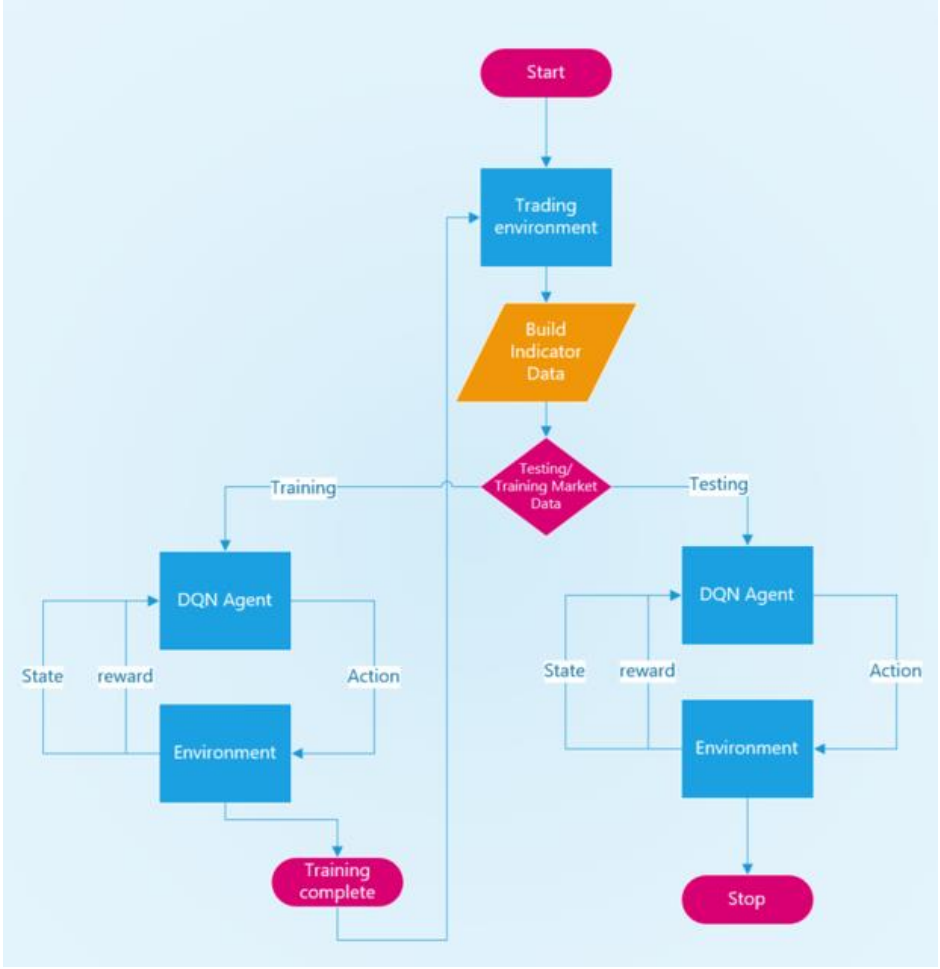
Some Applications of RL in Quantitative Trading

Weizhi Zhu

Trading with Deep Reinforcement Learning



State Vector

- $[ADX(t), RSI(t), CCI(t), position, unrealized\ return]$

ADX(Average directional movement index) is a trend strength indicator.

RSI(Relative Strength Index) is classified as a momentum oscillator, measuring the velocity and magnitude of directional price movements.

CCI(Commodity Channel Index) measures a security's variation from the statistical mean.

- could use a LSTM to extract more features.

Action

- The agent could take three actions – Buy, Sell or Hold

- Could set levels of buy/sell.
- Could involve a pair of stocks.

Reward

- The reward objective is set to maximize realized PnL from a round trip trade. It also includes commission fee.

Learning Algorithms

- DQN
- Double DQN
- Dueling DQN
- Actor Critic
- PPO
- DDPG

DQN

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For t = 1,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

Double DQN

Here is an example: consider a single state s where the true Q value for all actions equal 0, but the estimated Q values are distributed some above and below zero. Taking the maximum of these estimates (which is obviously bigger than zero) to update the Q function leads to the overestimation of Q values.

