Machine Learning For Finance

An application of deep reinforcement learning to algorithmic trading

Anas ESSOUNAINI Taha-Abderrahman EL OUAHABI

April 19, 2021

Agenda

- Algorithmic trading : a brief overview
- Reinforcement Learning framework
- Algorithmic trading : A reinforcement learning perspective
- Deep reinforcement Learning
- Empirical results
- Conclusion

Algorithmic trading

A brief overview

Trading mechanism

- A trader interacts sequentially with the stock marke tthrough an order book containing a set of bids and asks given information to which he has access.
- More formally, the trading activity can be described as follows :
  1. Update available market information i_t
  2. Execute trading policy $\pi(a_t|i_t)$ to get action a_t
  3. Perform a_t
  4. $t \rightarrow t+1$, and loop back to step 1

Algorithmic trading

Algorithmic trading can be defined as the process of making trading decisions in an automatic manner.

Algorithmic trading

Setup

Experimental setup

In this study,

- We present empirical results applied to the stock market[a].
- Algorithmic trading problem is equivalent to managing a single stock portfolio. And, we write:

$$\underbrace{v_t}_{\text{value of the trading portfolio}} = \underbrace{v_t^c}_{\text{cash component}} + \underbrace{v_t^s}_{\text{stock component}} \quad (1)$$

- We consider a discretized trading activity on a *daily* basis with *only one decision per day*.

[a] such approach can be easily used for other classes of assets such as FOREX, commodity futures, cryptocurrencies ... etc

Reinforcement Learning

Paradigm (1/2)

Framework

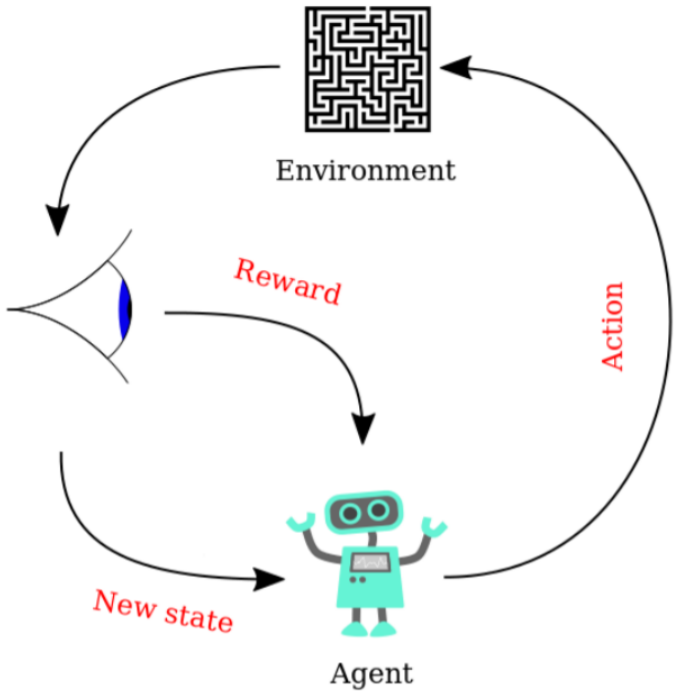
- Reinforcement learning can be described as learning:
  - by *"trial and error"*
  - to behave in an unknown, shochastic environement by maximizing some real-valued *reward* signal
- Learning process can be described as follows :
  1. A learning agent sequentially interacts with its environment by performing actions
  2. Each action provides an instantaneous reward
  3. Learning agent evolves to new state

Reinforcement Learning

Paradigm (2/2)

Agent's goal is to find a : proper strategy of choosing an action in a given state, called *policy* that maximizes the cumulative reward, *value function* in each state.

