Risk-aware Trading agent using Rainbow-IQN

Jungsub Rhim[1]
Department of Artificial Intelligence
Seoul National University
jsrimr@snu.ac.kr

Abstract— **There were several approaches to adapt Deep Reinforcement Learning(DRL) algorithms into trading. However, using RL algorithms in real-world trading invokes many concerns because of the lack of robustness. DRL agents trained in simulation environment could simply show poor performance and this could lead to huge loss. In this paper, we use several extensions of DQN and Internal Qunatile Networks to manage the risks.**

This project aims to investigate the effect of risk-aware strategies in margin trading environment. To do so, three OpenAI Gym mimicing environment is developed. Long-only, short-available, leverage-available environments. In the simulation environment, same algorithm agents with or without risk-aware mechanisms are compared. We show that variant CVaR risk measures control risk sensitivity and this risk-sensitivity could significantly diminish the loss and finally outperform the benchmark return with great margins. We exploit this risk management skills to maximize the profit using leverage and show the suggested algorithm works in real-world revealing trade results on binance.

I. INTRODUCTION

Reinforcement learning (RL) is a powerful algorithmic paradigm encompassing a wide array of contemporary algorithmic approaches. RL methods have been shown to be effective on a large set of simulated environments, and there were few attempts to adapt RL frameworks to the trading methods. However, algorithm trading is known to be easily outperformed by benchmark returns (Moon et al., 2012). This is also revealed in our experiments. In Fig.1, we checked that no matter what algorithms we use, it was hard to beat the baseline return, which is just full long at the initial time and hold it until the end. This might be caused by suboptimal hyperparameter selection or lack of useful information in data to generalize the pattern.

Nonetheless, we posit the major reason algorithms can't beat the baseline return is due in large part to algorithm lack of ability to catch some strongest price soaring trend which is hard for the algorithm to detect. This skyrocketing is extremely unique and happens in arbitrary manner, so the data is highly sparse and almost impossible to generalize. We remain this restriction to the further work, and we bypass these problems for now. What we interestingly spotted is that most trading algorithm focuses trading on a classical trading where only long positions are available. Rather, we suggest to do research on margin trading where we can use short positions and leverages. Short position allows to earn in the bear market and leverage allows asymmetric return where systemic trading could flower.

If the agent could perform well using short positions and leverage, it's possible for the agent to beat the baseline return without capturing the random extreme price soaring wave. For your information, using short positions and leverage is not a radical assumption, since many crypto exchanges offer to use margin trading with fair amount of fees.

Still, this additional options doesn't directly mean agent's yield curve to rise. Rather, the agent could lose its money even in the bull market and one single fail in prediction could lead to irreducible liquidation. Thus, risk management plays key role, and we suggest to use distributional RL to integrate these risks into agent's consideration.

Distributional reinforcement learning focuses on the intrinsic randomness of returns within the RL framework. As the agent interacts with the environment, irreducible randomness seeps in through the stochasticity of these interactions. Distributional RL aims to model the distribution over returns, whose mean is the traditional value function, and to use these distributions to evaluate and optimize a policy.

Particulary, we use Implicit Qunatile Networks (IQN) and distort the sampling method to control the risk sensitivity. Risk-averse agents choose actions based on the worst estimated q-value and which leads to conservative decisions. We also use other extensions in DQN to train the agent. Also, we use transformer layer to effectively and efficiently extract featrues from time-series data. We implement 3 environments which are long-position only env, short-position available environment, leverage available environment and show our approach achieves uncomparable yields.

II. BACKGROUND / RELATED WORK

A. Implicit quantile networks(IQN)

Any distributional RL algorithm is characterized by two aspects: the parameterization of the return distribution, and the distance metric or loss function being optimized. Together, these choices control assumptions about the random returns and how approximations will be traded off. Categorical DQN (Bellemare et al., 2017, C51) combines acategorical distribution and the cross-entropy loss. However, these algorithms are restricted to assigning probabilities to an a priori fixed, discrete set of possible returns. Dabneyet al. (2018) propose an alternate pair of choices, parameterizing the distribution by a uniform mixture of Diracs whose locations are adjusted using quantile regression. Their algorithm, QR-DQN, while restricted to a discrete set of quantiles, automatically adapts return quantiles to minimize

