UiO : Department of Mathematics
University of Oslo

Trading Financial Markets Using Reinforcement Learning

Application and Analysis

Fábio Rodrigues Pereira

June 16, 2022

Table of contents

About me

About me

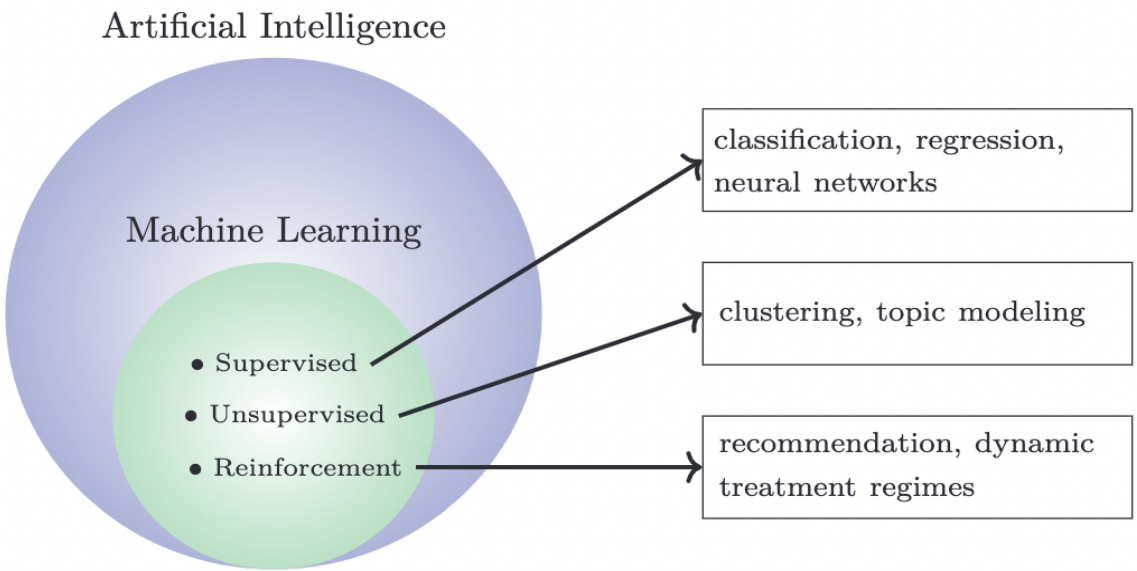
- I am 34 years old, and have Brazilian and Italian citizenships
- I have being investing in the financial markets since 2011
- (2007–2012) BSc. Law
- (2012) I passed the bar exam and started acting as a public defender for the state of Sao Paulo/Brazil
- (2012–2014) Pos-graduation in enterprise labor law
- (2014) I discovered a brain tumor, and changed life style
- (2015) I quit my job and career as lawyer, and moved to Europe
- (2016–2019) BSc. Mathematics and Economics at UiO
- (2020–2022) MSc Computer Science: Applied mathematics and risk analysis at UiO

Introduction

Motivation

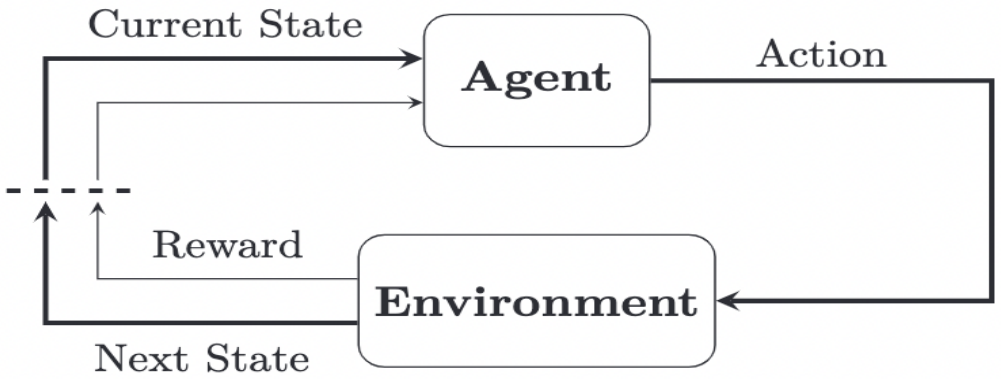
- The 4th Industrial revolution with Artificial intelligence
- AI has presence in many segments, **why not in finance?**
- Unexplored field, specifically RL in Financial Trading Systems

Differences among AI's sub-families



- SUPERVISED: Features/Instances and Classes/Labels/Targets
- UNSUPERVISED: Only Features
- REINFORCEMENT: Trial—and—error cumulative reward method

Why Reinforcement Leaning (RL)?



- The financial markets are dynamic and turbulent structures, too complex for straightforward algorithms.
- Automated strategies must be flexible, not wholly reliant on pattern–based strategy, and capable of learning and adjusting on the go–just like humans, but faster.
- Reinforcement learning models can handle a Markov decision process (MDP) without developing a theoretical model (transition probabilities and rewards), and without succumbing to the curse of costly computations owing to their dimensionality [2, AG15].

Foundation theory

Markov chains theory

Definition (Markov process)

A stochastic process $\{S_t,\ t = 0, 1, 2, \ldots\}$, taking values in the state space $\mathcal{S}$, e.g., $S_t = s \in \mathcal{S}$, is said to be a Markov process if following property is satisfied:

$$P(S_{t+1} = s' | S_0 = s_0, \ldots, S_{t-1} = s_{t-1}, S_t = s) = P(S_{t+1} = s' | S_t = s),$$

for all $s_0,\ \ldots\ ,\ s_{t-1},\ s,\ s' \in \mathcal{S}$ and $t \in \mathbb{N} \cup \{0\}$.

Remark

The **STATES** of a process are information from the environment, at time t, converted in mathematical language, e.g., scalar numbers, vectors, matrices, etc.

Markov decision process

Definition (Markov decision process)

The Markov decision process is obtained by embedding controls with feedback loops into the framework of a Markov process. In other words, a Markov decision process is a combination of a stochastic process (see 2.1.1), Markov chains theory (see 2.1.3), and a control mechanism (see 2.2.3), which can be defined mathematically by the tuple:

$$\Big\{\mathcal{S},\ \mathcal{A},\ p(s' \mid s,\ a),\ \mathcal{R},\ \gamma\Big\},$$

for $S_t = s,\ S_{t+1} = s' \in \mathcal{S}$, $A_t = a \in \mathcal{A}(S_t) \subseteq \mathcal{A}$, and $t \in \mathbb{N} \cup \{0\}$.

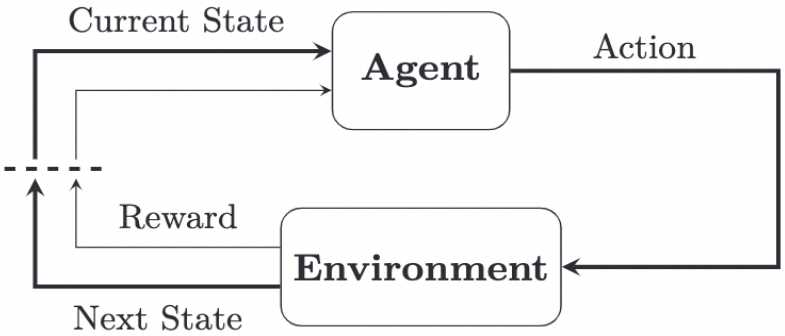
Remark

The symbol γ denotes a **DISCOUNT FACTOR**, and $p(s' \mid s,\ a)$ is the **TRANSITION PROBABILITY** of the process.

Reinforcement Learning framework

The dynamics of a reinforcement learning system can be described as a Markov decision process with four elements:

- **STATE:** $S_t \in \mathcal{S}$, or
  FEATURE: $\varphi_t \in \varphi(S_t)$
- **ACTION:** $A_t \in \mathcal{A}(S_t) \subseteq \mathcal{A}$
- **POLICY:** $\psi \in \psi(S_t,\ A_t)$
- **REWARD:** $R_t \in \mathcal{R}$



Remark

When the agent takes an action, the following transition occurs:

$$(S_{t+1}, R_{t+1}) \overset{\text{def}}{\sim} p(\circ|S_t, A_t),$$

Total cumulative reward (return)

The goal of the RL is to come up with a way to choose actions that will maximize the expected total cumulative reward. For that, we need to compute the return $G_{t:T}$, which is a function of the total cumulative reward. In a time–discounted case, it is the sum of the product between a received reward R_t and its discount factor γ that yields

$$G_{t:T} \stackrel{\text{def}}{=} \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T, \tag{1}$$

up to the final step T (terminal), which can be ∞ in some cases. Also, to save on costly computations, we can employ a computationally recursive method:

$$\begin{aligned} G_t &\stackrel{\text{def}}{=} \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \ldots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots \right) \\ &= R_{t+1} + \gamma G_{t+1}, \end{aligned} \tag{2}$$

where $T = \infty$ in this case.