Figure: Reinforcement Learning paradigm



Reinforcement Learning

Modeling

Key Questions

1. How can we model mathematically a reinforcement learning agent ?
2. How can we quantify a "good policy" ?

Reinforcement Learning

Markov Decision Process

The Reinforcement Learning modeling paradigm is based on *Markov Decision Processes* defined as follows :

Definition (Markov Decision Process - MDP)

A MDP is parameterized by a tuple $(\mathcal{S}, \mathcal{A}, R, P)$ where

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu(s,a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu(s,a) \in \Delta(\mathbf{R})$ is the reward distribution for the state-action pair (s,a)
- $P = (p(\cdot|s,a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s,a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair (s,a)

Reinforcement Learning

Markov property

Definition (Markov Decision Process - MDP)

In a MDP, the sequence of sucessive states / actions / rewards :

$$\boxed{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{t1}, a_{t-1}, r_{t-1}, s_t}$$

satisfies some extension of the Markov property :

$$\mathbf{P}(s_t = s, r_{t-1} = r | s_0, a_0, r_0, s_1, a_1, r_1, \ldots, a_{t-1}, r_{t-1}, s_{t-1}) = \mathbf{P}(s_t = s, r_{t-1} = r | s_{t-1}, a_{t-1}) \quad (2)$$

Reinforcement Learning

Policies

Definition (Policies)

A (Markovian) policy is a sequence $\pi = (\pi_t)_{t \in \mathbf{N}}$ of mappings :

$$\pi_t : \mathcal{S} \mapsto \Delta(\mathcal{A})$$

where $\Delta(\mathcal{A})$ is the set of probability distributions over the action space.

Remark.

1. A policy may be deterministic or stochastic
2. **Terminology** : policy = strategy = decision rule = control
3. A good policy is a policy that yield a large value in each state,which is always some notion of cumulative reward, called value function.

Reinforcement Learning

Value function, Q-function for infinite horizon setup

Definition (Infinite time horizon with a discount parameter)

Given a discount factor $\gamma \in (0, 1)$, we define :

- *Value function :*

$$V^{\pi}(s) = \mathbf{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \tag{3}$$

- *Q function :*

$$Q^{\pi}(s, a) = \mathbf{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a] \tag{4}$$

Reinforcement Learning

Optimal policy

Definition (Optimal policy)

An optimal policy satisfies :

$$\pi^* \in \underset{\pi}{\operatorname{argmax}} V^{\pi} \tag{5}$$

And in this case,

$$V^* = V^{\pi^*}$$

is the optimal value function

Remark. Same definition for optimal Q-function

Algorithmic trading: a reinforcement learning perspective

Observation space $\mathcal{S}$ (1/3)

RL environment

The reinforcement learning environment is the complex trading world gravitating around the agent. The main challenge is the observability of its states, a considerable amount of information is withheld and hidden from the trading agent such as companies' confidential information and other agents' strategies. At each time step t, we consider an available public information stream i_t and the current market state s_t.

Algorithmic trading: a reinforcement learning perspective

Observation space $\mathcal{S}$ (2/3)

The state will be resumed to o_t defined as :

$$o_t = \{S_u, D_u, T_u, I_u, M_u, N_u, E_u\}_{t-\tau \leq u \leq t} \tag{6}$$

Where τ is a time window and :

- S_u: state information of the RL agent at time step u (current trading position, number of shares owned by the agent, available cash)
- D_u: information gathered by the agent at time step u concerning the OHLCV (Open-High-Low-Close-Volume) data characterising the stock market
- T_u: information regarding the trading time step u
- I_u: information regarding multiple technical indicators about the stock market targeted at time step u
- M_u: macroeconomic information at the disposal of the agent at time step u
- N_u: news information gathered by the agent at time step u
- E_u: any extra useful information at the disposal of the trading agent at time

Algorithmic trading: a reinforcement learning perspective

Observation space $\mathcal{S}$ (3/3)

Observation space reduction

$$o_t = \{\{ \overbrace{p_u^O}^{\text{opening price}} , \overbrace{p_u^H}^{\text{highest price}} , \overbrace{p_u^L}^{\text{lowest price}} , \underbrace{p_u^C}_{\text{closing price}} , \underbrace{V_u}_{\text{volume}} \}_{t-\tau \leq u \leq t}, \underbrace{P_t}_{\text{position}} \} \quad (7)$$