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$

for each iteration **do**

  for each environment step **do**

    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$

    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$

    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$

  for each update step **do**

    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$

    Compute target Q value:

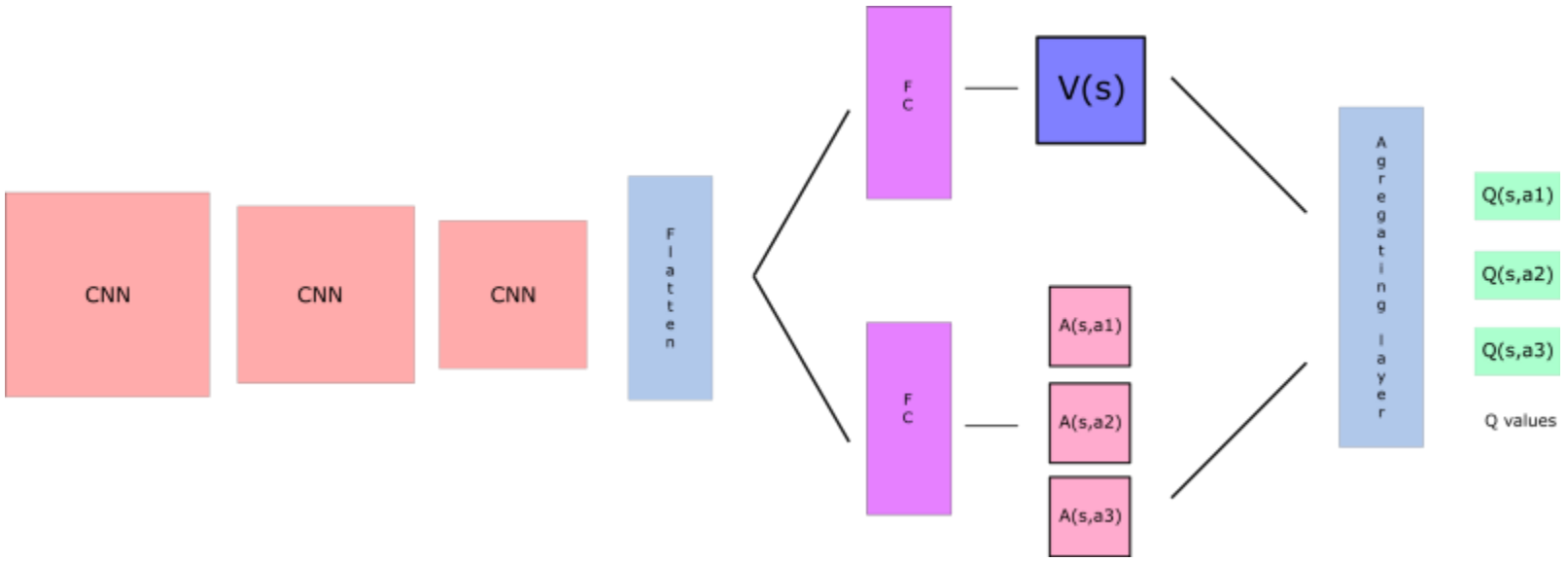
      $Q^*(s_t, a_t) \approx r_t + \gamma\ Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$

    Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$

    Update target network parameters:

      $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$

Dueling DQN



Motivation: it is unnecessary to know the value of each action at every timestep. The authors give an example of the Atari game Enduro, where it is not necessary to know which action to take until collision is imminent.

Dueling DQN

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha) + A(s,a;\theta,\beta)$$
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha) + A(s,a;\theta,\beta) - \max_{a\in\mathcal{A}} A(s,a;\theta,\alpha,\beta)$$

The first one have problem of identifiability.
The second one would force Q value on selected action equal to the V value.

Policy Gradient

$$J(\pi_\theta) = E_{\tau \sim \pi_\theta}[R(\tau)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Phi_t\right]$$

