Reinforcement Learning in Finance

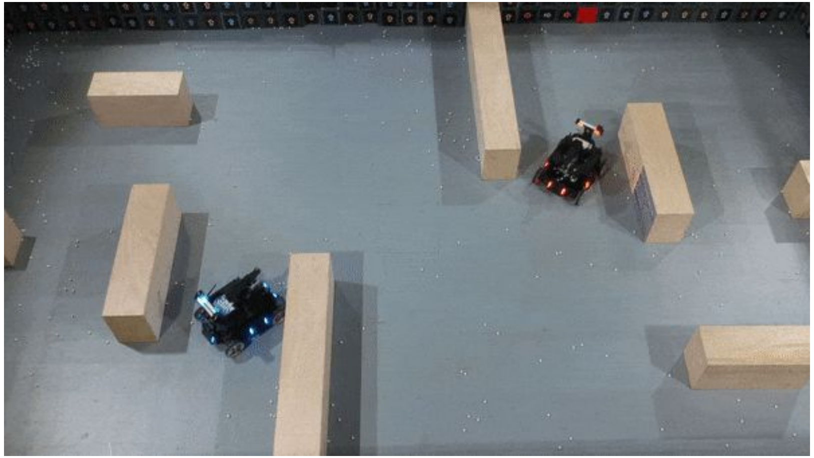
Presentation outline

Part 1: Introduction to Reinforcement Learning

Part 2: Practical use-case of RL in Finance

A bit of background..

AXYON.AI



[Link](#)

RoboMaster AI Challenge

HOST

HOST

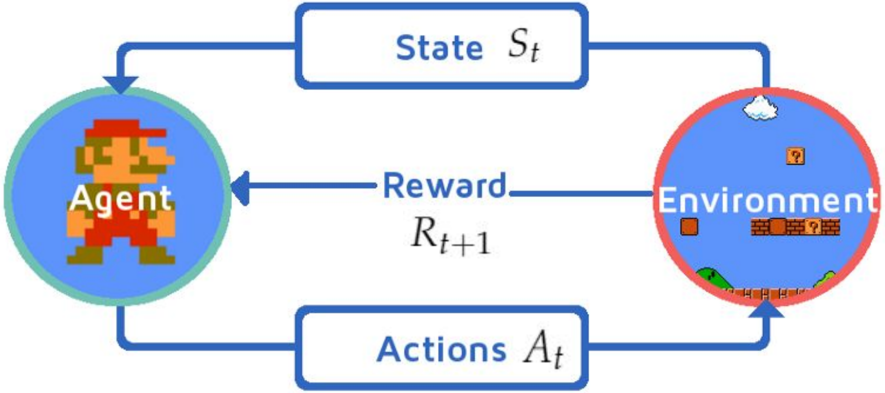
ORGANIZER

Reinforcement Learning Introduction

AXYON.AI

Introduction

AXYON.AI



GOAL: Learn how to take actions in order to maximize reward

SPOILER: How to solve the problem?

Approach 1:

Estimate the **value** of being in a given **state**

Value-based Methods

Approach 2:

Learn a **policy** that maps **states** to **actions**

Policy-based Methods

Mixed approach:

Actor-Critic Methods

More applications here

AXYON.AI

Some mathematical definitions

Cumulative reward

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad = R_{t+1} + \gamma G_{t+1}$$

γ Discount rate

Value function

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$
$$= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$$
for all $s \in \mathcal{S}$

Policy

$$\pi(s,a) = \mathbb{P}[a|s]$$

Stochastic

Deterministic

Optimal value function

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

PROBLEMS:

- Traditional methods, based on dynamic programming require perfect knowledge of transition probabilities
- State and action spaces may be too large