Action space $\mathcal{A}$

At each step t, the trading agent executes an action $a_t \in \mathcal{A}$ resulting from policy $\pi(a_t|o_t)$. The trader (agent) needs to answer to the following questions: How, whether and how much ?
Actions will be modelled with $Q_t \in \mathbf{Z}$, quantity of shares to buy, *i.e* :

$$a_t = Q_t \tag{8}$$

We consider the following cases:

- $Q_t > 0$: agent buys by posting new bid orders on the order book
- $Q_t < 0$: agent sells by posting new ask orders on the order book
- $Q_t = 0$: agent holds

Algorithmic trading: a reinforcement learning perspective

Key assumptions

Assumptions

1. Trading actions occur at market closure with execution price $p_t \approx p_t^c$.
2. We need to account for the trading costs for a realistic and accurate setup. They are due to:
   - costs and taxes
   - slippage costs (spread costs, market impact and timing costs)

Remark. If the first type is easy to model, the second one poses a true challenge for modelling. For the sake of simplicity, we make the hypothesis that costs are proportional to asset's price.

Algorithmic trading: a reinforcement learning perspective

Trading portfolio (1/2)

Portfolio update

$$v^c_{t+1} = v^c_t - Q_t p_t - \overbrace{C}^{\text{cost proportionality constant}} |Q_t| p_t \tag{9}$$

$$v^s_{t+1} = \underbrace{(n_t + Q_t)}_{n_{t+1}} p_{t+1} \tag{10}$$

where $n_t \in \mathcal{Z}$ is the number of shares owned at time step t. Note that negative values are allowed, this corresponds to shares borrowed and sold with obligation to repay the lender in the future.

Algorithmic trading: a reinforcement learning perspective

Trading portfolio (2/2)

Constraints

- We need the cash value v_t^{cash} to remain positive:

$$v_t^c \geq 0 \tag{11}$$

- We need to adress the risk associated with the impossibility to repay the share lender if agent suffers significant losses. Consequently, v_t^c must be sufficiently large when a negative number of shares is owned. The agent must be able to get back to a neutral position. In a formal manner,

$$v_{t+1}^c \geq -n_{t+1} p_t (1+\epsilon)(1+C) \tag{12}$$

where ϵ quantifies maximum relative change in prices assumed by RL agent prior to trading activity with following condition condition assumed to be fulfilled:

$$|\frac{p_{t+1} - p_t}{p_t}| \leq \epsilon$$

Algorithmic trading: a reinforcement learning perspective

Action space bounds

Upper and lower bounds for action space

(11) and (12) yield upper and lower bounds and we write :

$$\mathcal{A} = \{Q_t \in \mathbf{Z} \cap [\underline{Q_t}, \overline{Q_t}]\} \tag{13}$$

Where:

- $\underline{Q_t} = \frac{v_t^c}{p_t(1+C)}$
- $\overline{Q_t} = \begin{cases} \frac{\Delta_t}{p_t(1+C)\epsilon} & , \Delta_t < 0 \\ \frac{\Delta_t}{p_t(2C+\epsilon(1+C))} & , \Delta_t \geq 0 \end{cases}$

Where $\Delta_t = -v_t^c - n_t p_t(1+\epsilon)(1+C)$

Algorithmic trading: a reinforcement learning perspective

Action space reduction

We reduce $\mathcal{A}$ to:

$$a_t = Q_t \in \mathcal{A} = \{Q_t^{long}, Q_t^{short}\} \tag{14}$$

for feasibility purposes.