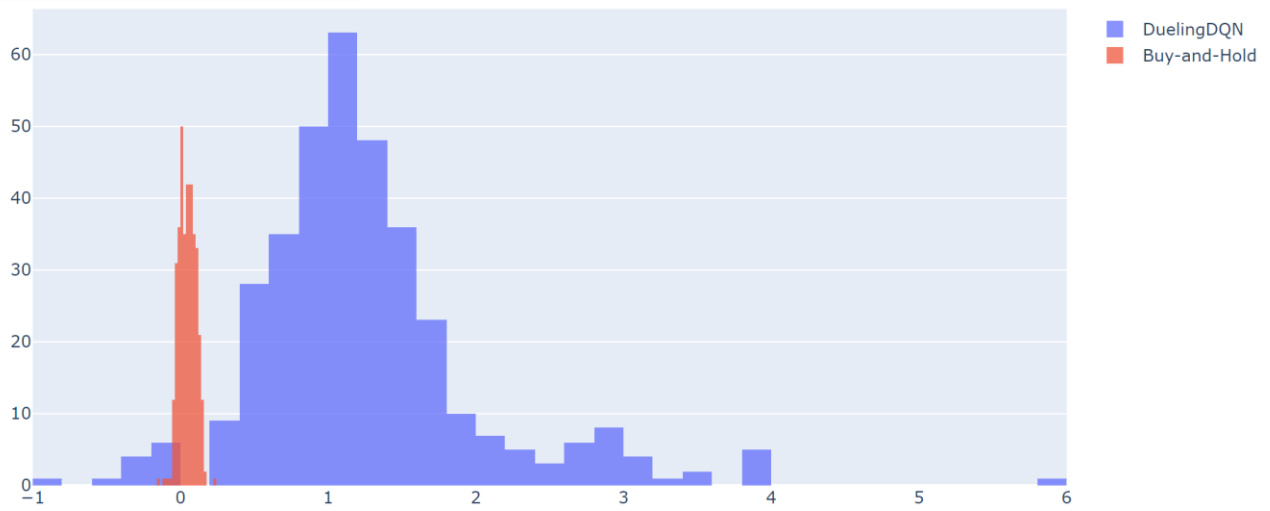
where different choices of Φ_t leads to different variants of policy algorithm.
In particular, $\Phi_t = Q^\pi(s_t, a_t) - V^\pi(s_t)$ leads to Advantage Actor Critic.

Advantage Actor Critic (A2C)

Input a differentiable policy parameterization $\pi(a \mid s, \theta_\pi)$
Input a differentiable state-value parameterization $\hat{v}(s, \theta_{\hat{v}})$
Select step-size parameters $0 < \alpha_\pi, \alpha_{\hat{v}} \leq 1$
Initialize the parameters θ_{pi}, $\theta_{\hat{v}}$
Loop through n episodes (or forever):
    Begin the episode s_0
    Continue to loop until the episode ends:
        Get action A_t from π: $\pi(S_t, \theta_\pi) \rightarrow A_t$.
            Take action A_t and observe reward (R_t) and the new state (S_{t+1})
            Calculate the TD target: $G_t \leftarrow R_t + \gamma \hat{v}(S_{t+1}, \theta_{\hat{v}})$
            Calculate the TD error: $\delta_t \leftarrow R_t + \gamma \hat{v}(S_{t+1}, \theta_{\hat{v}}) - \hat{v}(S_t, \theta_{\hat{v}})$
            Calculate the critic loss $L(\theta_{\hat{v}}) =< br > \frac{1}{T} \sum_{t=1}^{T} \left(\hat{v}(S_t, \theta_{\hat{v}}) - G_t\right)^2$
            Calculate the actor loss $L(\theta_\pi) = -\sum_{t=1}^{T} ln\big(\pi(A_t \mid S_t, \theta_{pi})\big)\delta_t$
            Update actor parameters through backpropagation: $\theta_\pi := \theta_\pi + \alpha_\pi \nabla_\pi L(\theta_\pi)$
            Update critic parameters through backpropagation: $\theta_{\hat{v}} := \theta_{\hat{v}} + \alpha_{\hat{v}} \nabla_{\hat{v}} L(\theta_{\hat{v}})$

Results

	Symbols	Buy and Hold Sharpe	Strategy Sharpe	Number of Trades	Buy and Hold Total Return	Strategy Total Return
0	COTY	-0.001793	0.522897	24.0	-0.010763	0.425965
1	CMS	0.008286	1.647612	21.0	0.021951	0.332393
2	CNC	0.082073	0.643396	4.0	0.542984	0.296637
3	CSRA	-0.022823	0.502373	7.0	-0.042829	0.056752
4	CSCO	0.084183	0.830318	13.0	0.283655	0.222759



Market Making

- Provide liquidity by quoting both bid and ask. Make profit from bid-ask spread.
- Mathematically, the market making problem corresponds to the choice of optimal quotes (i.e. the bid and ask prices) that such agents provide to other market participants, taking into account their inventory limits and their risk constraints often represented by a utility function.

