE, REINFORCEwith baseline, Actor-Critic, Actor-Critic with eligibility traces and soft-actor Critic.

Exploring model-free reinforcement leanring algorithms in portfolio allocation that can be generalized to any type of features provides ground work to integrate signal discovery into portfolio allocation.

Stones on the way (Problems we faced)

We will focus particularly on the problem that we faced related to the proposed models.

1. Slow convergence : REINFORCE and REINFORCE with baselines experienced extremely slow convergence in the test set. This make us consider the impracticity of the algorithms for real world solutions where the number of features are complexity of the data require a higher dimensional space.
2. Complex model pipelines: RL implementation requires more complex model pipelines than other machine learning models due to the necesity of creating different assets likes:environment, actors and policies. The interaction of assets in the algorithm creates a complex relation that is not simple to paralelize or transport to other devices. For example; in Soft Actor Critic, the agent has a 4 architectures one model for the policy mean one for the policy variance and 2 for a twin Q function. Each of this model is a Neural Network that needs to be trained in synchrony at each step.
3. Sampling efficiency. As with any reinforcement leanring algorithm, a great amount of time is spent in sampling sars from the environment.
4. Parametrizing the standard deviation on the normal policy did not seem to bring any improvement as we couldnt achieve learning on this parameter.

Model Test and proper model function

For the control dataset, we simulated different assets using a classical geometric Brownian motion process for each of the assets i.e.

$$dS_t = \mu S_t dt + \sqrt{\sigma} S_t dB_t$$

The control data set is built to measure the porformance of each model/algorithm against known solutions given a constant drift and a constant volatility.

We measured each algorithm on a 2-asset simulated data using two different reward windows.

1. Next period return: On each observation the agent gets as reward the return of the portfolio for the next period.
2. Negative of squared return : On each observation the agent gets as reward the negative of the squared return of the portfolio in the next period.

With only two assets we expect that our algorithm will converge to the asset with highest return for the next period return reward and to the asset with the smaller volatility in the second reward.

Test Runs

To run test on simulated assets, user just neeed to define a dictionary with the assets characteristics. and use the class method 'build_environment_from_simulated_assets' from the Environment class. Below we show the run for REINFORCE and ACTOR_CRITIC.

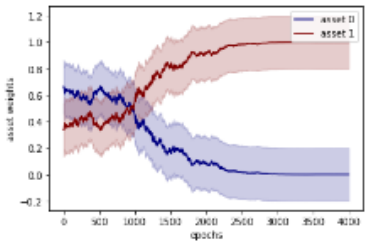

Figure 5: REINFORCE, reward= max return

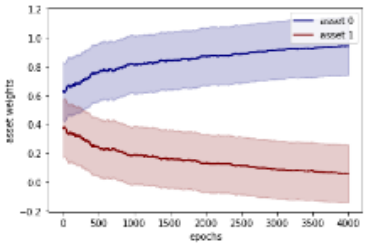

Figure 6: REINFORCE, reward= min variance

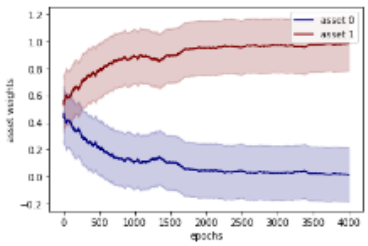

Figure 7: REINFORCE with baseline, reward= max return

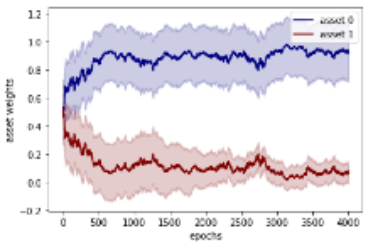

Figure 8: REINFORCE with baseline, reward= min variance

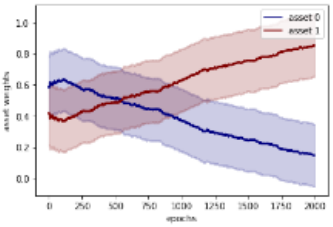

Figure 9: Actor Critic, reward= max return

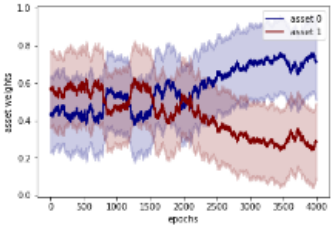

Figure 10: Actor Critic reward= min variance

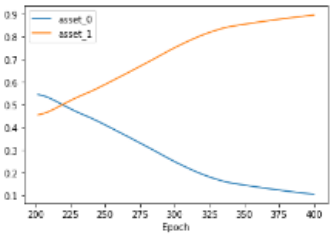

Figure 11: Soft Actor Critic, reward= max return

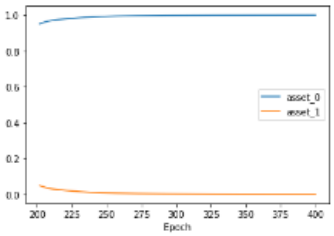

Figure 12: Soft Actor Critic reward= min variance