- Long action:
  Q_t^{long} induces a long trading position with $n_t \geq 0$. It converts maximum of $v_t^c \rightarrow v_t^s$. And we write:

$$Q_t^{long} = \begin{cases} \frac{v_t}{p_t(1+C)} & , a_{t-1} \neq Q_{t-1}^{long} \\ 0 & , otherwise \end{cases} \tag{15}$$

- Short action:
  Q_t^{short} converts $v_t^s \rightarrow v_t^c$ such that RL agent owns a number of shares equal to $-N_t^{long}$. And we write:

$$\hat{Q}_t^{short} = \begin{cases} -2n_t - \frac{v_t}{p_t(1+C)} & , a_{t-1} \neq Q_{t-1}^{short} \\ 0 & , otherwise \end{cases} \tag{16}$$

  However, Q_t^{short} needs to respect lower bound of $\mathcal{A}$. Hence,

$$Q_t^{short} = max(\hat{Q}_t^{short}, \underline{Q_t}) \tag{17}$$

Agent's reward

We set the RL agent's reward to be the daily return of the portfolio :

$$r_t = \frac{v_{t+1} - v_t}{v_t} \tag{18}$$

There are several reasons to justify this choice:

1. This reward is independent from the number of held shares
2. $r_t \geq 0 \implies$ profitable strategy
3. (18) enables us to avoid sparsity that is very complex to adress in a Reinforcement Learning setup

Intro to deep reinforcement learning

Elements in control theory (1/2)

Definition (Dynamic Programming operator)

The optimal Bellman operator is defined as follows:

1. Value function :

$$T^* : \mathbf{R}^{|\mathcal{S}|} \longrightarrow \mathbf{R}^{|\mathcal{S}|}$$
$$V \longmapsto T^*(V)$$

Where $T^*(V)(s) = \max_a \{r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s')\}$

2. Q function :

$$T^* : \mathbf{R}^{|\mathcal{S}|} \times \mathbf{R}^{|\mathcal{A}|} \longrightarrow \mathbf{R}^{|\mathcal{S}|} \times \mathbf{R}^{|\mathcal{A}|}$$
$$Q \longmapsto T^*(Q)$$

Where $T^*(Q)(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_b Q(s',b)$

Intro to deep reinforcement learning

Elements in control theory (2/2)

Theorem

Q^* and V^* are fixed points for their respective dynamic programming operators.

Remark. In a finite horizon setup, this theorem is replaced with a backward linear system.

Temporal difference algorithms

1. TD (temporal difference) learning is a key pillar in reinforcement learning.
2. **IDEA**: find the solution of the fixed point problem for the optimal Value function and Q-function given their noisy evaluation with Stochastic Approximation (**Robbins-Monro algorithm**).
3. Main algorithms: TD(λ), Q-learning, SARSA ...etc

Temporal difference update rule

The TD update rule for :

1. Value function :

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \tag{19}$$

2. Q-function :

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{20}$$

Where α is the learning rate.

Intro to deep reinforcement learning

Q-learning

Q-learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small *epsilon* > 0 among other learning and environment parameters

- Initialize $Q(s, a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

foreach *episode* **do**

  - Initialize S **foreach** *step of episode* **do**

    - Choose A from S using policy derived from Q (e.g., ε-greedy)
    - Take action A, observe R, S'
    - $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ - $S \leftarrow S'$