Mathematical Formulation

• Mid price follows $dS_t = \sigma dW_t$. bid, ask prices quoted by market maker are denoted by S_t^a, S_t^b respectively
• Inventory $q_t = N_t^b - N_t^a$, where N_t^b and N_t^a are point processes giving the number of shares the market maker bought and sold.
•. Assume intensity λ^b, λ^a associated to N^b, N^a depend on the difference between quote and mid price.

$$\lambda^b(\delta^b) = Ae^{-k\delta^b}, \delta^b = s - s^b; \lambda^a(\delta^a) = Ae^{-k\delta^a}, \delta^b = s^a - s.$$

where A and k are parameters characterizing the liquidity of market.
• As a consequence of his trade, market maker has an amount of cash evolving according to the following dynamics,

$$dX_t = (S_t + \delta_t^a)dN_t^a - (S_t - \delta_t^b)dN_t^b.$$

Goal

Find optimal (state dependent) δ_t^b, δ_t^a that maximize CARA utility.

$$u(s,x,q,t) = \sup_{(\delta_t^a)_t,(\delta_t^b)_t} \mathbb{E}\{-\exp(\gamma(X_T + q_T S_T))\}$$

Solving u and optimal $(\delta_t^{a,b})_t$ boils down to Hamilton-Jacobi-Bellman equation. For example two-steps procedure is introduced in Marco and Sasha (2008).

$$\begin{cases} u_t + \frac{1}{2}\sigma^2 u_{ss} + \max_{\delta^b} \lambda^b(\delta^b)\big[u(s, x-s+\delta^b, q+1, t) \\ \quad - u(s, x, q, t)\big] + \max_{\delta^a} \lambda^a(\delta^a)\big[u(s, x+s+\delta^a, q-1, t) \\ \quad - u(s, x, q, t)\big] = 0, \\ u(s, x, q, T) = -\exp(-\gamma(x+qs)). \end{cases}$$

Simulation

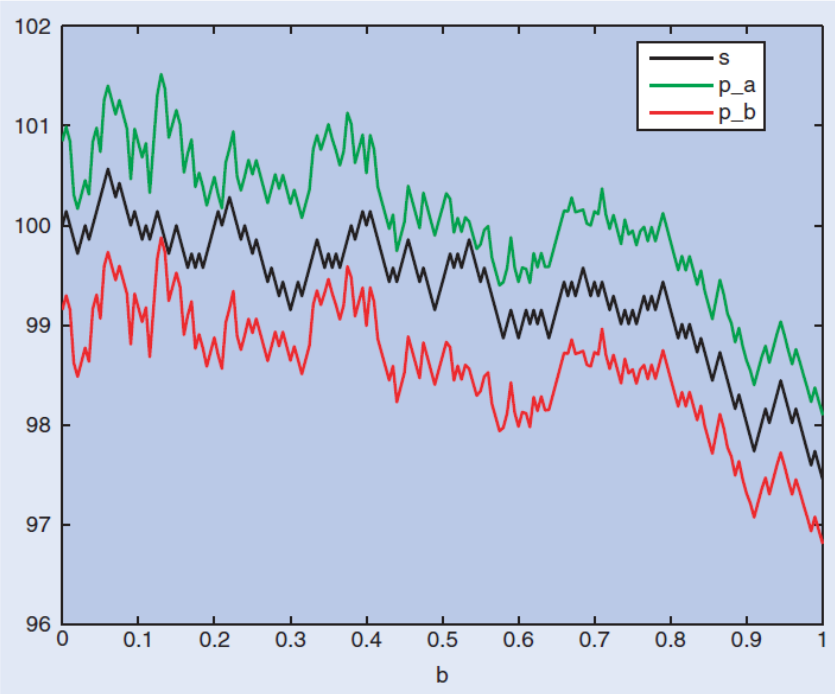
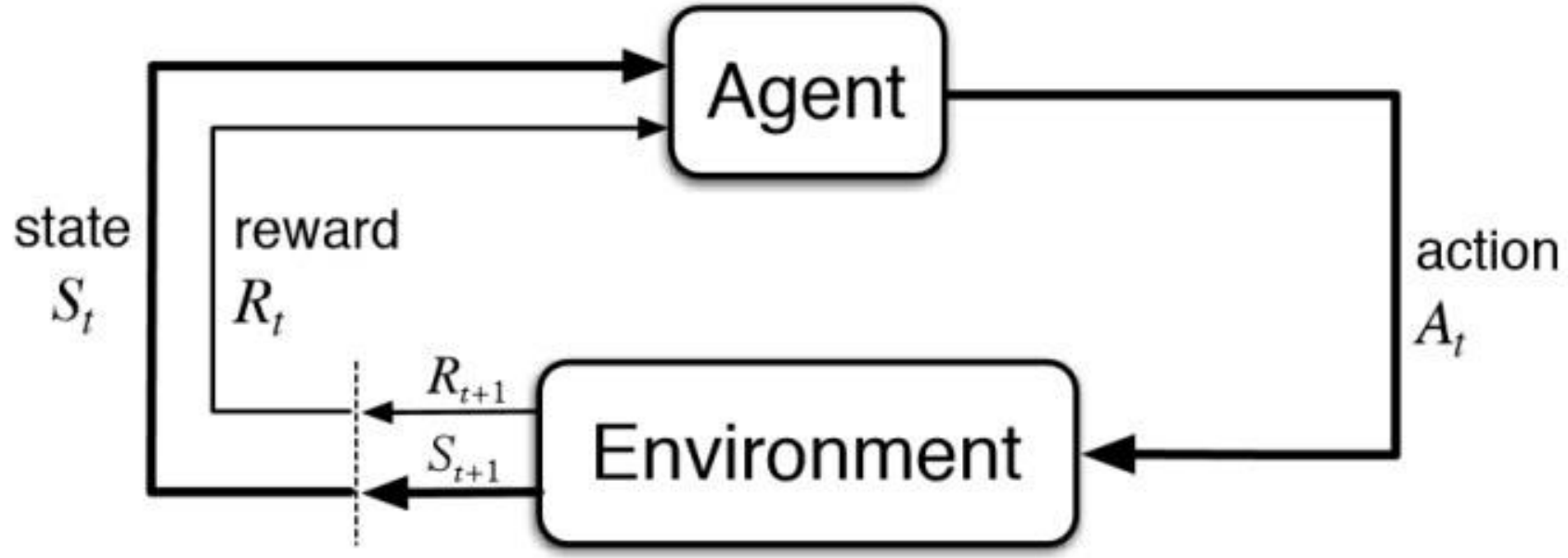