Deep Reinforcement Learning

AXYON.AI

SOLUTION: Approximate policy and value function with neural networks

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s) \longrightarrow J(\mathbf{w})$$

$$= \mathbb{E}_\pi \left[(v_\pi(S) - \hat{v}(S, \mathbf{w}))^2 \right]$$

$$\nabla_{\mathbf{w}} J(\mathbf{w})$$

Gradient descent

$$\pi_\theta(s, a) = \mathbb{P}[a | s, \theta] \longrightarrow J_{avR}(\theta)$$

= Function of the average Reward

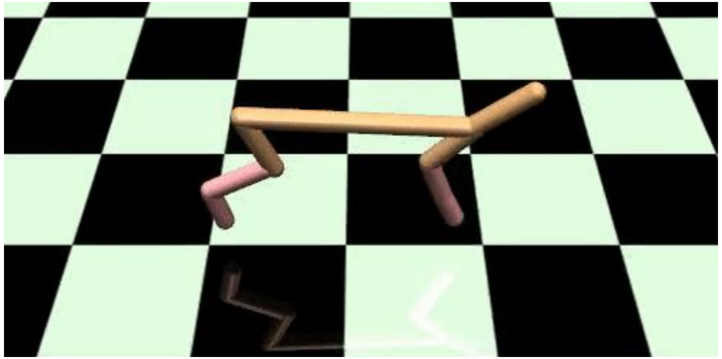
$$\nabla_\theta J(\theta)$$

Gradient ascent

AXYON.AI

WARNING: Current Limitations of RL

- Transferring to real world
- Sample inefficiency
- Reward function shaping



RL use-cases in Finance

- ➔ Option Pricing [1]
- ➔ Order-book execution [2][3]
- ➔ FX Trading [4]

[1] Reinforcement Learning Applied to Option Pricing K.S. Martin
[2] Machine Learning for Market Microstructure and High Frequency Trading ∗ Michael Kearns† Yuriy Nevmyvaka
[3] "Active Learning in Trading Algorithms" by David Fellah, Head of the EMEA Linear Quant Research Group at J.P. Morgan
[4] https://www.jpmorgan.com/global/markets/machine-learning-fx

RL Application:
Portfolio Management

Problem Background

AXYON.AI

"Portfolio management is the process of selecting and investing in a mix of different financial products, with the goal of maximizing the long-term value of the portfolio while constraining the risk to an acceptable level."

Traditional quantitative methods:

- Markovitz Model
- Black & Litterman
- Factor Models

RL-Driven Portfolio

AXYON.AI

GOAL: Develop an RL agent that optimally allocates a portfolio between a set of assets (mix of equity indexes)

Starting project: PGPortfolio (allocation of a portfolio of cryptocurrencies [1])

Why use RL?

→ Fully capture data dynamics, end-to-end automated process

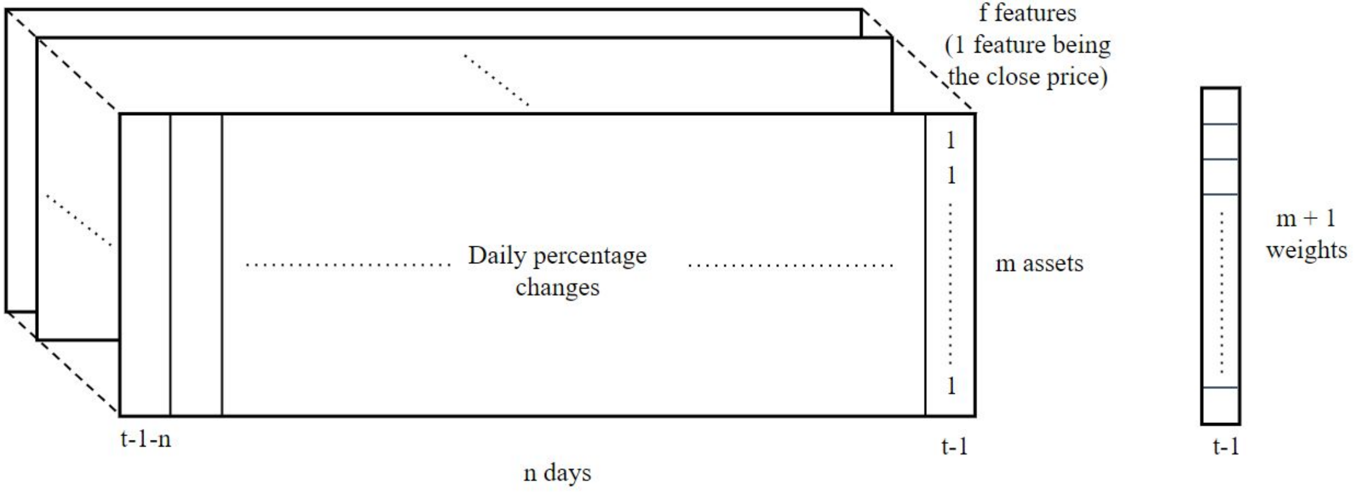
→ Match loss function with investor's goals i.e. maximize profits and constrain risk