Value Functions (1:2)

It is the **EXPECTED TOTAL CUMULATIVE REWARD**, or simply, **EXPECTED RETURN.** There are two different value functions:

1.) The value function that maps $V^{\psi} : \mathcal{S} \longrightarrow \mathbb{R}$ is called the **STATE–VALUE FUNCTION**, or simply, **STATE FUNCTION.** This function specifies a conditional expected return when the system begins in the state $S_0 = s$ and follows a fixed policy ψ thereafter. That is expressed mathematically as

$$V_t^{\psi}(s) \stackrel{\text{def}}{=} E^{\psi}\Big[G_t \mid S_t = s\Big], \tag{3}$$

for all $s \in \mathcal{S}$, where G_t is the return, and the superscript ψ on the expectation sign $E^{\psi}[\cdot]$ indicates that this expectation is also conditioned on a specific ψ being followed.

Value Functions (2:2)

2.) The value function that maps $Q^{\psi} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$ is called **STATE–ACTION–VALUE FUNCTION**, or simply, **ACTION FUNCTION.** In this variation, a state $S_0 = s$ and a first action $A_0 = a$ are arbitrarily selected, and afterward, their subsequent decisions are chosen by applying a fixed policy ψ. This policy selects all the subsequent actions and evaluates "*how good it is to perform a given action in a given state*" [1, SRBA19]. That is mathematically defined as

$$Q_t^{\psi}(s,\ a) \stackrel{\text{def}}{=} E^{\psi}\Big[G_t \mid S_t = s,\ A_t = a\Big], \tag{4}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.

Recursive relationship (1:2)

- For **STATE FUNCTIONS**:

$$
\begin{aligned}
V_t^{\psi}(s) &\overset{\text{def}}{=} E^{\psi}\Big[G_t \mid S_t = s\Big] \\
&\overset{(1)}{=} E^{\psi}\Big[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \mid S_t = s\Big] \\
&\overset{(2)}{=} E^{\psi}\Big[R_{t+1} + \gamma G_{t+1} \mid S_t = s\Big] \\
&\overset{(3)}{=} E^{\psi}\Big[R_{t+1} + \gamma V_{t+1}^{\psi}(S_{t+1}) \mid S_t = s\Big] \\
&\overset{(*)}{=} \sum_{a \in \mathcal{A}} \psi(a|s) \sum_{r,\ s'} p(r,\ s' \mid s,\ a)\ \Big[r + \gamma\ V_{t+1}^{\psi}(s')\Big],
\end{aligned}
$$

for all $r \in \mathcal{R}$ and $s' \in \mathcal{S}$, where G_{t+1} is an unbiased estimate for $V_{t+1}^{\psi}(S_{t+1})$. $(*)$ The **BELLMAN EQUATION** for state functions.

Recursive relationship (2:2)

- For **ACTION FUNCTIONS**:

$$
\begin{aligned}
Q_t^{\psi}(s,\ a) &\overset{\text{def}}{=} E^{\psi}\Big[G_t \mid S_t = s,\ A_t = a\Big] \\
&\overset{(1)}{=} E^{\psi}\Big[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \mid S_t = s,\ A_t = a\Big] \\
&\overset{(2)}{=} E^{\psi}\Big[R_{t+1} + \gamma G_{t+1} \mid S_t = s,\ A_t = a\Big] \\
&\overset{(4)}{=} E\Big[R_{t+1} + \gamma\ Q_{t+1}^{\psi}(S_{t+1}, A_{t+1}) \mid S_t = s,\ A_t = a\Big] \\
&\overset{(*)}{=} \sum_{s',r} p(s',\ r \mid s,\ a)\ \Big[r + \gamma\ Q_{t+1}^{\psi}(s', a')\Big],
\end{aligned}
$$

for all $r \in \mathcal{R}$, $s' \in \mathcal{S}$, and $a' \in \mathcal{A}$, where G_{t+1} is an unbiased estimate for $Q_{t+1}^{\psi}(S_{t+1}, A_{t+1})$. $(*)$ The **BELLMAN EQUATION** for action functions.

Function Approximation Reinforcement Learning

Why not tabular RL methods?

1. **DYNAMIC PROGRAMMING** (*necessary transition probabilities*)
   - Value iteration
   - Policy iteration

2. **MONTE CARLO** (*model–free algorithm*)
   - From real world state–action transitions' and rewards' samples, it is possible to estimate the real value function by averaging expected returns.
   - Must wait a complete trajectory of episodes.

3. **TEMPORAL DIFFERENCE** (*model–free algorithm*)
   - Also estimate the real value function by averaging expected returns from real world samples.
   - It does not need to wait a complete trajectory of episodes (on-line updates).

Stochastic gradient descent (1:8)

The objective is to develop a weighted linear function capable of approximating the value function in such a way that:

- $\widehat{V}(S_t;\ \overrightarrow{w}) \approx V^{\Psi}(S_t)$ as $t \longrightarrow \infty$, or
- $\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}) \approx Q^{\Psi}(S_t,\ A_t)$ as $t \longrightarrow \infty$.

Remark

- These estimate functions must be differentiable for all $S_t \in \mathcal{S}$, and $A_t \in \mathcal{A}$ (exclusively for action functions), with respect to w_j.

Stochastic gradient descent (2:8)

First, we need a weight vector, which is composed of real–valued components in the following manner:

$$\overrightarrow{w}_t \stackrel{\text{def}}{=} \Big[w_1, w_2, \ldots, w_d \Big]^T,$$

for $j = 1, 2, \ldots, d$, where T after the square brackets indicates that the weight vector is transposed, denoting a column vector.

Remark

- The dimensionality of $\overrightarrow{w}$ is typically less than the number of states, and thus altering one weight impacts the projected values of numerous other states.

Stochastic gradient descent (3:8)

The SGD technique attempts to decrease inaccuracy in estimations, e.g., to minimize a **MEAN SQUARED ERROR** (MSE) metric, by adjusting the weight vector after each iteration step t, such that

$$\begin{aligned}\overrightarrow{w}_{t+1} &\stackrel{\text{def}}{=} \overrightarrow{w}_t - \frac{1}{2}\,\eta_t\,\overrightarrow{\nabla}\Big[Q^{\Psi}(S_t,\ A_t) - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)\Big]^2 \\ &= \overrightarrow{w}_t + \eta_t\,\Big[Q^{\Psi}(S_t,\ A_t) - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)\Big]\,\overrightarrow{\nabla}\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t),\end{aligned} \tag{5}$$

where η_t is the learning rate at iteration step t, and $\overrightarrow{\nabla}\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)$ is the gradient vector for the value function that can be computed by:

$$\overrightarrow{\nabla}\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t) = \left[\frac{\partial\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)}{\partial w_1}, \ldots, \frac{\partial\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)}{\partial w_d}\right]^T. \tag{6}$$

Stochastic gradient descent (4:8)

Problem (1)

All of this theory is highly beneficial if we know the actual value function, or also called **TARGET VALUE FUNCTION**, $V^{\Psi}(S_t)$ or $Q^{\Psi}(S_t,\ A_t)$.

We may substitute the **TARGET VALUE FUNCTION** with an unbiased estimate, represented by θ_t. Note that if θ_t is an unbiased estimate for a **action equation**, then

$$Q^{\Psi}(S_t,\ A_t) \approx E^{\Psi}\Big[\theta_t | S_t = s,\ A_t = a\Big],$$

and the equation (5) turns to be

$$\overrightarrow{w}_{t+1} \overset{\text{def}}{=} \overrightarrow{w}_t + \eta_t \left[\theta_t - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)\right] \overrightarrow{\nabla}\widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t), \qquad (7)$$

which is guaranteed to converge to a local optimum [1, SRBA19].

Stochastic gradient descent (5:8)

Definition (Feature Vector)

A vector $\overrightarrow{\varphi}_t \in \mathbb{R}^d$, which has the same number of dimensions d as in $\overrightarrow{w}_t$, such that

$$\overrightarrow{\varphi}_t \stackrel{\text{def}}{=} \Big[\varphi_1(S_t,\ A_t), \varphi_2(S_t,\ A_t), \ldots, \varphi_d(S_t,\ A_t)\Big]^T, \tag{8}$$

for $i = 1, 2, \ldots, d$, where T after the square brackets indicates that the vector is transposed, being a column vector.

Definition (Basis Function)

Each component of the feature vector is a mapping $\varphi_i : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}^d$, which must be differentiable, and receives the name of **BASIS FUNCTION**.

Stochastic gradient descent (6:8)

Definition (Estimate Function)

Hence, the estimate value function is linearly formed with a combination of a weight vector $\overrightarrow{w}_t$ and a feature vector $\overrightarrow{\varphi}_t$, i.e., is given by the inner vector product between the weight vector and the feature vector:

$$\begin{aligned} \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t) &\stackrel{\text{def}}{=}\ \overrightarrow{w}_t^T\ \overrightarrow{\varphi}_t \\ &= \sum_{i,j=1}^{d}\ w_j\ \varphi_i(S_t, A_t), \end{aligned} \tag{9}$$

for all $S_t \in \mathcal{S}$ and $A_t \in \mathcal{A}$.