Figure 1. The mid-price and the optimal bid and ask quotes.

Table 1. 1000 simulations with $\gamma = 0.1$.

Strategy	Average spread	Profit	Std (Profit)	Final q	Std (Final q)
Inventory	1.49	65.0	6.6	0.08	2.9
Symmetric	1.49	68.4	12.7	0.26	8.4

Model free Reinforcement Learning Approach



Action Space

Action ID	0	1	2	3	4	5	6	7	8
Ask (θ_a)	1	2	3	4	5	1	3	2	5
Bid (θ_b)	1	2	3	4	5	3	1	5	2
Action 9	MO with $\text{Size}_m = -\text{Inv}(t_i)$								

$$\delta_t^a = \theta^a * MA(Spread_t)/2, S_t^a = S_t - \delta_t^a,$$
$$\delta_t^b = \theta^b * MA(Spread_t)/2, S_t^b = S_t + \delta_t^b.$$

Reward

PnL:

$$r_i = \Psi(t_i).$$

Symmetrically dampened PnL:

$$r_i = \Psi(t_i) - \eta \cdot \mathrm{Inv}(t_i)\Delta m(t_i).$$

Asymmetrically dampened PnL:

$$r_i = \Psi(t_i) - \max[0,\ \eta \cdot \mathrm{Inv}(t_i)\Delta m(t_i)].$$

- Dampening reduce the reward gained from speculation.
- Symmetric version dampens both profit and loss, Asymmetic version keep lose but reduce profit.

State

- Bid – ask spread
- Mid-price move
- Book/queue imbalance
- Signed Volume
- Volatility
- Relative strength index

Learning Algorithms

- Q-Learning, SARSA and its variants (e.g. Double Q-Learning, Expected SARSA…)

- Deep Q-Learning, Actor Critic and its variant.

Results

Table 6: Comparison of the out-of-sample normalised daily PnL (ND-PnL) and mean absolute positions (MAP) of the bench mark strategies against the final presented reinforcement learning agent.

