POLITECNICO
MILANO 1863

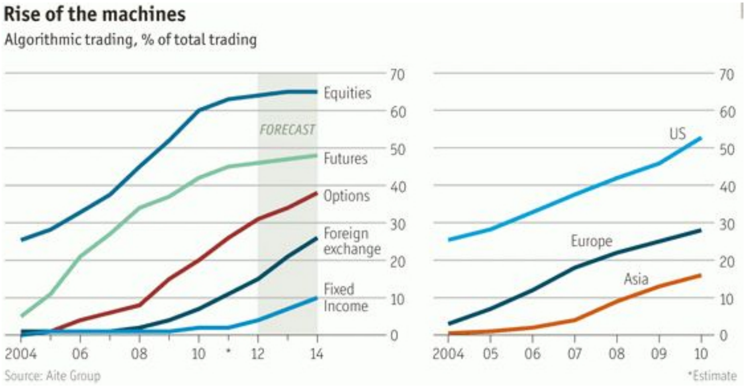
Policy Gradient Algorithms for the Asset Allocation Problem

Pierpaolo Necchi
pierpaolo.necchi@gmail.com

December 10, 2016

Animal Spirits

The Sound of Silence


Rise of the machines
Algorithmic trading, % of total trading
70
60
50
40
30
20
10
0
FORECAST
Equities
Futures
Options
Foreign exchange
Fixed Income
2004
06
08
10
*
12
14
US
Europe
Asia
05
07
09
Source: Aite Group
*Estimate