Stochastic gradient descent (7:8)

Remark

$$
\begin{aligned}
\overrightarrow{\nabla}\widehat{Q}(S_t, A_t;\ \overrightarrow{w}_t) &\overset{(6)}{=} \left[\frac{\partial\widehat{Q}(S_t, A_t;\ \overrightarrow{w}_t)}{\partial w_1}, \ldots, \frac{\partial\widehat{Q}(S_t, A_t;\ \overrightarrow{w}_t)}{\partial w_d}\right]^T \\
&\overset{(9)}{=} \left[\frac{\partial}{\partial w_1}\left[w_1 \cdot \varphi_1(S_t, A_t)\right], \ldots, \frac{\partial}{\partial w_d}\left[w_d \cdot \varphi_d(S_t, A_t)\right]\right]^T \\
&\overset{(*)}{=} \left[\varphi_1(S_t, A_t), \ldots, \varphi_d(S_t, A_t)\right]^T \\
&\overset{(8)}{=} \overrightarrow{\varphi}_t.
\end{aligned}
$$

$(*)$ From partial–derivative rules.

Stochastic gradient descent (8:8)

Remark

Notice now that the weight's update routine, given by equation (7), yields

$$\begin{aligned}
\overrightarrow{w}_{t+1} &\overset{\text{def}}{=} \overrightarrow{w}_t + \eta_t \left[G_t - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t) \right] \overrightarrow{\varphi}_t \\
&\overset{(2)}{=} \overrightarrow{w}_t + \eta_t \left[R_{t+1} + G_{t+1} - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t) \right] \overrightarrow{\varphi}_t \\
&\overset{(4)}{=} \overrightarrow{w}_t + \eta_t \left[R_{t+1} + \widehat{Q}(S_{t+1},\ A_{t+1};\ \overrightarrow{w}_t) - \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t) \right] \overrightarrow{\varphi}_t
\end{aligned} \tag{10}$$

which is guaranteed to converge to a point near the local minimum under the learning rate's usual conditions [6, RM51].

Control Mechanisms

- **ON–POLICY SARSA**: We attain the chosen actions $A_t, A_{t+1}, \ldots$ by the ϵ**–GREEDY POLICY**, such that

$$A_t = \begin{cases} \arg\max_{a \in A(S_t)} \widehat{Q}(S_t,\ a;\ \overrightarrow{w}_t) & \text{if prob. } 1-\epsilon, \\ U_t\Big(\mathcal{A}(S_t)\Big) & \text{if prob. } \epsilon, \end{cases}$$

  where U_t randomly draws a uniformly distributed action from $\mathcal{A}(S_t)$.

- **OFF–POLICIES Q–LEARNING** and **GREEDY–GQ**: We attain the chosen action $A_t, A_{t+1}, \ldots$ by the **GREEDY POLICY**, such that

$$A_t = \arg \max_{a \in A(S_t)} \widehat{Q}(S_t,\ a;\ \overrightarrow{w}_t).$$

Greedy–GQ's weight update

In summary, we need a supplementary sequence of weights $\overrightarrow{\kappa}_t \in \mathbb{R}^d$ and an extra step–size parameter ζ_t. This innovation culminates in the same target value function θ_{t+1} as in Q–learning, but with different weight–update guidelines:

$$
\begin{aligned}
\theta_{t+1} &= R_{t+1} + \gamma \max_{a' \in \mathcal{A}} \widehat{Q}(S_{t+1},\ a'; \overrightarrow{w}_t) \\
\vartheta_{t+1} &\longleftarrow \left[\theta_{t+1} - \widehat{Q}(S_t,\ A_t; \overrightarrow{w}_t)\right] \\
\delta_{t+1} &\longleftarrow \vartheta_{t+1} \overrightarrow{\nabla} \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t)\ -\ \gamma\, \widehat{Q}(S_t,\ A_t; \overrightarrow{\kappa}_t) \\
\overrightarrow{w}_{t+1} &\longleftarrow \overrightarrow{w}_t\ +\ \eta_t\, \delta_{t+1}\ \overrightarrow{\nabla} \widehat{Q}(S_{t+1},\ A_{t+1};\ \overrightarrow{w}_t) \\
\overrightarrow{\kappa}_{t+1} &\longleftarrow \overrightarrow{\kappa}_t + \zeta_t \left[\vartheta_{t+1} - \widehat{Q}(S_t,\ A_t; \overrightarrow{\kappa}_t)\right] \overrightarrow{\nabla} \widehat{Q}(S_t,\ A_t;\ \overrightarrow{w}_t),
\end{aligned}
$$

for $A_{t+1} \sim \arg\max_{a' \in \mathcal{A}} \widehat{Q}(S_{t+1},\ a'; \overrightarrow{w}_t)$.

Application

State–of–the–art RL algorithms applied to financial markets

- **Framework**: function approximation reinforcement learning.
- **Control Mechanisms**: On-policy SARSA, and Off-policies Q–Learning and Greedy–GQ.
- **Market**: B3 Futures – Sao Paulo/Brazil Stock–exchange.
- **Framed Intervals**: 60min, and 500k–tick [3, MLPrado21].
- **Contracts**: WINJ21, WINM21, WINQ21, WINV21, WING21, WING22.
- **Period**: 02.17.21–02.15.22.
- **Initial Capital**: BRL 5.600,00 (two margin-calls) = 28k points.
- **Orders' Type**: Limit.
- **Number of Contracts Traded per Time**: One.
- **Broker's Commission**: BRL 0,00

State Matrix

Time	Open	High	Low	Close
05.29.2021 09:00:00	10.00	13.00	09.00	12.00
05.29.2021 10:00:00	12.00	14.00	11.00	14.00
05.29.2021 11:00:00	14.00	16.00	10.00	10.00

Table: Example of a state matrix $\mathbf{S} \in \mathbb{R}^{(2+1)\times 4}$, where $n = 2$ and $t =$"05.29.2021 11:00".

Remark

Our reinforcement learning problem is formalized as a discrete–time stochastic control process $\{\mathcal{S}, \mathcal{A}, \mathcal{R};\ t = 0, 1, 2, \ldots\}$ in which an algorithm trader interacts with a market environment. This algorithm begins by gathering an initial observation ($S_t = \mathbf{S}$, where $S_t \in \mathcal{S}$) of the market. Every market observation $S_t \in \mathbb{R}^{(n+1)\times 4}$ comprises a matrix of $(n+1)$ continuing time with OHLC[a] price values.

[a] Open, high, low, close prices.

Possible Actions

If *tradingStatus*$= -1$ (short on the market), then

$$a = \begin{cases} 0 & \text{do nothing,} \\ +1 & \text{go long.} \end{cases}$$

If *tradingStatus*$= 0$ (not on the market), then

$$a = \begin{cases} -1 & \text{go short,} \\ 0 & \text{do nothing,} \\ +1 & \text{go long.} \end{cases}$$

If *tradingStatus*$= 1$ (long on the market), then

$$a = \begin{cases} -1 & \text{go short,} \\ 0 & \text{do nothing.} \end{cases}$$

Reward Function

We study three choices for this function:

a.) The first one we identify by the name of *minusMean*,

$$R_{t+1} \stackrel{\text{def}}{=} \mathrm{R}_{t+1} - \text{mean}(\overrightarrow{R}).$$

b.) The second one is referred to as *immediate*, which simply takes the immediate reward of the current time step.

c.) We specify the third one as *mean*

$$R_{t+1} \stackrel{\text{def}}{=} \text{mean}(\overrightarrow{R}),$$

which computes the mean of the appended evolution of the rewards.

Feature Vector

We employ a **BLOCK REPRESENTATION** of the feature vector, also known as **FEATURE EXPANSION TECHNIQUE**, which was first proposed by [4, GDRRH11] and further enhanced by [5, GWRH13]. This system, which was developed specifically for estimations of linear reinforcement learning, copies the basis vector to one of the three slots in the feature vector, applying the constraint:

$$\overrightarrow{\varphi} = \begin{cases} \begin{bmatrix} \overrightarrow{b} & \overrightarrow{0} & \overrightarrow{0} \end{bmatrix}^T & \text{if } A_t = -1, \\ \begin{bmatrix} \overrightarrow{0} & \overrightarrow{b} & \overrightarrow{0} \end{bmatrix}^T & \text{if } A_t = 0, \\ \begin{bmatrix} \overrightarrow{0} & \overrightarrow{0} & \overrightarrow{b} \end{bmatrix}^T & \text{if } A_t = 1, \end{cases} \tag{11}$$

where $\overrightarrow{0} \in \mathbb{R}^{n+1}$, and $\overrightarrow{\varphi} \in \mathbb{R}^d$, where $d = |\mathcal{A}| \cdot (n+1)$.

Basis Function

We introduce three types of basis functions:

a.) A **SIGMOID** function

$$b_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}}. \tag{12}$$

b.) A **HYPERBOLIC TANGENT** function

$$b_{\text{hypTanh}}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{13}$$

c.) A **SIGMOID123**[1] function

$$b_{\text{sigmoid123}}(x) = \frac{2}{1 + e^{-10^{15} \cdot x}} + 1, \tag{14}$$

[1] We derived this formulation from academic papers [Corazza15] and [Corazza19], in which the authors achieved positive outcomes.