Intro to deep reinforcement learning

Reinforcement Learning with approximation: case of Deep Q-learning

Two problems

1. How to handle large state/action spaces in memory?
2. How to generalise over state/action spaces to learn faster?

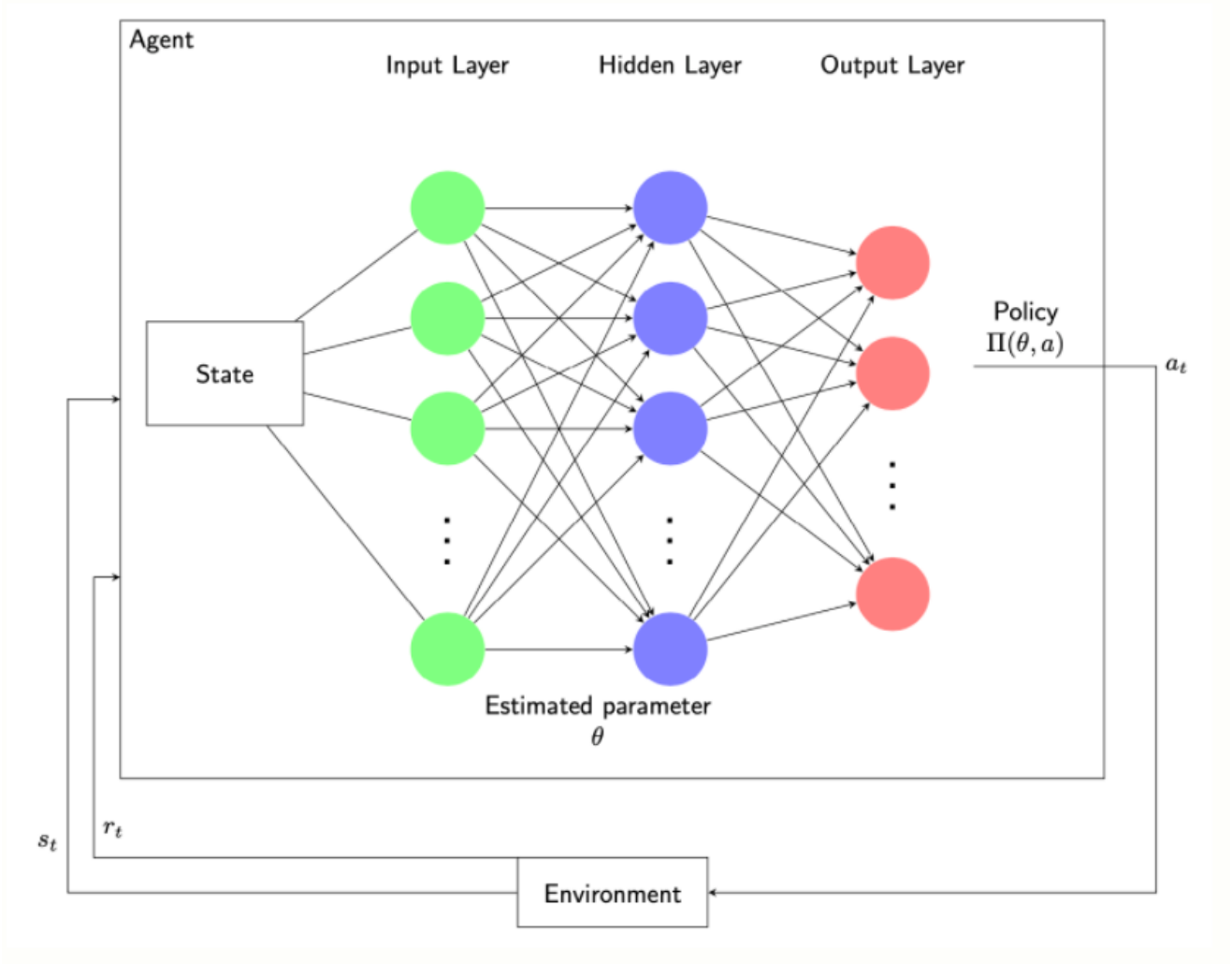
Solution

- Use a parametric set of functions to approximate Q : $(Q_\theta)_\theta$
- Case of Deep Q-learning
  - $(Q_\theta)_\theta \sim$ Neural Nets
  - Neural nets are universal approximators

Intro to deep reinforcement learning

Reinforcement Learning with approximation: case of Deep Q-learning

Figure: Deep Reinforcement Learning



Intro to deep reinforcement learning

Reinforcement Learning with approximation: case of Deep Q-learning

Deep Q-learning

1 Algorithm parameters: step size $\alpha \in (0, 1]$, discount factor $\gamma \in (0, 1)$ among other learning and environment parameters