→ Directly take into account also other factors such as commission costs

[1] A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem

State space

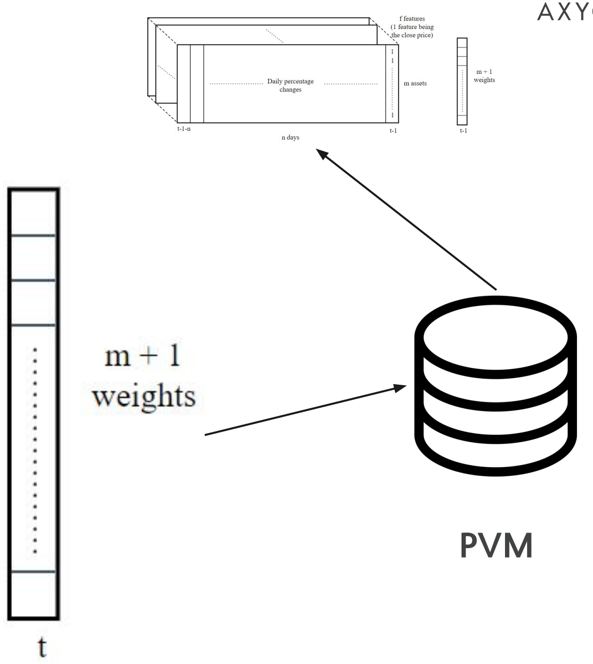
State space: price tensor (normalized prices + features) + last allocation weights



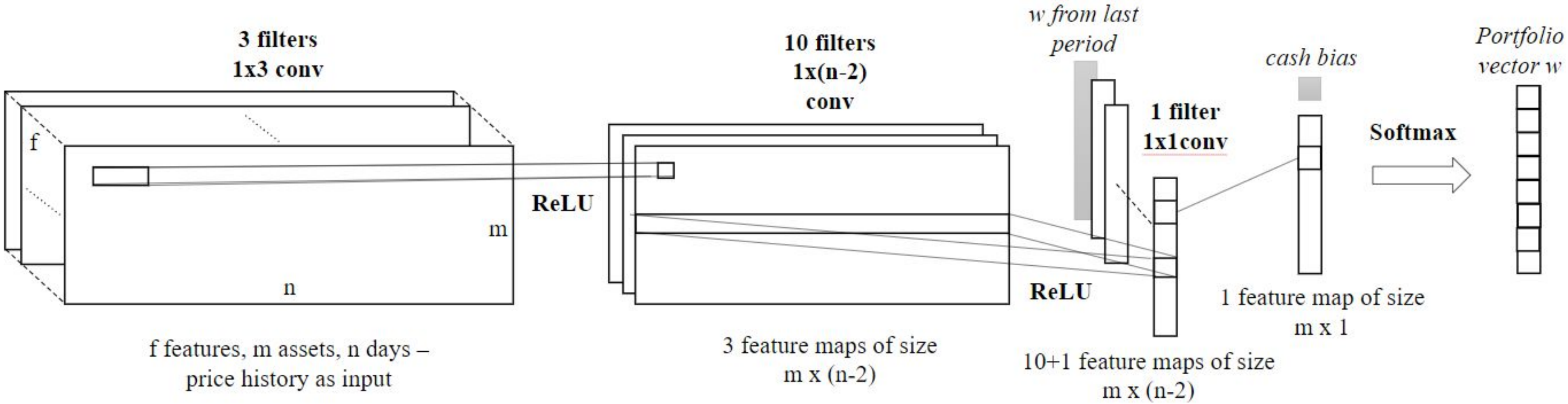
Action Space

Action space: new allocation weights w

Output allocation weights are stored in a Portfolio Vector Memory, and then re-used in the state definition



Policy Network



Reward function definition

$$L(w) = -\lambda_r R_r(w) + \lambda_a R_a(w) + \lambda_c R_c(w)$$

- L(w): loss function to be **minimized**, function of the action (allocation weights w)
- R_r: average **profits**, including transaction costs
- R_a: portfolio **exposure** to few assets
- R_c: **weights changes** between consecutive days