Basis Vector

The basis vector will be an element of the feature vector subsequently. This basis vector $\overrightarrow{b} \in \mathbb{R}^{n+1}$ has the following elements:

$$b_i = \begin{cases} b\Big(I(\mathbf{S})\Big) & \text{if } i = 1, 2, \ldots, n. \\ b\Big(I(\overline{R})\Big) & \text{if } i = n+1, \end{cases} \tag{15}$$

where the function $I(\cdot)$ is defined as the log return, and $\overline{R}$ is the current trade profit or loss (PL)'s ratio, such that

$$I(x) = \begin{cases} \ln\left(\frac{x[j+1,\ 4]}{x[j,\ 4]}\right) & \text{if } x = \mathbf{S}, \\ \ln\left(\frac{\text{current close price}}{|\text{entry price}|}\right) & \text{if } x = \overline{R}, \end{cases} \tag{16}$$

for $j = 1, 2, \ldots, n$, and the column index number 4 represents the close time in the state matrix.

An Algorithm example

Algorithm 6: Pseudocode for estimate SARSA algorithm.

begin
  By initializing the constructor with
  - $\mathbf{S} \longleftarrow$ Environment.
  - $A \sim \epsilon$–greedy policy.

for *each time step* **do**
  $A \longrightarrow$ Environment.
  $\mathbf{S}', R' \longleftarrow$ Environment.
  $A' \sim \epsilon$–greedy policy.
  $\delta \longleftarrow r(R') + \gamma \widehat{Q}(\mathbf{S}',\ A';\ \overrightarrow{w}) - \widehat{Q}(\mathbf{S},\ A;\ \overrightarrow{w})$.
  $\overrightarrow{w} \longleftarrow \overrightarrow{w} + \eta\ \delta\ \overrightarrow{\nabla} \widehat{Q}(\mathbf{S}, A;\ \overrightarrow{w})$.
  $\eta \longleftarrow lrSchedulerFct.$
  $\mathbf{S} \longleftarrow \mathbf{S}'$.
  $A \longleftarrow A'$.
  $t \longleftarrow t + 1$.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 72.

Results

Benchmark – Buy and Hold Strategy

Code	Period	First trade	Last trade	B&H PL
WINJ21	17.02.21 - 13.04.21	119,695	119,010	-685
WINM21	14.04.21 - 15.06.21	119,490	130,075	10,585
WINQ21	16.06.21 - 17.08.21	130,705	117,150	-13,555
WINV21	18.08.21 - 12.10.21	118,890	112,100	-6,790
WINZ21	13.10.21 - 14.12.21	113,370	106,520	-6,850
WING22	15.12.21 - 15.02.22	108,400	115,135	6,735
Total	17.02.21 - 15.02.22	-	-	-10,560

Table 6.1: Buy–and–hold benchmark strategy, comprising the first and last trade prices, as well as the profit or loss of the periods, all of which are measured by points.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 75.

Pipeline and Hyper–Parameter Search

For each framed intervals of 60min and 500k–tick, we performed:

hyper–parameters	Values
$rlType$	SARSA, QLearn, Greedy–GQ
n	5, 25, 50
$basisFctType$	sigmoid, sigmoid123, hypTanh
$rewardType$	minusMean, immediate, mean
η	0.1, 0.01
ζ	0.1, 0.01
γ	1, 0.95
ϵ	0.15, 0.1
$lrScheduler$	0, 200
$initType$	uniform01, zeros
$seeds$	1, ..., 51

Table 6.2: hyper–parameters' settings for the pipeline.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 76.

Results for 60min: Top Loser

	hyper–parameters	Final PL
1	**QLearn, 50, sigmoid123, minusMean, 0.01, 0.95, 200**	**-20,487**
2	**Greedy–GQ, 25, sigmoid, minusMean, 0.01, 1, 0.01, 0**	**-13,995**
3	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 1, 0.01, 200	-13,995
4	QLearn, 50, sigmoid123, minusMean, 0.01, 0.95, 0	-13,304
5	Greedy–GQ, 5, sigmoid123, minusMean, 0.01, 1, 0.01, 0	-10,477
6	Greedy–GQ, 50, sigmoid123, minusMean, 0.01, 0.95, 0.1, 200	-8,880
7	Greedy–GQ, 50, sigmoid123, minusMean, 0.01, 0.95, 0.1, 0	-8,838
8	Greedy–GQ, 5, sigmoid123, immediate, 0.1, 1, 0.1, 200	-8,512
9	Greedy–GQ, 50, sigmoid123, minusMean, 0.01, 1, 0.1, 200	-8,292
10	Greedy–GQ, 50, sigmoid123, immediate, 0.1, 1, 0.1, 200	-7,828
11	Greedy–GQ, 50, sigmoid123, minusMean, 0.01, 1, 0.1, 0	-7,706
12	QLearn, 50, sigmoid, minusMean, 0.01, 0.95, 0	-7,650
13	**SARSA, 50, sigmoid123, minusMean, 0.01, 0.95, 0.15, uniform01, 200**	**-7,531**
14	Greedy–GQ, 5, sigmoid, minusMean, 0.01, 1, 0.01, 0	-7,314
15	SARSA, 50, sigmoid123, minusMean, 0.01, 1, 0.1, uniform01, 200	-7,230
16	QLearn, 5, hypTanh, immediate, 0.1, 0.95, 0	-7,050
17	QLearn, 5, hypTanh, immediate, 0.1, 1, 0	-7,050
18	Greedy–GQ, 25, sigmoid, mean, 0.1, 0.95, 0.01, 0	-7,008
19	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 1, 0.1, 0	-6,995
20	SARSA, 50, sigmoid123, minusMean, 0.01, 0.95, 0.15, zeros, 200	-6,762

Table 6.3: Top 20 worst combination of hyper–parameters for 60 minutes' time–framed periods. Final profit and loss values (Final PL) presented in points.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 77.

Results for 60min: Top Winner

	hyper–parameters	Final PL
1	**Greedy–GQ, 5, sigmoid, minusMean, 0.01, 0.95, 0.1, 200**	**16,896**
2	**QLearn, 5, sigmoid, minusMean, 0.01, 0.95, 0**	**16,073**
3	QLearn, 5, sigmoid123, minusMean, 0.01, 0.95, 0	15,615
4	QLearn, 5, sigmoid123, minusMean, 0.01, 0.95, 200	15,450
5	QLearn, 25, sigmoid, minusMean, 0.1, 0.95, 0	14,028
6	QLearn, 5, sigmoid, minusMean, 0.01, 0.95, 200	13,076
7	QLearn, 25, sigmoid, minusMean, 0.1, 0.95, 200	12,721
8	QLearn, 25, sigmoid123, minusMean, 0.1, 1, 200	12,550
9	Greedy–GQ, 50, sigmoid, minusMean, 0.01, 0.95, 0.01, 200	12,374
10	QLearn, 5, sigmoid, minusMean, 0.1, 0.95, 0	12,033
11	QLearn, 25, hypTanh, minusMean, 0.1, 1, 0	12,003
12	QLearn, 25, hypTanh, minusMean, 0.1, 0.95, 0	12,001
13	QLearn, 25, hypTanh, minusMean, 0.1, 1, 200	11,348
14	QLearn, 25, hypTanh, minusMean, 0.1, 0.95, 200	11,348
15	QLearn, 50, hypTanh, minusMean, 0.1, 1, 200	10,926
16	QLearn, 50, hypTanh, minusMean, 0.1, 0.95, 200	10,926
17	**SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.1, zeros, 200**	**10,776**
18	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.1, zeros, 200	10,689
19	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.1, zeros, 200	10,610
20	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.1, zeros, 200	10,610

Table 6.4: Top 20 best combination of hyper–parameters for 60 minutes' time–framed periods. Final profit and loss values (Final PL) presented in points.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 79.