2 - Use two neural networks:

1. A slow-learning target network (θ^-)
2. A fast learning Q-network (θ)

- Use experience replay (fill in a replay buffer D with transitions generated by $\pi = f(Q_{\theta^-}(s, a))$
- Shuffle samples in the replay buffer and minimize:

$$J(\theta) = \sum_{(s,a,r,s') \in D} [r + \gamma max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)]^2$$

$$\theta \leftarrow \theta + \alpha(r + \gamma max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)) \nabla_\theta Q_\theta(s, a)$$

- Every N training steps $\theta^- \leftarrow \theta$

Some improvments Dueling, regularization, exploration with decay, processing state/action pair ...etc

Empirical results

Back Testing approach

- Here we explain the Backtesting method
- The initial approach consisted of:
  - Taking into account a large universe of stocks explaining a significant portion of market heterogeneity
  - Train TDQN on each stock
  - Back test and get performance metrics for each stock
- This is not feasible because of training time
- In Google Colab, each stock required -on average- 90 minutes of training

Empirical results

Alternative Back Testing approach

- We chose to the agent for one stock
- This stock should exhibit multiple phases: Bullish phase, Bearish
- Microsoft stock between 2012-1-26 and 2018-12-26 was chosen

Figure: Training environment

Figure: Backtesting environment

Empirical results

Technical Indicators

- Under some circumstances, passive strategies do not provide an unbiased benchmark of the performance
  1. Case of a positive trend
  2. Case of a negative trend
- Technical Indicators are widely used by traders (comparing it to a reinforcement learning is an important step)
  $\Longrightarrow$ The use of technicals would help us better understand the performance of the agent
- RSI (Momentum), Bollinger Bands (Volatility), SMA and MACD (Trend) were used

Empirical results

Technical Indicators: RSI

Relative Strength Index: RSI evaluates overbought and oversold stocks.
Long signal: RSI crosses down 30
Short signal: RSI value crosses up 70

Figure: RSI use

Empirical results

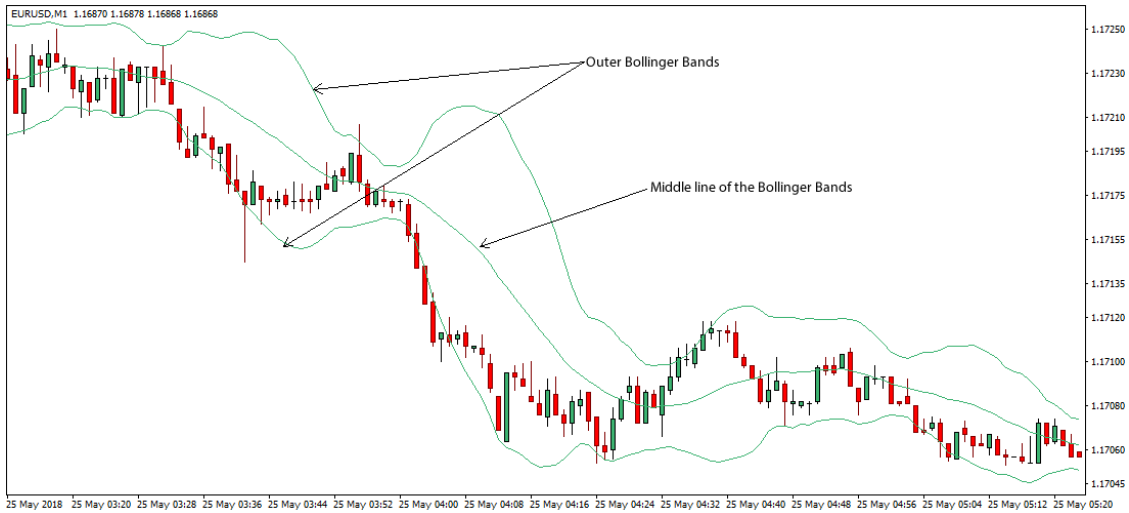
Technical Indicators: Bollinger Bands

Relative Strength Index: Compares current price levels to historic extreme levels