Plan

1. Basics of Reinforcement Learning

2. Policy Gradient Algorithms

3. Asset Allocation with Transaction Costs

4. Conclusions

Plan

1. Basics of Reinforcement Learning

2. Policy Gradient Algorithms

3. Asset Allocation with Transaction Costs

4. Conclusions

The Reinforcement Learning Framework

Environment

$\mathcal{P}, \mathcal{R}$

The Reinforcement Learning Framework

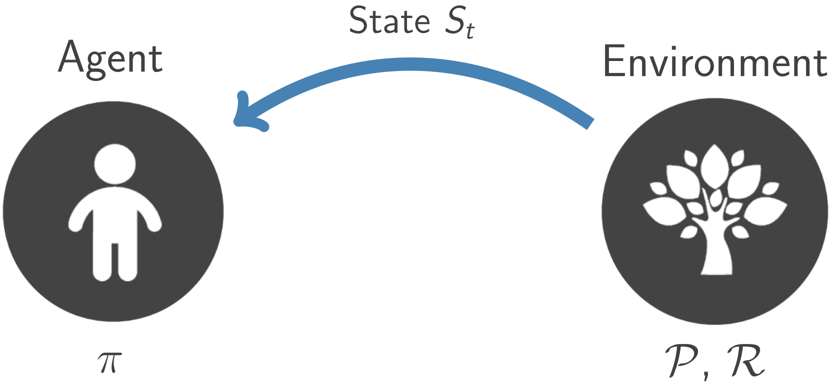
Agent

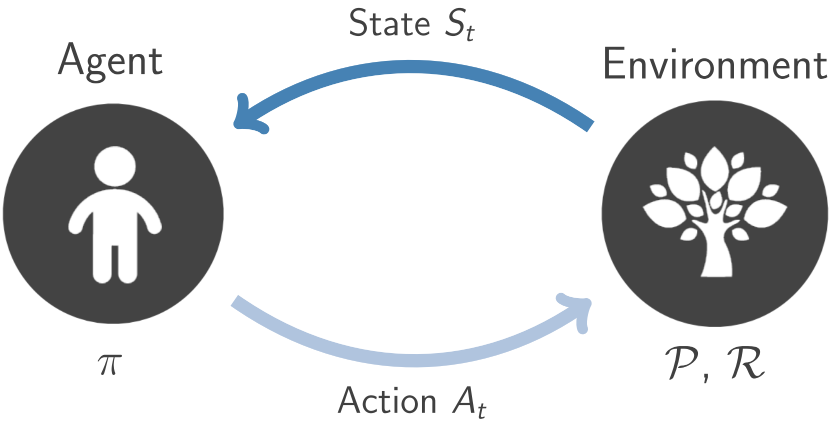
π

Environment

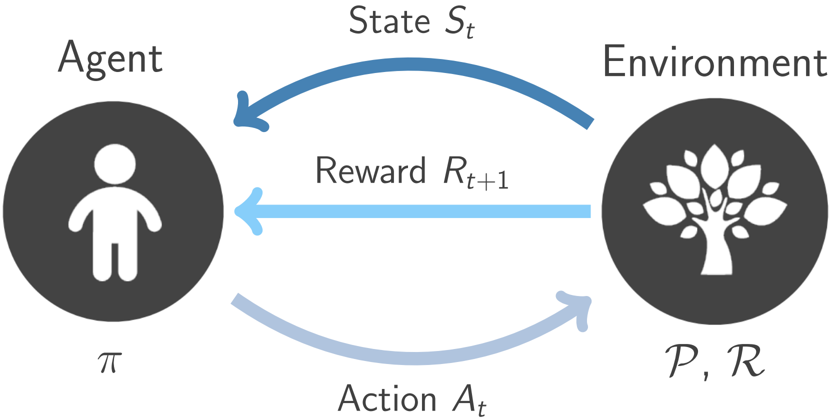
$\mathcal{P}, \mathcal{R}$

The Reinforcement Learning Framework




State S_t
Agent
Environment
π
$\mathcal{P}, \mathcal{R}$
Action A_t

The Reinforcement Learning Framework



Mathematical Formulation

State-value function

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s \right]$$

Mathematical Formulation

State-value function

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s\right]$$

Action-value function

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s, A_0 = a\right]$$

Mathematical Formulation

State-value function

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, S_0 = s\right]$$

Action-value function

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, S_0 = s, A_0 = a\right]$$

Optimal value functions

$$V_*(s) = \max_\pi V_\pi(s) \qquad Q_*(s, a) = \max_\pi Q_\pi(s, a)$$

Mathematical Formulation

State-value function

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\,\middle|\, S_0 = s\right]$$

Action-value function

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\,\middle|\, S_0 = s, A_0 = a\right]$$

Optimal value functions

$$V_*(s) = \max_\pi V_\pi(s) \qquad Q_*(s, a) = \max_\pi Q_\pi(s, a)$$

Optimal policy

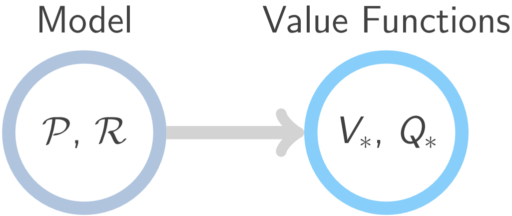
$$\pi_* \text{ s.t. } V_{\pi_*}(s) = V_*(s)$$