Results for 500k–tick: Top Loser

	hyper–parameters	Final PL
1	QLearn, 25, sigmoid, minusMean, 0.01, 0.95, 200	-70,280
2	QLearn, 25, sigmoid, minusMean, 0.01, 0.95, 0	-61,570
3	Greedy–GQ, 5, sigmoid, minusMean, 0.01, 1, 0.1, 0	-52,360
4	QLearn, 25, sigmoid, mean, 0.01, 0.95, 200	-48,430
5	Greedy–GQ, 5, sigmoid, minusMean, 0.01, 1, 0.1, 200	-47,554
6	Greedy–GQ, 5, sigmoid, minusMean, 0.01, 0.95, 0.1, 0	-47,363
7	QLearn, 25, sigmoid, immediate, 0.01, 0.95, 0	-43,732
8	Greedy–GQ, 5, sigmoid, minusMean, 0.01, 0.95, 0.1, 200	-41,756
9	QLearn, 50, sigmoid, minusMean, 0.01, 0.95, 200	-41,670
10	QLearn, 25, sigmoid, mean, 0.01, 0.95, 0	-40,501
11	QLearn, 50, sigmoid, immediate, 0.01, 0.95, 200	-37,944
12	SARSA, 25, sigmoid, minusMean, 0.01, 0.95, 0.1, uniform01, 0	-36,189
13	QLearn, 25, sigmoid, minusMean, 0.01, 1, 200	-34,410
14	QLearn, 5, sigmoid123, minusMean, 0.01, 1, 200	-33,302
15	QLearn, 5, sigmoid, mean, 0.1, 0.95, 200	-31,826
16	QLearn, 25, sigmoid, minusMean, 0.01, 1, 0	-30,961
17	QLearn, 25, sigmoid123, minusMean, 0.01, 0.95, 200	-28,284
18	SARSA, 25, sigmoid, immediate, 0.01, 0.95, 0.15, uniform01, 0	-23,956
19	Greedy–GQ, 5, sigmoid123, minusMean, 0.01, 1, 0.01, 200	-23,856
20	SARSA, 25, sigmoid, immediate, 0.01, 0.95, 0.1, uniform01, 0	-23,089
21	SARSA, 25, sigmoid, minusMean, 0.01, 0.95, 0.1, uniform01, 200	-22,099
22	SARSA, 25, sigmoid, minusMean, 0.01, 0.95, 0.1, zeros, 0	-21,759
23	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 0.95, 0.1, 200	-21,698
24	SARSA, 25, sigmoid, immediate, 0.01, 0.95, 0.1, zeros, 0	-21,313
25	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 1, 0.1, 200	-20,817
26	QLearn, 5, sigmoid, minusMean, 0.01, 0.95, 0	-20,262
27	QLearn, 5, sigmoid123, minusMean, 0.01, 0.95, 200	-20,210
28	SARSA, 25, sigmoid123, minusMean, 0.01, 0.95, 0.1, zeros, 0	-19,976
29	SARSA, 25, sigmoid, mean, 0.01, 0.95, 0.1, uniform01, 0	-19,959
30	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 0.95, 0.1, 0	-19,846
31	SARSA, 25, sigmoid, minusMean, 0.01, 0.95, 0.15, uniform01, 0	-19,450
32	Greedy–GQ, 5, sigmoid123, minusMean, 0.01, 1, 0.01, 0	-19,302
33	Greedy–GQ, 25, sigmoid, minusMean, 0.01, 1, 0.1, 0	-19,237
34	QLearn, 5, hypTanh, immediate, 0.1, 1, 0	-18,865
35	Greedy–GQ, 5, sigmoid123, minusMean, 0.1, 0.95, 0.01, 200	-18,638
36	QLearn, 5, sigmoid, minusMean, 0.01, 0.95, 200	-18,635
37	Greedy–GQ, 5, sigmoid, immediate, 0.1, 0.95, 0.1, 200	-18,472
38	SARSA, 25, sigmoid, minusMean, 0.01, 0.95, 0.15, zeros, 0	-18,393
39	QLearn, 25, sigmoid123, minusMean, 0.01, 0.95, 0	-18,316
40	SARSA, 25, sigmoid, immediate, 0.01, 0.95, 0.15, zeros, 0	-18,232
⋮	⋮	⋮
49	QLearn, 5, hypTanh, immediate, 0.1, 0.95, 200	-15,083
50	QLearn, 5, hypTanh, immediate, 0.1, 1, 200	-15,083

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 83.

Results for 500k–tick: Top Winner

	hyper–parameters	Final PL
1	**SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.1, uniform01, 0**	**48,246**
2	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.1, uniform01, 200	47,756
3	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.1, uniform01, 200	47,756
4	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.1, uniform01, 0	47,655
5	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.1, uniform01, 200	45,236
6	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.1, uniform01, 200	45,209
7	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.1, zeros, 200	43,735
8	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.1, zeros, 200	43,735
9	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.1, zeros, 200	43,723
10	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.1, zeros, 200	43,723
11	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.1, uniform01, 0	43,168
12	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.1, uniform01, 0	43,167
13	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.15, uniform01, 200	42,157
14	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.15, uniform01, 200	42,157
15	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.15, zeros, 200	41,477
16	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.15, zeros, 200	41,477
17	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.15, zeros, 200	41,449
18	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.15, zeros, 200	41,449
19	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.15, zeros, 0	41,343
20	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.15, zeros, 0	41,343
21	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.15, zeros, 0	40,997
22	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.1, zeros, 0	40,997
23	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.15, zeros, 0	40,873
24	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.1, zeros, 0	40,798
25	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.1, zeros, 0	40,214
26	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.1, zeros, 0	40,214
27	**QLearn, 5, hypTanh, minusMean, 0.1, 0.95, 200**	**36,675**
28	QLearn, 5, hypTanh, minusMean, 0.1, 1, 200	36,604
29	SARSA, 5, hypTanh, minusMean, 0.01, 1, 0.15, uniform01, 0	36,039
30	SARSA, 5, hypTanh, minusMean, 0.01, 0.95, 0.15, uniform01, 0	36,039
31	QLearn, 5, hypTanh, minusMean, 0.1, 0.95, 0	31,825
32	QLearn, 5, hypTanh, minusMean, 0.1, 1, 0	31,701
33	QLearn, 5, hypTanh, minusMean, 0.01, 0.95, 200	29,864
34	QLearn, 5, hypTanh, minusMean, 0.01, 1, 200	29,834
35	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.15, uniform01, 200	28,968
36	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.15, uniform01, 200	28,968
37	SARSA, 5, hypTanh, minusMean, 0.1, 0.95, 0.15, uniform01, 0	26,993
38	SARSA, 5, hypTanh, minusMean, 0.1, 1, 0.15, uniform01, 0	26,768
39	QLearn, 50, sigmoid, immediate, 0.1, 0.95, 0	26,287
40	QLearn, 50, sigmoid, mean, 0.1, 0.95, 0	25,858
⋮	⋮	⋮
49	**Greedy–GQ, 5, sigmoid123, minusMean, 0.1, 0.95, 0.1, 200**	**21,159**
50	SARSA, 25, hypTanh, minusMean, 0.01, 1, 0.15, uniform01, 200	20,655

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 84.

Return on Investment (ROI)

Algorithm	Period	ROI for 60 min	ROI for 500k ticks
Buy & Hold	17.02.21 - 15.02.22	-37.71%	-37.71%
Greedy–GQ	17.02.21 - 15.02.22	60.34%	75.56%
QLearn	17.02.21 - 15.02.22	57.40%	130.98%
SARSA	17.02.21 - 15.02.22	38.48%	172.30%

Table 6.7: Return on investment (ROI) for the different and best set ups. The initial investment capital considered for our application cases was 28,000 points.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 85.

Analysis

Best Models

Algorithm	Parameters	time–framed interval
Greedy–GQ	5, sigmoid, minusMean, 0.01, 0.95, 0.1, 200	60 min
Greedy–GQ	5, sigmoid123, minusMean, 0.1, 0.95, 0.1, 200	500k ticks
QLearn	5, sigmoid, minusMean, 0.01, 0.95, 0	60 min
QLearn	5, hypTanh, minusMean, 0.1, 0.95, 200	500k ticks
SARSA	5, hypTanh, minusMean, 0.01, 1, 0.1, zeros, 200	60 min
SARSA	5, hypTanh, minusMean, 0.1, 1, 0.1, uniform01, 0	500k ticks

Table 6.8: Optimal parameters for each reinforcement learning model from **Section 6.1**.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 86.

Best Results

		Mean final return	
Algorithm	**time–framed interval**	**50 seeds**	**500 seeds**
Greedy–GQ	60 min	44,896	39,907
Greedy–GQ	500k ticks	49,159	44,622
QLearn	60 min	44,073	42,372
QLearn	500k ticks	64,675	65,942
SARSA	60 min	38,776	36,604
SARSA	500k ticks	76,246	74,744

Table 6.9: Mean final returns in points for different number of seeds (simulations). The values for 50 seeds were extracted from **Section 6.1**.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 86.