	Abernethy and Kale (MMMW) Benchmark		Fixed ($\theta_{a,b} = 5$) Benchmark		Consolidated Agent	
	ND-PnL [10^4]	MAP [units]	ND-PnL [10^4]	MAP [units]	ND-PnL [10^4]	MAP [units]
CRDI.MI	−1.44 ± 22.78	7814 ± 1012	−0.14 ± 1.63	205 ± 351	**0.15 ± 0.59**	1 ± 2
GASI.MI	−1.86 ± 9.22	5743 ± 1333	**0.01 ± 1.36**	352 ± 523	0.00 ± 1.01	33 ± 65
GSK.L	−3.36 ± 13.75	8181 ± 1041	0.95 ± 2.86	1342 ± 1210	**7.32 ± 7.23**	57 ± 105
HSBA.L	1.66 ± 22.48	7330 ± 1059	2.80 ± 10.30	2678 ± 1981	**15.43 ± 13.01**	104 ± 179
ING.AS	−6.53 ± 41.85	7997 ± 1265	**3.44 ± 23.24**	2508 ± 1915	−3.21 ± 29.05	10 ± 20
LGEN.L	−0.03 ± 11.42	5386 ± 1297	0.84 ± 2.45	986 ± 949	**4.52 ± 8.29**	229 ± 361
LSE.L	−2.54 ± 4.50	4684 ± 1507	0.20 ± 0.63	382 ± 553	**1.83 ± 3.32**	72 ± 139
NOK1V.HE	−0.97 ± 8.20	5991 ± 1304	**−0.52 ± 4.16**	274 ± 497	−5.28 ± 33.42	31 ± 62
SAN.MC	−2.53 ± 26.51	8865 ± 671	1.52 ± 11.64	3021 ± 2194	**5.67 ± 13.41**	4 ± 9
VOD.L	1.80 ± 22.83	7283 ± 1579	1.26 ± 4.60	1906 ± 1553	**5.02 ± 6.35**	46 ± 87

Results

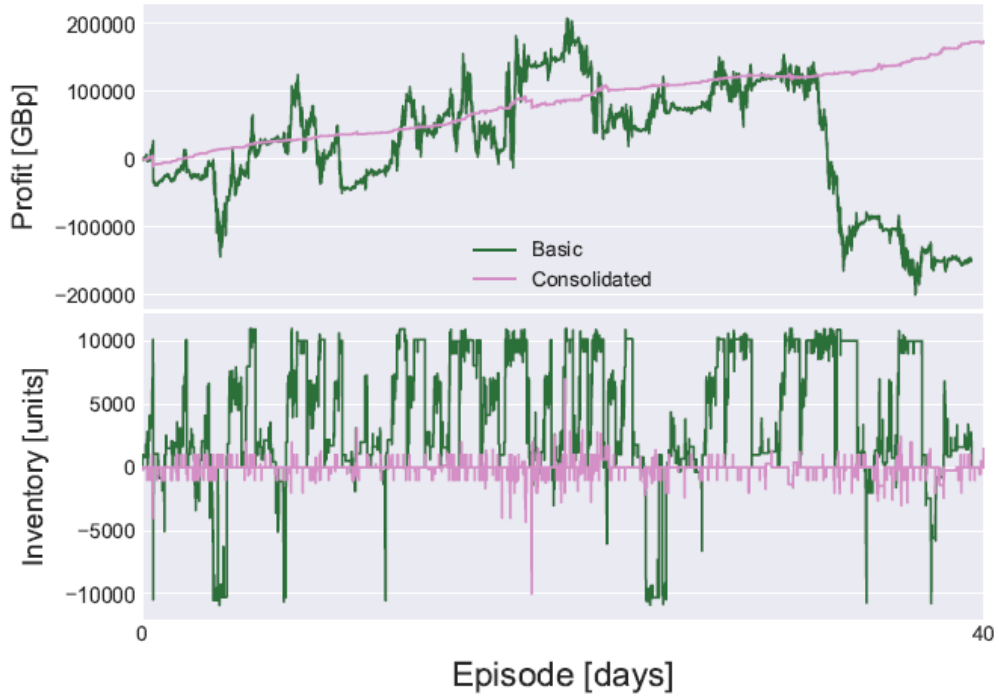


Figure 5: Out-of-sample equity curve and inventory process for the basic and consolidated agents, evaluated on HSBA.L.

Reference

- T. Beysolow II, *Applied Reinforcement Learning with Python*
- M. AVELLANEDA and S. STOIKOV, High-frequency trading in a limit order book
- Thomas and Hans, Optimal dealer pricing under transactions and return uncertainty.
- Olivier et al.. Dealing with the Inverntory Risk: A solution to the market making problem.
- Thomas et al.. Market Making via Reinforcement Learning.

Other Applications

- Deep Hedging
- Order Execution
- Forex, Crypto Currency trading bot
- …