Long signal: Price crosses down lower band

Short signal: Price crosses up upper band

Figure: Bollinger Bands use



Empirical results

Technical Indicators: MACD and SMA

MACD: Provides an indication of the stock strength, direction and trend
Long signal: Indicator changes sign from negative to positive
Short signal: Indicator changes sign from positive to negative

Figure: MACD cross over strategy

Empirical results

Technical Indicators: wrap up

Technical Indicator	Definition	Long Signal	Short Signal
RSI	Evaluates overbought and oversold stocks	Crosses down 30	Crosses up 70
MACD	Provides an indication of the stock strength, direction and trend	Crosses up 0	Crosses down 0
Bollinger Bands	Compares current price levels to historic extreme levels	Price crosses down lower band	Price crosses up upper band
SMA	Provides an indication of the stock strength, direction and trend	Crosses up 0	Crosses down 0

Empirical results

Comparison

	Trading Strategy						
Metric	B&H	S&H	RSI	BBANDS	MACD	SMA	TDQN
Sharpe Ratio (Higher is better)	0.21	0.07	NaN	1.3	0.17	0.81	0.79
Sortino Ratio (Higher is better)	0.29	0.11	NaN	2.01	0.23	1.17	1.2
Max Drawdown	-0.18	-0.27	0	-0.09	-0.12	-0.12	-0.15
Profit Ratio	0.71	1.39	NaN	0.03	0.07	0.55	0.57
Annualized Retruns (Higher is better)	6%	%	0%	27%	3%	22%	20%
Annualized Volatility (Lower is better)	28%	%	0%	21%	15%	27%	26%