Greedy–GQ: 60min

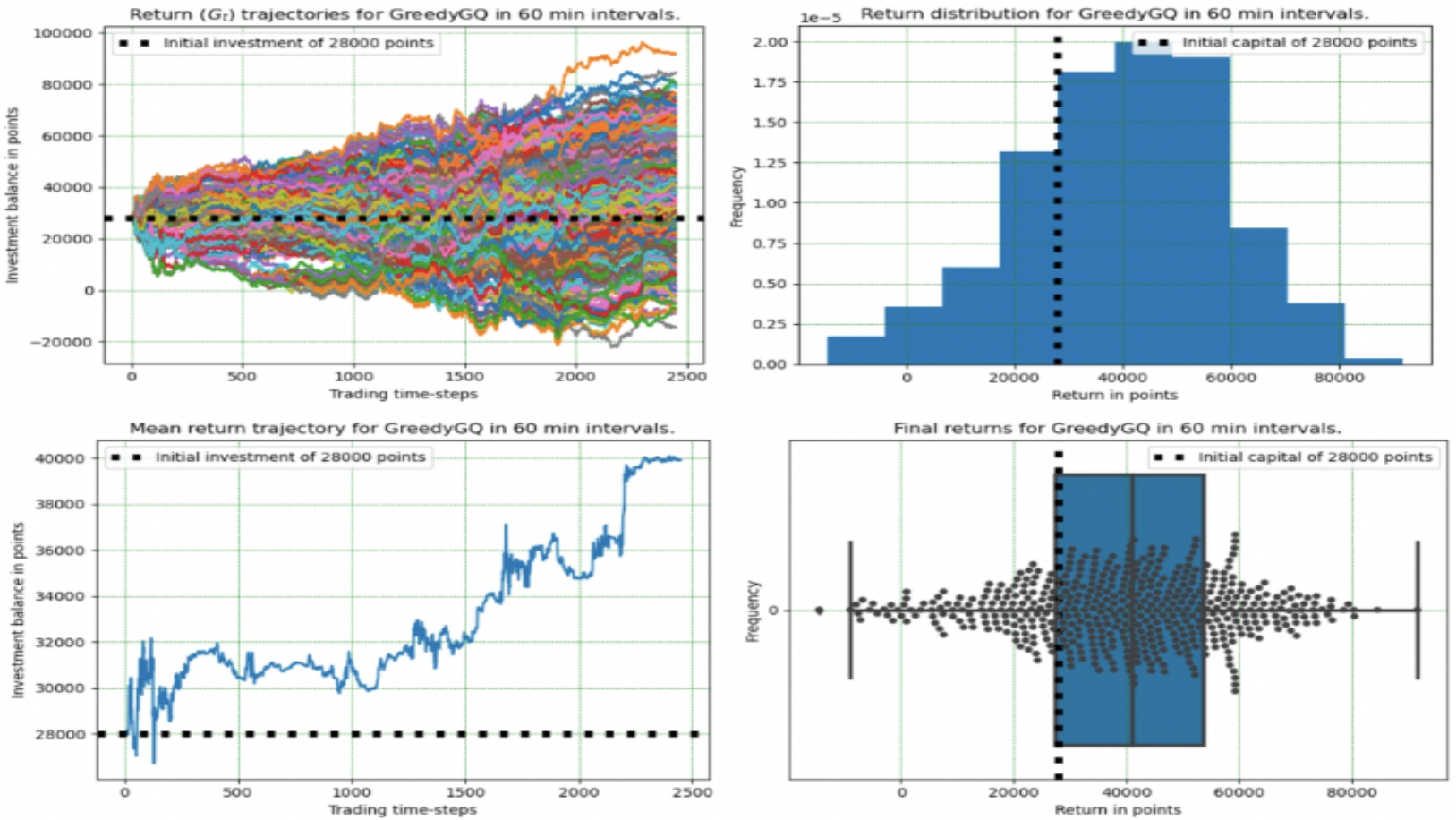


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 87.

Q-Learning: 60min

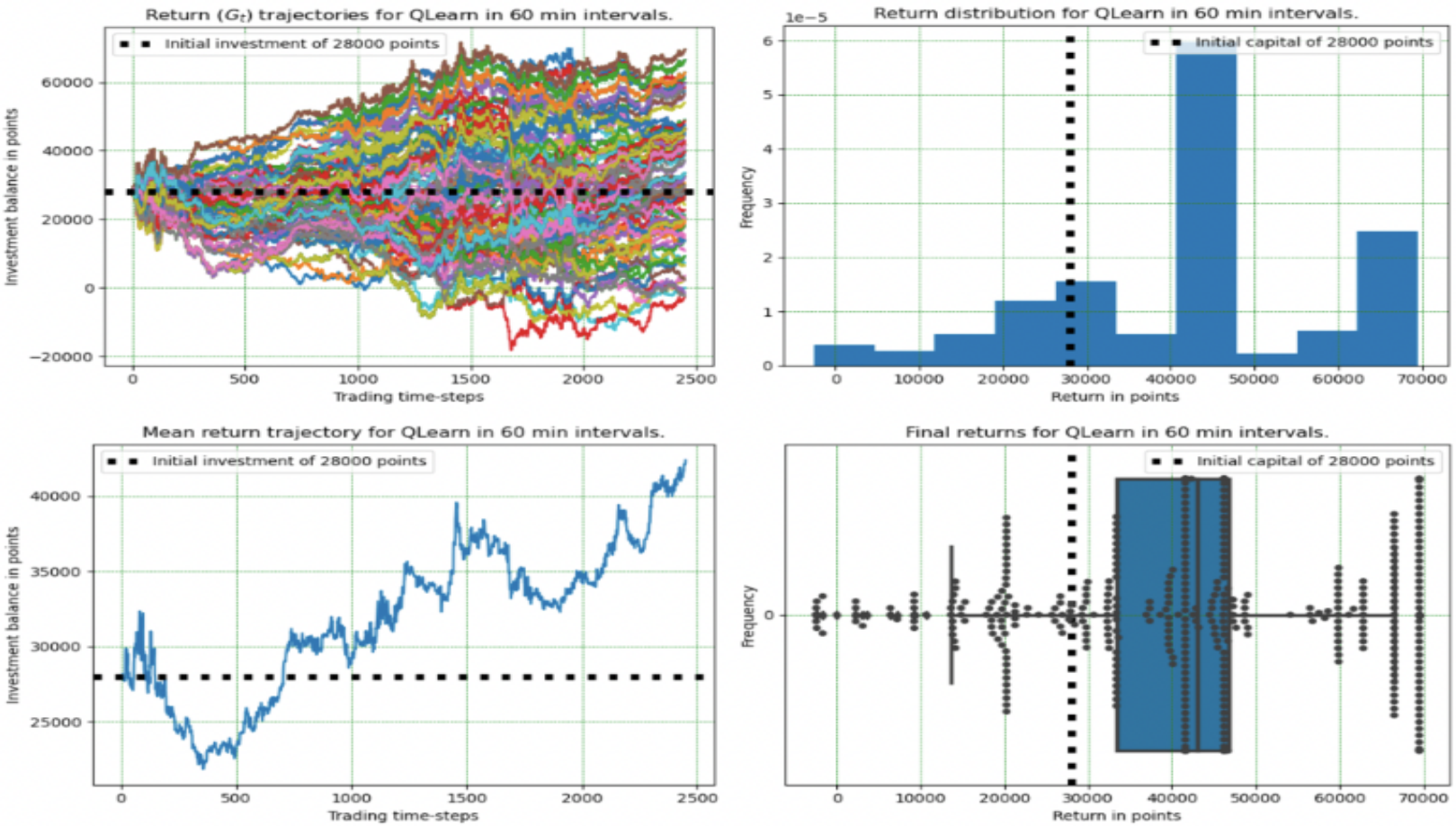


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 87.