Taxonomy of RL Algorithms

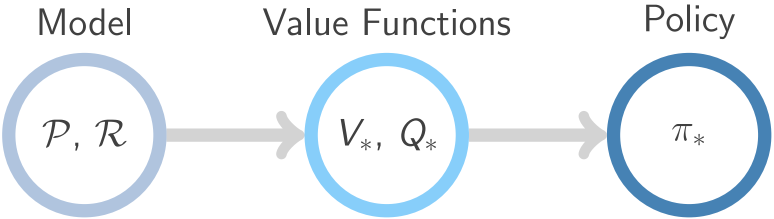
Model

$\mathcal{P}, \mathcal{R}$

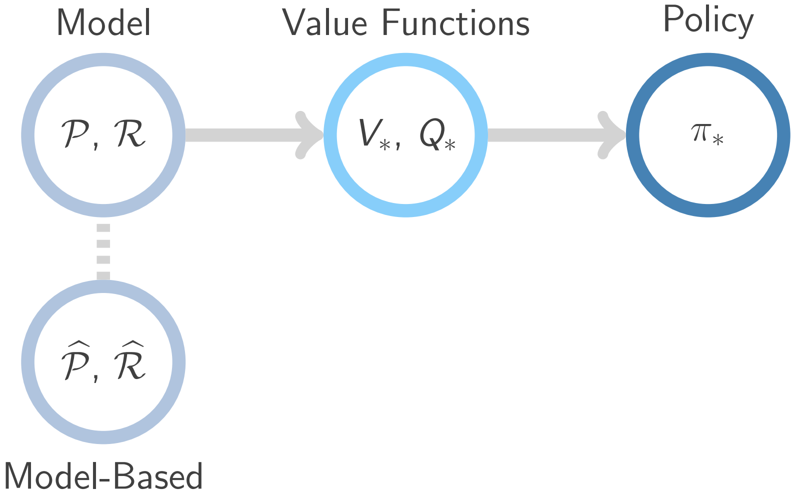
Taxonomy of RL Algorithms



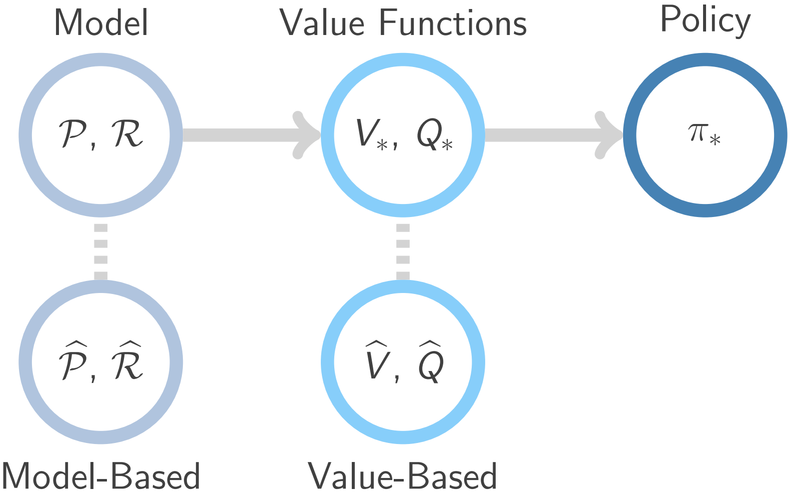
Taxonomy of RL Algorithms



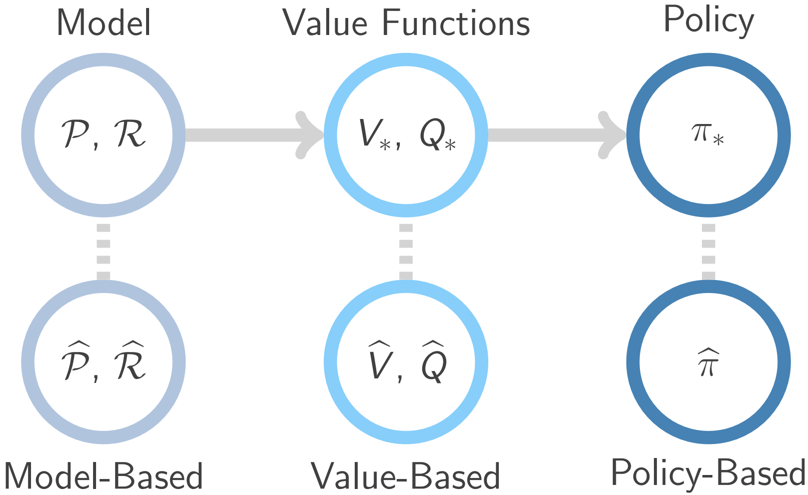
Taxonomy of RL Algorithms



Taxonomy of RL Algorithms



Taxonomy of RL Algorithms



Plan

1. Basics of Reinforcement Learning
2. Policy Gradient Algorithms
3. Asset Allocation with Transaction Costs
4. Conclusions