λ_r, λ_a, λ_c: scalar values that balance the contribution of each reward/penalty

Training loop

Train:

1. Initialize PVM to equally weighted allocation
2. Load a batch of price tensors
3. Pass it as input to the network with previous weights → compute new batch of weights
4. Perform optimization on loss function
5. Store new weights in PVM
6. Repeat from 2 (e.g. 40k times)

Backtest + online training:

1. Load one price tensor
2. Output single prediction and execute trade
3. Add new data to train set and perform few training iterations
4. Repeat from 1 until end of backtest

Results

Maximize portfolio value:

- $\lambda_r = 1$
- $\lambda_a = 0$
- $\lambda_c = 0$

$$L(w) = -\lambda_r R_r(w) + \lambda_a R_a(w) + \lambda_c R_c(w)$$

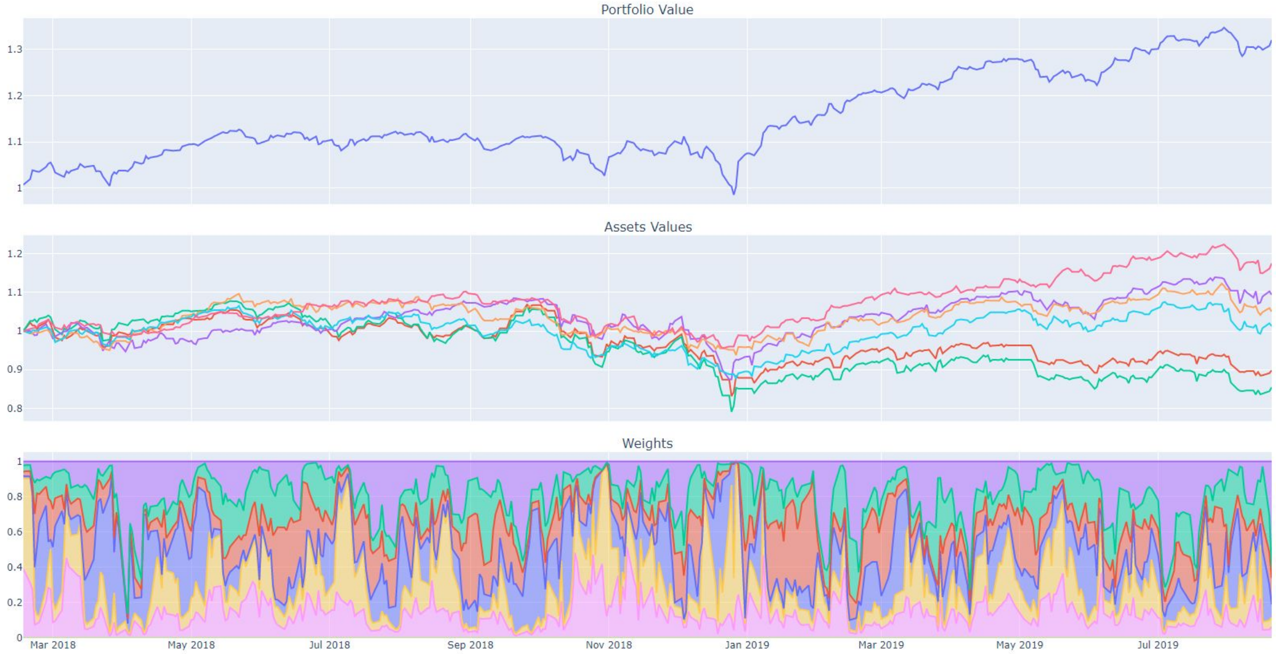
	Return	Sharpe
RL agent	57%	2.13
Equally weighted	2.5%	0.23

Results

Constrain exposure and daily switches:

- $\lambda_r = 1$
- $\lambda_a = 0.5$
- $\lambda_c = 0.5$

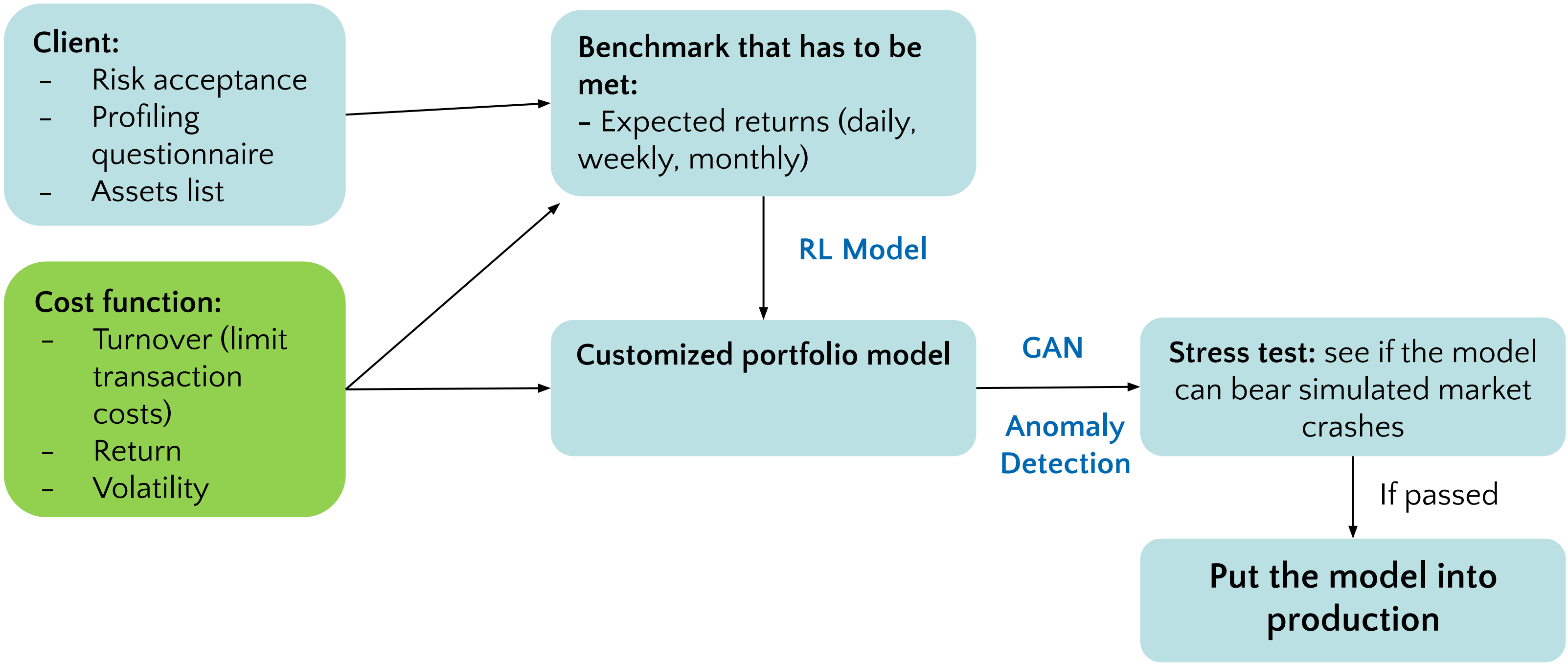
$$L(w) = -\lambda_r R_r(w) + \lambda_a R_a(w) + \lambda_c R_c(w)$$



	Return	Sharpe
RL agent	31%	1.57
Equally weighted	2.5%	0.23

AXYON.AI

AI-driven portfolio strategy creation



Looking for thesis projects - internships?

Drop us your CV at jacopo.credi@axyon.ai

For any question about the topic add my LinkedIn https://www.linkedin.com/in/fbassetto/

Suggested material about RL

Books:

- Reinforcement Learning: An Introduction

Courses:

- UCL - Course on RL (best theoretical explanation)
- HSE University - Practical RL (more hands-on)

Used indexes

AXYON.AI

ASX ALLORDINARIES

MSCI EURO

SP500

JAPAN SMALL CAP

TOPIX

Other parameters

AXYON.AI

Features:

- Rate of change (5d, 10d, 20d, 40d)
- Weighted Moving Average ratio (5d, 10d, 20d, 40d)

Window size:

- 40d

Normalization method:

- Normalize the close prices by the last day of the price tensor
- Leave all other features untouched