SARSA: 60min

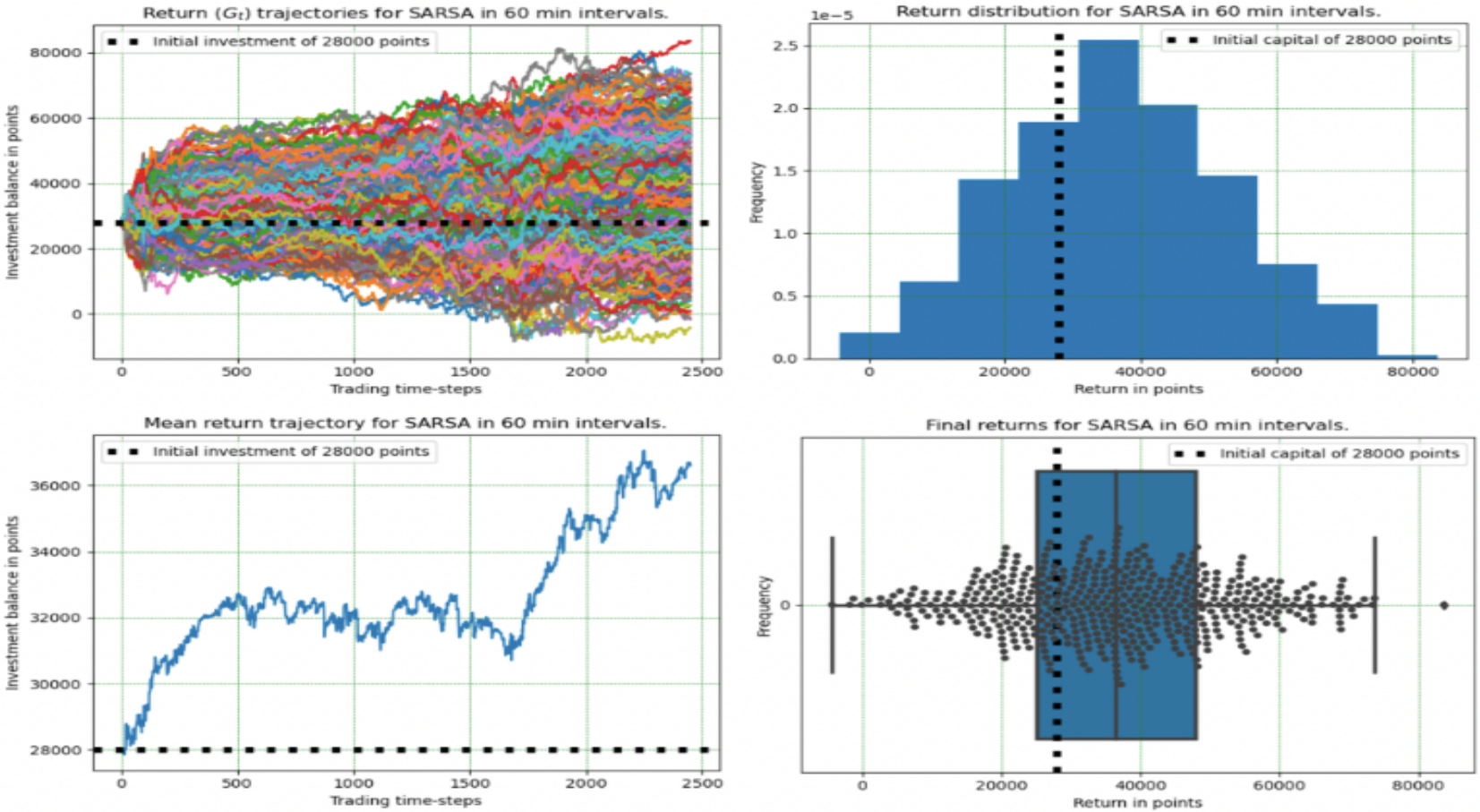


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 88.

Greedy–GQ: 500k–Tick

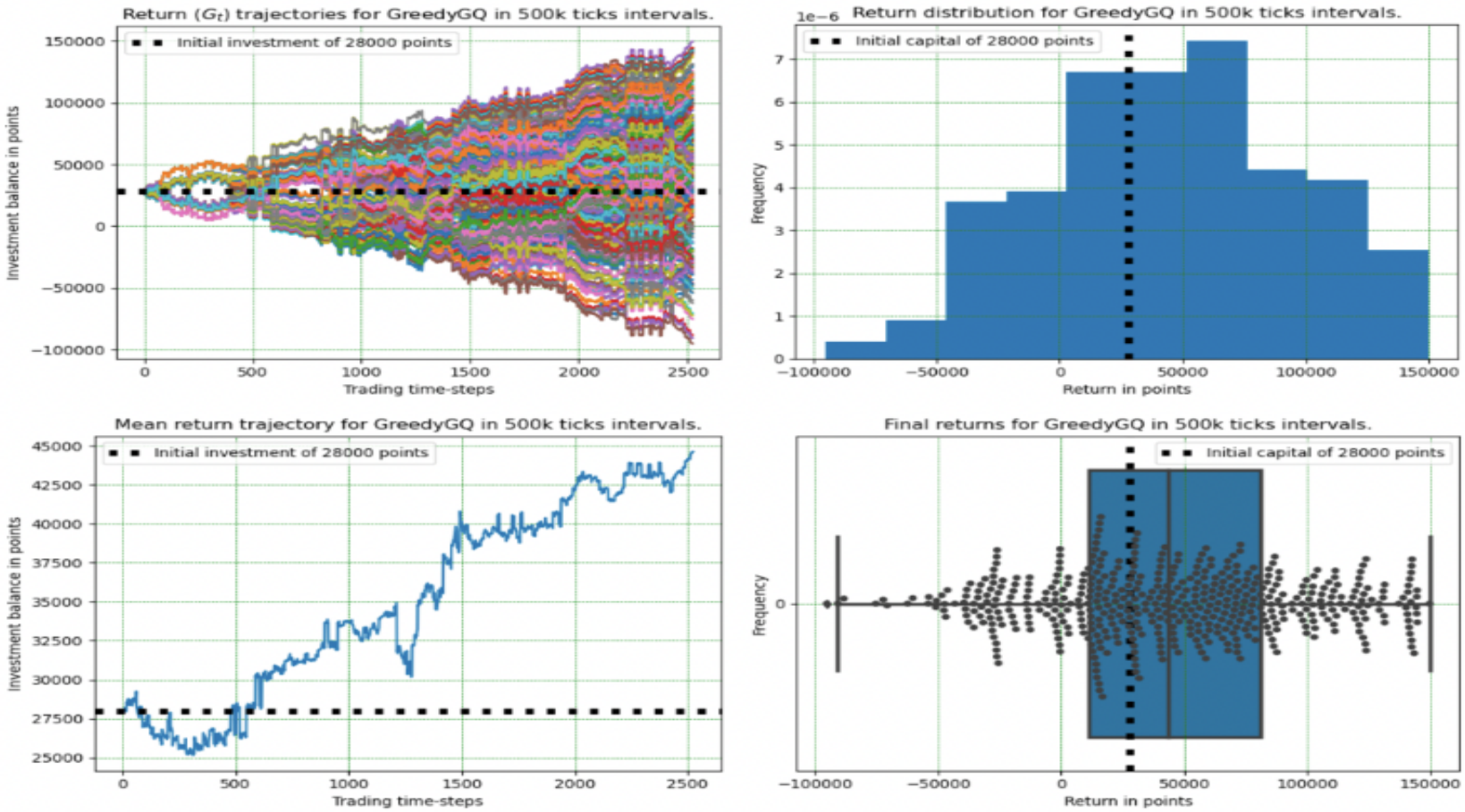


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 88.

Q-Learning: 500k–Tick

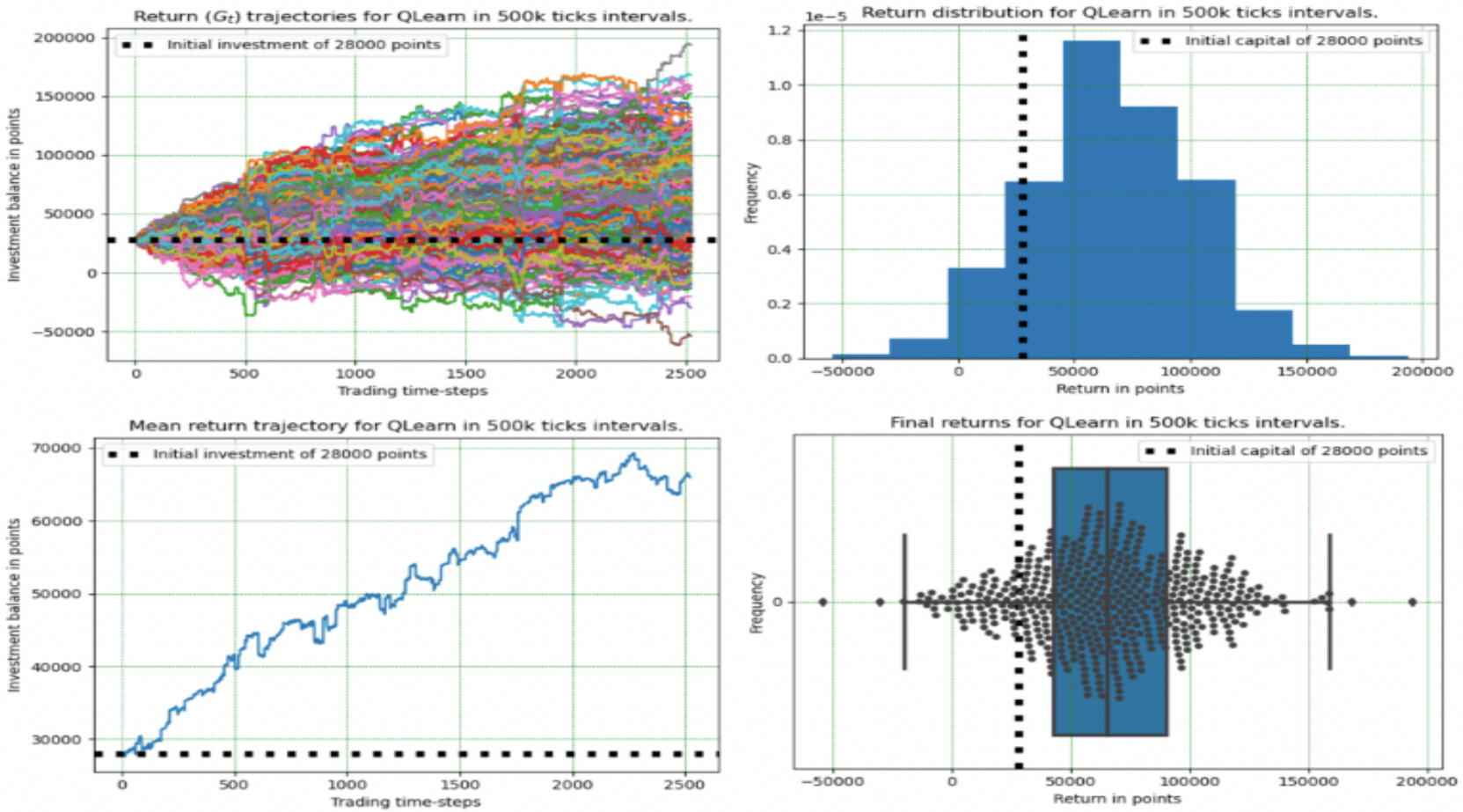


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 89.