Key Idea

1. The optimal policy π_* is approximated with a parametric policy π_{θ^*}

Policy Gradient Algorithms

Key Idea

1. The optimal policy π_* is approximated with a parametric policy π_{θ^*}
2. The parameters θ^* of the policy solve the optimization problem

$$\max_{\theta \in \Theta} J(\theta) = V_{\pi_\theta}(s)$$

Policy Gradient Algorithms

Key Idea

1. The optimal policy π_* is approximated with a parametric policy π_{θ^*}
2. The parameters θ^* of the policy solve the optimization problem

$$\max_{\theta \in \Theta} J(\theta) = V_{\pi_\theta}(s)$$

3. Which is solved numerically via gradient descent

$$\theta_{k+1} = \theta_k + \alpha_k + \ \nabla_\theta J\left(\theta_k\right)$$

Policy Gradient Algorithms

Key Idea

1. The optimal policy π_* is approximated with a parametric policy π_{θ^*}
2. The parameters θ^* of the policy solve the optimization problem

$$\max_{\theta \in \Theta} J(\theta) = V_{\pi_\theta}(s)$$

3. Which is solved numerically via gradient descent

$$\theta_{k+1} = \theta_k + \alpha_k + \nabla_\theta J(\theta_k)$$

?

The Keystone of Policy Gradient Algorithms

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{S \sim d^\theta \\ A \sim \pi_\theta}} \left[\nabla_\theta \log \pi_\theta(S, A) Q_\theta(S, A) \right]$$

The Keystone of Policy Gradient Algorithms

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{S\sim d^\theta \\ A\sim\pi_\theta}} \left[\nabla_\theta \log \pi_\theta(S, A) Q_\theta(S, A)\right]$$

For an episodic environment, the policy gradient can be approximated via Monte Carlo

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log \pi_\theta(s_0, a_0) Q_\theta(S, A)$$

The Keystone of Policy Gradient Algorithms

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{S \sim d^\theta \\ A \sim \pi_\theta}} \left[\nabla_\theta \log \pi_\theta(S, A) Q_\theta(S, A) \right]$$

For an episodic environment, the policy gradient can be approximated via Monte Carlo

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log \pi_\theta(s_0, a_0) Q_\theta(S, A)$$

However, this estimate is characterized by a large variance. Possible improvements:

1. Optimal baseline
2. Actor-critic methods
3. Natural gradient

Policy Gradient with Parameter-Based Exploration (PGPE)

Key Idea

1. Actions are selected using a parametric and deterministic controller F_θ

Policy Gradient with Parameter-Based Exploration (PGPE)

Key Idea

1. Actions are selected using a parametric and deterministic controller F_θ
2. The policy parameters are drawn from a probability distribution p_ξ

Policy Gradient with Parameter-Based Exploration (PGPE)

Key Idea

1. Actions are selected using a parametric and deterministic controller F_θ
2. The policy parameters are drawn from a probability distribution p_ξ
3. The search for an optimum is performed in the space of the hyperparameters ξ

Policy Gradient with Parameter-Based Exploration (PGPE)

Key Idea

1. Actions are selected using a parametric and deterministic controller F_θ
2. The policy parameters are drawn from a probability distribution p_ξ
3. The search for an optimum is performed in the space of the hyperparameters ξ

More formally, the update scheme is

$$\xi_{k+1} = \xi_k + \alpha_k \nabla_\xi J(\xi_k)$$

where the gradient is given by

$$\nabla_\xi J(\xi) = \mathbb{E}_{\substack{S \sim d^\xi \\ \theta \sim p_\xi}} \left[\nabla_\xi \log p_\xi(\theta) Q_\xi \left(S, F_\theta(S)\right)\right]$$

Plan

Problem Formulation

Investor's Goal

How to dynamically invest the available capital in a portfolio of different assets in order to maximize the expected total return or another relevant performance measure.

Problem Formulation

Investor's Goal

How to dynamically invest the available capital in a portfolio of different assets in order to maximize the expected total return or another relevant performance measure.

Rewards: portfolio log-return with transaction costs

$$R_{t+1} = \log\left\{1 + \sum_{i=0}^{I}\left[a_t^i X_{t+1}^i - \delta_i\left|a_t^i - \tilde{a}_t^i\right| - \delta_s\left(a_t^i\right)^-\right] - \delta_f \mathbf{1}_{a_t \neq \tilde{a}_{t-1}}\right\}$$

Problem Formulation

Investor's Goal

How to dynamically invest the available capital in a portfolio of different assets in order to maximize the expected total return or another relevant performance measure.