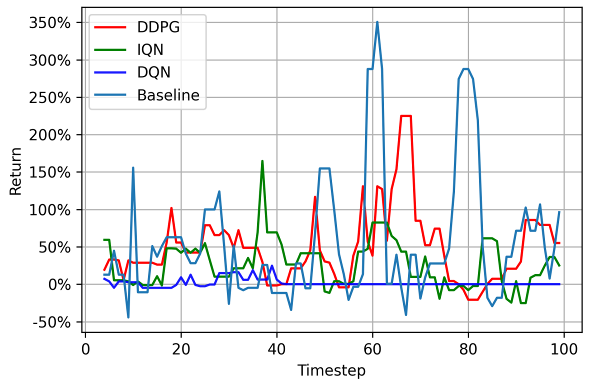

Fig. 1. Several RL algorithms's return from BTC trading. Algorithms are often outperformed by Baseline return which is just buy and hold strategy.

the Wasserstein distance between the Bellman updated and current return distributions. In this paper, we use the extended approach of Dabney et al.(2018), which is internal qunatile networks(IQN) from learning a discrete set of quantiles to learning the full quantile function, a continuous map from probabilities to returns. When combined with a base distribution, such as U([0,1]), this forms an implicit distribution capable of approximating any distribution over returns given sufficient network capacity.

IQN is a deterministic parametric function trained to reparameterizesamples from a base distribution, e.g. $\tau \sim U([0,1])$, to the respective quantile values of a target distribution. IQN provides an effective way to learn an implicit representation of the return distribution, yielding a powerful function approximator for a new DQN-like agent. The algorithm provides, a fully integrated distributional RL agent without prior assumptionson the parameterization of the return distribution. IQN can be trained with as little as a single sample from each state-action value distribution, or as many as computational limits allow to improve the algorithm's data efficiency. Furthermore, IQN allows us to expand the class of control policies to a large class of risk-sensitive policies connectedto distortion risk measures.

B. Risk-sensitive RL

Risk-sensitve RL refers to choose action not an argmax from average Q-value but from worst-case Q-value. IQN samples percentiles from some distribution, for example, uniform distribution, if we choose actions based on 25. In this paper, we use conditional value-at-risk (CVaR) $CVaR(\eta,\tau) = \eta\tau$. because its implementation as a modification to the sampling distribution of τ is particularly simple, as it changes $\tau \sim U([0,1])$ to $\tau \sim U([0,\eta])$, yet showed powerful performance in the paper.

C. Transformer

Transformer is a self-attention model. Recently, attention mechanisms have become an integral part of models that must capture global dependencies (Bahdanau et al., 2014; Xu et al., 2015; Yang et al., 2016; Gregor et al., 2015; Chen et al., 2018). In particular, self-attention (Cheng et al., 2016; Parikh et al., 2016), also called intra-attention, calculates the response at a position in a sequence by attending to all positions within the same sequence.

D. PER

Prioritized Experience replay. DQN samples uniformly from the re-play buffer. Ideally, we want to sample more frequently those transitions from which there is much to learn. As a proxy for learning potential, prioritized experience replay(Schaul et al. 2015) samples transitions with probability relative to the last encountered absoluteTD error

$$p_t \propto |R_{t+1} + \gamma_{t+1} max_{\acute{a}} q_{\bar{\theta}}(S_{t+1}, \acute{a}) - q_\theta(S_t, A_t)|^\omega$$

where ω is a hyper-parameter that determines the shape of the distribution. New transitions are inserted into the replay buffer with maximum priority, providing a bias towards recent transitions. Note that stochastic transitions might also be favoured, even when there is little left to learn about them.

E. MultistepRL

Q-learning accumulates a single reward and then uses the greedy action at the next step to bootstrap. Alternatively, forward-view multi-step targets can beused (Sutton 1988). We define the truncated n-step return from a given state S_t as

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}$$

A multi-step variant of DQN is then defined by minimizingthe alternative loss,

$$(R_t^{(n)} + \gamma_t^{(n)} max_{\acute{a}} q_{\bar{\theta}}(S_{t+n}, \acute{a}) - q_\theta(S_t, A_t))^2$$

Multi-step targets with suitably tuned n often lead to fasterlearning (Sutton and Barto 1998).