SARSA: 500k–Tick

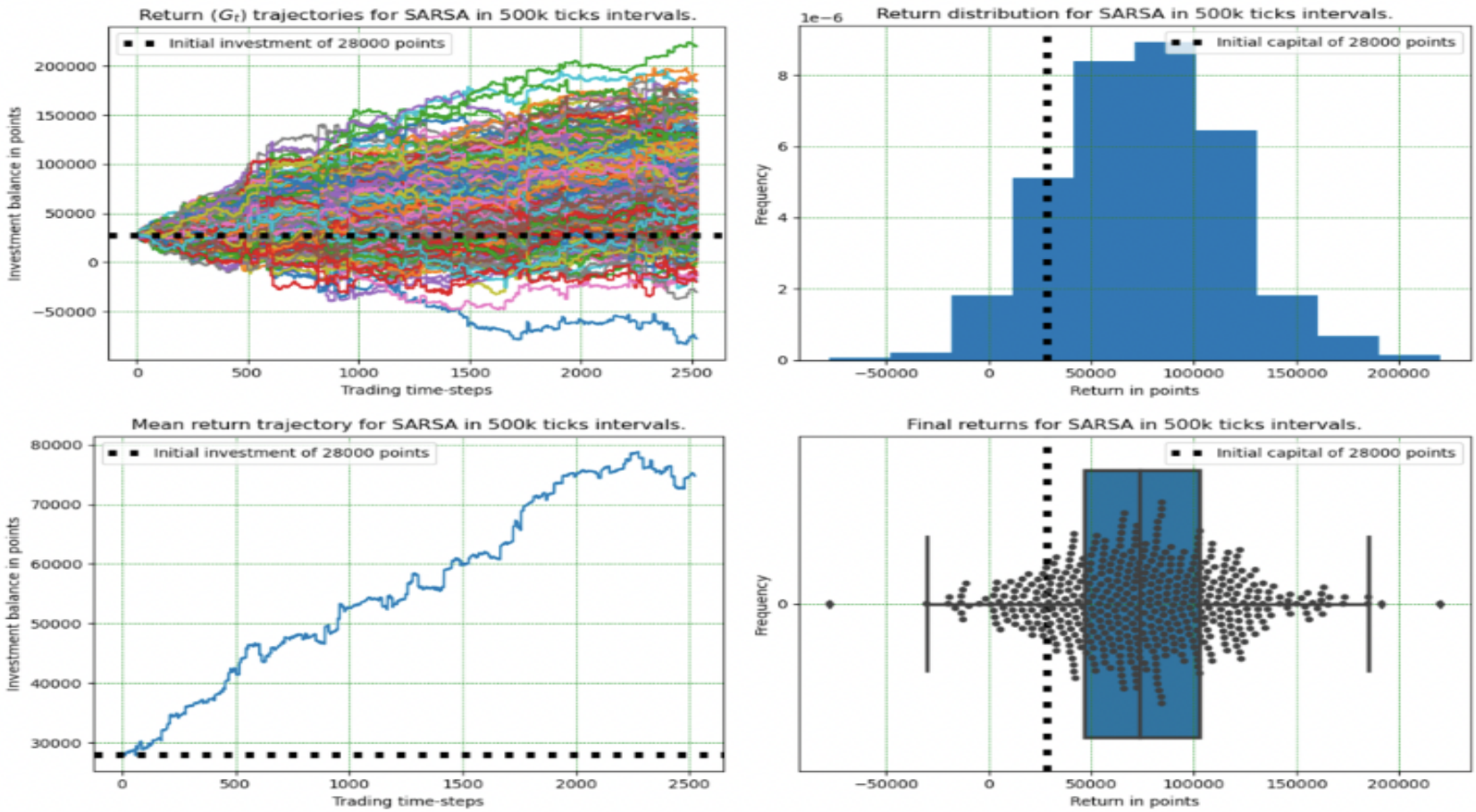


Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 89.

All Together

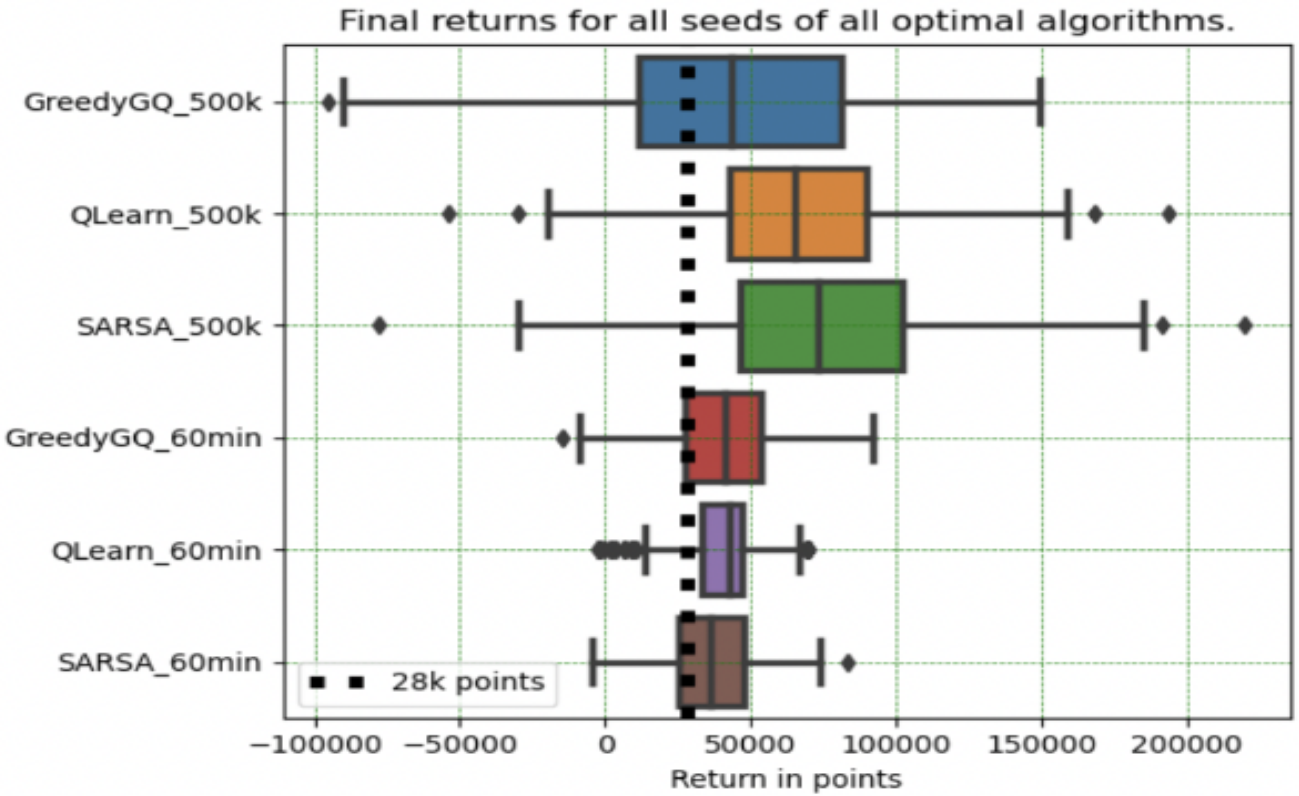


Figure 6.9: All optimal methods together.

Figure: Pereira, Fabio R. *Trading Financial Markets using Reinforcement Learning: Application and Analysis*. Pg. 92.

Conclusion

Conclusion

CAN WE HAVE CONSISTENT PROFITS USING REINFORCEMENT LEARNING IN THE FINANCIAL MARKET?

YES!

References

References I


Richard Sutton and Andrew Barto
Reinforcement Learning. An introduction.
The MIT Press, 2019.


Abhijit Gosavi
Simulation–Based Optimization.
Springer, 2015.


Marcos López de Prado
Advances in Financial Machine Learning.
Wiley, 2018.


Geramifard, Alborz and Doshi, Finale and Redding, Josh and Roy, Nicholas and How, Jonathan
'Online Discovery of Feature Dependencies.'.
Proceedings of the 28th International Conference on Machine Learning, ICML, 881–888, 2011.

References II

Geramifard, Alborz and Walsh, Thomas and Roy, Nicholas and How, Jonathan
'Batch–iFDD for representation expansion in large MDPs'.
Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI, 2013.

Herbert Robbins and Sutton Monro
'A Stochastic Approximation Method'.
The Annals of Mathematical Statistics, 3:400–407, 1951.

UiO : Department of Mathematics
University of Oslo

Fábio Rodrigues Pereira

Trading Financial Markets Using Reinforcement Learning

Application and Analysis