Rewards: portfolio log-return with transaction costs

$$R_{t+1} = \log \left\{ 1 + \sum_{i=0}^{I} \left[a_t^i X_{t+1}^i - \delta_i \left| a_t^i - \tilde{a}_t^i \right| - \delta_s (a_t^i)^- \right] - \delta_f \mathbf{1}_{a_t \neq \tilde{a}_{t-1}} \right\}$$

Actions: Portfolio weights

$$\{a_t^i\}_{i=0}^{I} \quad \text{s.t.} \quad \sum_{i=0}^{I} a_t^i = 1 \qquad \forall t \in \{0, 1, 2, \ldots\}$$

Problem Formulation

Investor's Goal

How to dynamically invest the available capital in a portfolio of different assets in order to maximize the expected total return or another relevant performance measure.

Rewards: portfolio log-return with transaction costs

$$R_{t+1} = \log \left\{ 1 + \sum_{i=0}^{I} \left[a_t^i X_{t+1}^i - \delta_i \left| a_t^i - \tilde{a}_t^i \right| - \delta_s (a_t^i)^- \right] - \delta_f \mathbf{1}_{a_t \neq \tilde{a}_{t-1}} \right\}$$

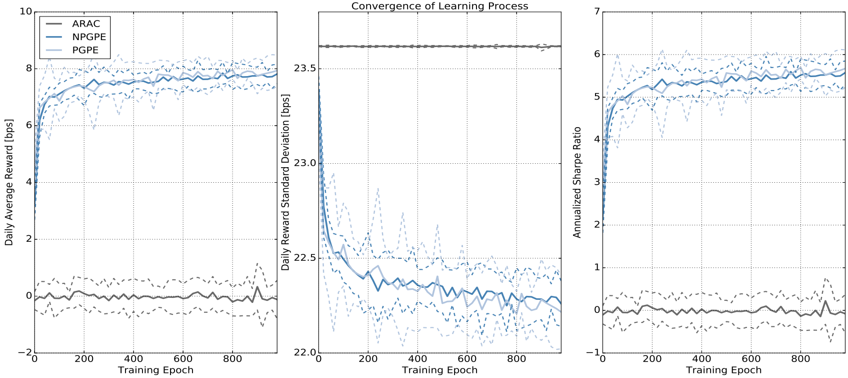
Actions: Portfolio weights

$$\{a_t^i\}_{i=0}^{I} \quad \text{s.t.} \quad \sum_{i=0}^{I} a_t^i = 1 \quad \forall t \in \{0, 1, 2, \dots\}$$

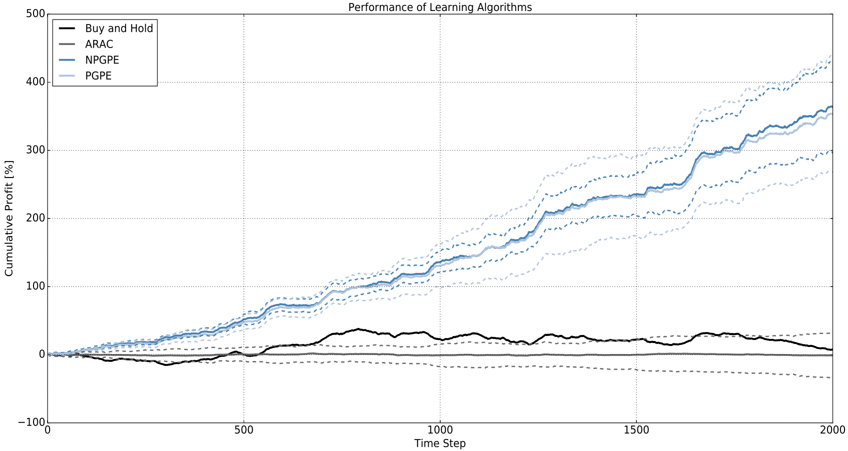
States: assets past returns and current allocation