Empirical results

Comparison

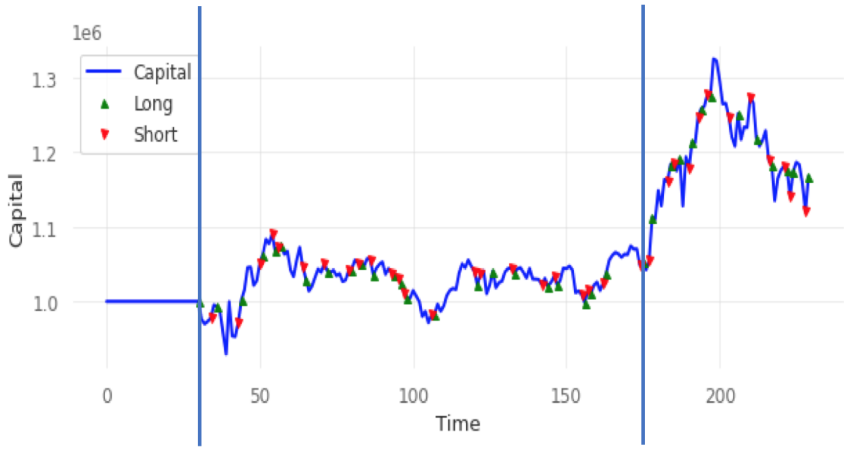


Figure: Agent Performance

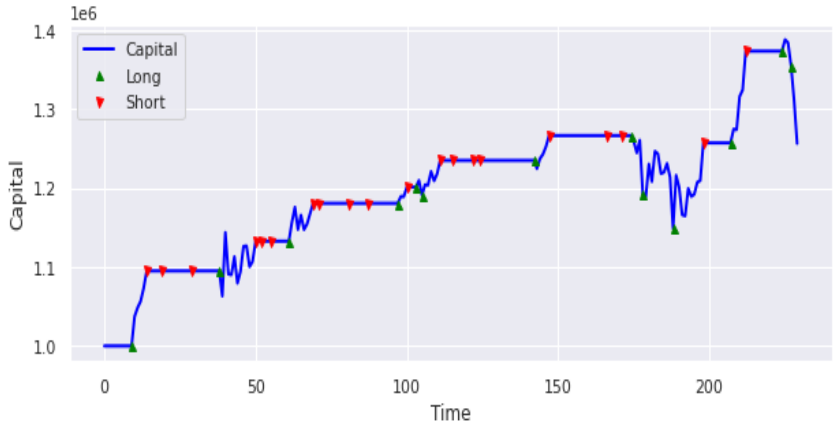


Figure: Bollinger Bands performance

Empirical results

Effects of transaction costs

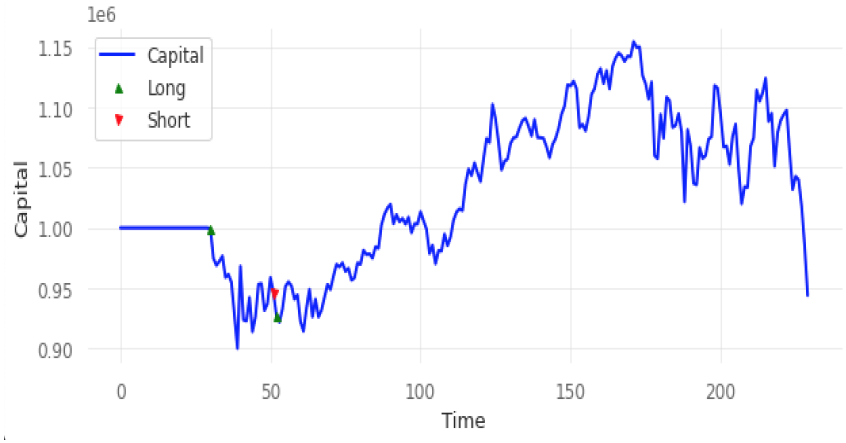


Figure: Agent Performance under 0.1% transaction costs

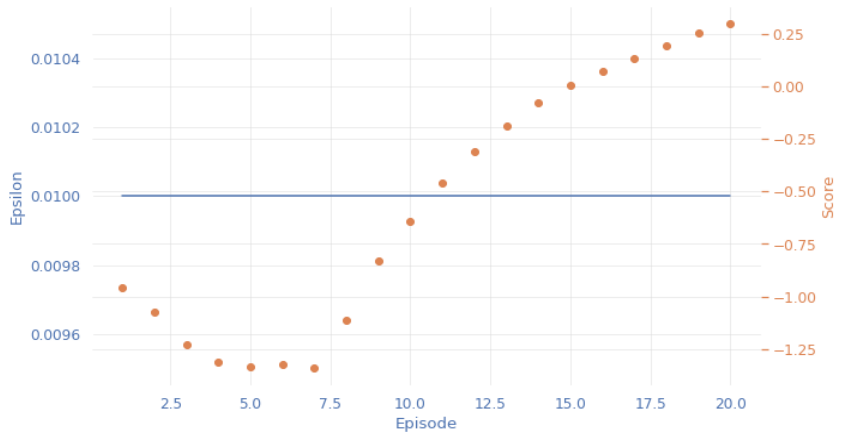


Figure: Evolution of score as function of episodes

Conclusion

To sum up

- A promising potentialfor the Trading Deep Q-Network
- Important dependence of the agent on the underlying dynamics of the environment
- The measurement of the TDQN performance could beimproved.

References

1. Théate, T., Ernst, D. (2021).
   - An application of deep reinforcement learning to algorithmic trading
   Expert Systems with Applications, 173, 114632

2. Kaufmann,E. (INRIA Lille) , Valko,M. (INRIA Lille & Deepmind)
   - Lecture notes on reinforcement learning
   Ecole Centrale de Lille

3. Michel Fliess (LIX, INRIA Saclay - Ile de France), Cédric Join (INRIA Saclay - Ile de France, CRAN)
   - A mathematical proof of the existence of trends in financial time series
   (arXiv:0901.1945).

Thanks !