$$S_t = \{X, X_t, X_{t-1}, \dots, X_{t-P}, \tilde{a}_t\}$$

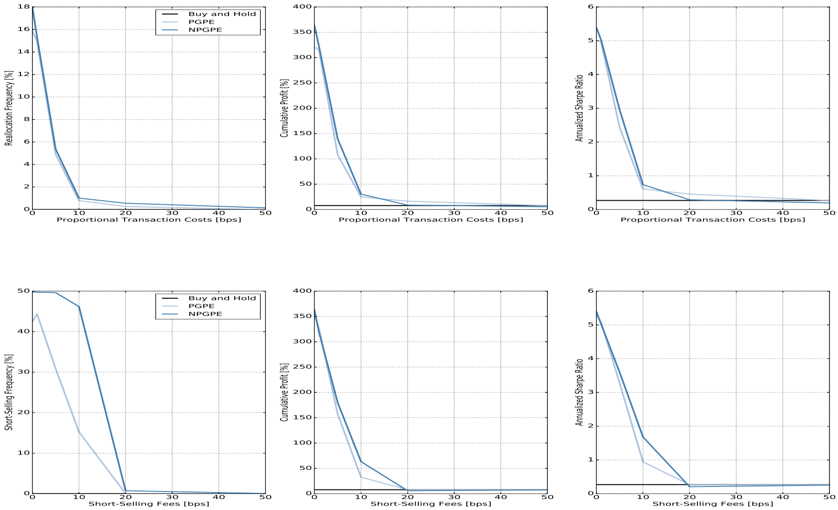
Synthetic Asset: Convergence



Synthetic Asset: Backtest Performance



Synthetic Asset: Impact of Transaction Costs



The Challenge of Historical Data

Plan

Conclusions

What Has Been Done

1. Applied state-of-the-art RL algos to find a profitable long-short trading strategy
2. RL strategies outperform the simple B&H for a synthetic asset
3. RL strategies are able to adapt to transaction costs
4. RL seems suitable to deal with many financial decision problems

Research Directions

1. Improve the algos performance on historical data
2. Developing more complex features for the trading strategy
3. Apply RL techniques to other financial problems

Thank you for your attention!

References

Joshi, M. S. (2008).
C++ design patterns and derivatives pricing, volume 2.
Cambridge University Press.

Moody, J., Wu, L., Liao, Y., and Saffell, M. (1998).
Performance functions and reinforcement learning for trading systems and portfolios.
Journal of Forecasting, 17:441–470.

Peters, J. and Schaal, S. (2008).
Reinforcement learning of motor skills with policy gradients.
Neural networks, 21(4):682–697.

Sutton, R. S. and Barto, A. G. (1998).
Introduction to reinforcement learning, volume 135.
MIT Press Cambridge.

Markov Decision Processes

Reinforcement Learning

General class of algorithms that allow an agent to learn how to behave in a stochastic and possibly unknown environment by trial-and-error.

Markov Decision Process (MDP)

stochastic dynamical system specified by $< \mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma >$

1. $(\mathbb{S}, \mathcal{S})$ is a measurable state space
2. $(\mathbb{A}, \mathcal{A})$ is a measurable action space
3. $\mathcal{P} : \mathbb{S} \times \mathbb{A} \times \mathcal{S} \to \mathbb{R}$ is a Markov transition kernel
4. $\mathcal{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is a reward function
5. $0 < \gamma < 1$ is the discount factor.

Experiment on Synthetic Asset

The synthetic asset price is given by

$$Z_t = \exp\left(\frac{z_t}{\max_t z_t - \min_t z_t}\right)$$

where $\{z_t\}$ is a random walk with autoregressive trend $\{\beta_t\}$

$$\begin{aligned} z_t &= z_{t-1} + \beta_{t-1} + \kappa\epsilon_t \\ \beta_t &= \alpha\beta_{t-1} + \nu_t \end{aligned}$$

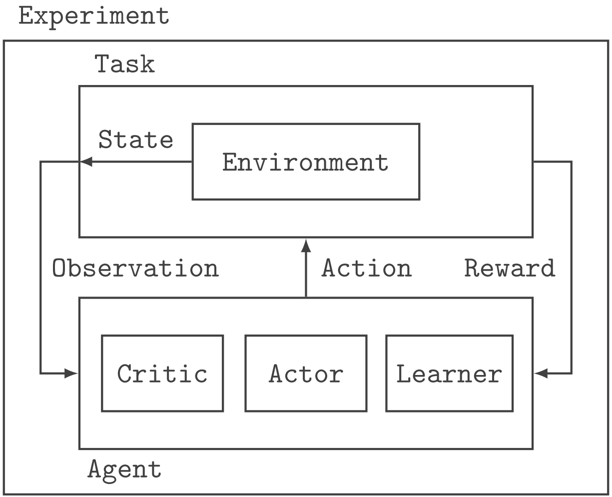
The policy used for the PGPE and the NPGPE algorithms is

$$F_\theta(s) = \text{sign}(\theta \cdot s)$$

where

$$\theta \sim \mathcal{N}(\mu, \text{diag}(\sigma))$$

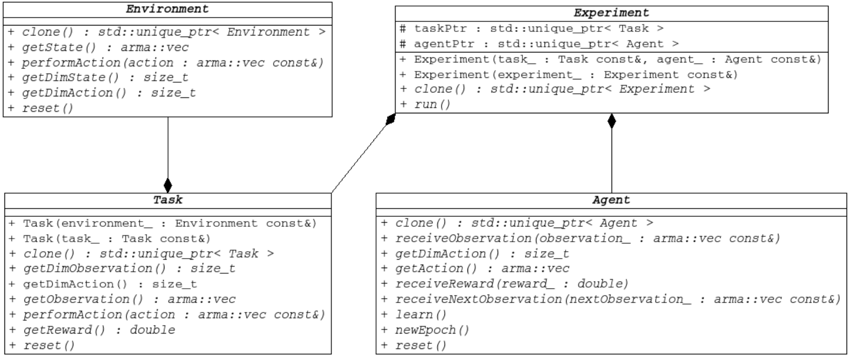
PyBrain's Architecture for a RL Problem



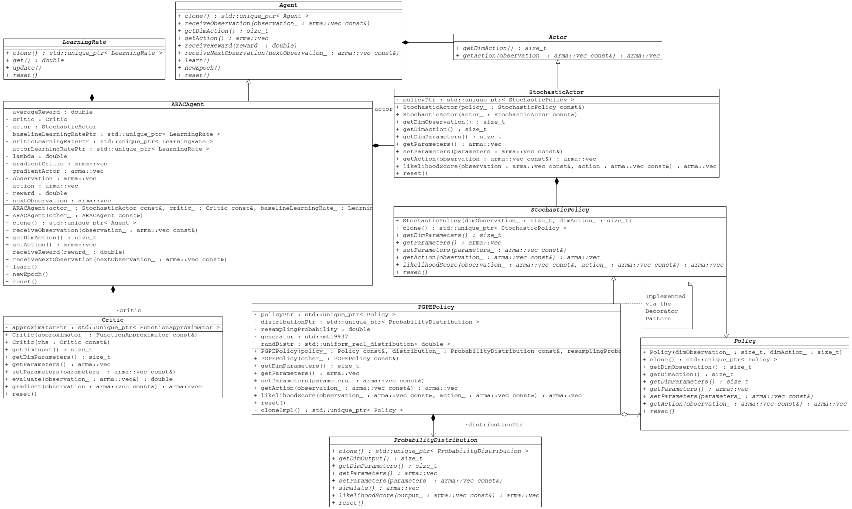
Agent-Environment Interaction in C++

Adapting PyBrain's Architecture

1. Defined standard interfaces via pure abstract classes
2. Achieved modularity via polymorphic composition



Agent's Architecture in C++



Execution Pipeline

experiment_launcher.py

1. Program execution is handled by a Python script
2. Responsible for analyzing the output of the C++ engine