III. Model

A. Architecture

The model basically outputs Q-value for each action when inputted observation(OHLCV data) and info. Info includes agent's current position and average holding price, remaining trading time until the end,remaining cash. The observation is put through the transformer encoder layer and then outputted features are concatenated with the info features These concatenated features again pass through the two hidden layers which are composed with 512, 32 layers each. Then, τ is sampled from U[0,1] and processed with

$$\pi_j(\tau) := ReLU(\sum_{i=0}^{n-1} cos(\pi i \tau)\omega_{ij} + b_j).$$

Finally, the product of the feature from neural net and processed τ becomes the expected Q-value of the action.

B. Model Training

Training the model follows the methodology of IQN. The model is a deterministic parametric function trained to reparameterize samples from a base distribution, e.g. $\tau \sim U([0,1])$, to the respective quantile values of a target distribution.

Let $F_Z^{-1}(\tau)$ be the quantile function at $\tau \in [0,1]$ for the random variable Z. For notational simplicity we write $Z_\tau := F_Z^{-1}(\tau)$, thus for $\tau \in U([0,1])$ the resulting state-action return distribution sample is $Z_\tau(x,a) \sim Z(x,a)$.

Let $\beta : [0,1] \rightarrow [0,1]$ be a distortion risk measure, with identity corresponding to risk-neutrality. Then, the distorted expectation of Z(x,a) under β is given by

$$\pi_\beta(x) = arg\max_{a \in A} Q_\beta(x,a)$$

For two samples $\tau, \tau' \sim U([0,1])$, and policy π_β, the sampled temporal difference (TD) error at step t is

$$\delta_t^{\tau,\acute{\tau}} = \gamma_t + \gamma Z_{\tau'}(x_{t+1}, \pi_\beta(x_{t+1})) - Z_\tau(x_t, a_t)$$

Then, the IQN loss function is given by

$$L(x_t, a_t, r_t, x_{t+1}) = \frac{1}{N'} \sum_{i=0}^{N} \sum_{j=1}^{N'} \rho_{\tau_i^\kappa}(\delta_t^{\tau,\tau'})$$

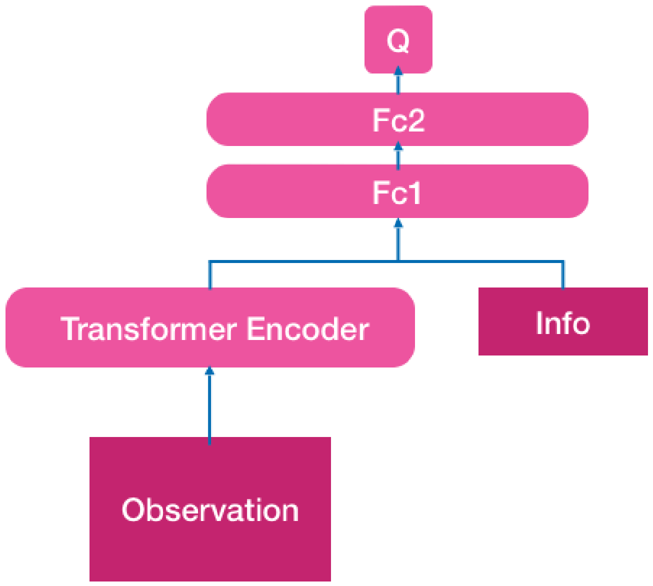

Fig. 2. Model architecture

IV. Environments

We implement 3 environments as mentioned in the introduction. Long-only, short-available, leverage-available environments.

A. Common settings

Every environment has same observation, reward, action mechanisms and episode progress mechanisms.

In every episode, subset of historical data(N=480) is sampled randomly from the total data. Total data refers to Bitcoin historical price data ranges from 2016-01-01 to 2021-04-30. Then at each step, observation is given which is 256 OHLCV(Open-High-Low-Close-Volume) data. Then, agent decides its new position based on the observed state and its current status. Agent's status includes agent's current position and average holding price, remaining trading time until the end,remaining cash. When agent's action is given, the environment calculates the yield.

$$Y = \frac{(P - M) * N}{B}$$

P=price, M=average holding price, N=amount of traded coins, B=initial budget. Note that simple long or short position doesn't occur any reward. The reward only occus when covering the position. For example, if you choose to extend your long position from 1BTC to 2BTC, you don't get any reward. On the other hand, if you have 1BTC and choose to convert your long position to short position, you get positive or negative reward for covering your long position, and get nothing from your starting short position.

For your information, since each episode has 480 data and agent observes 256 data at each step, we could infer that one episode is comprised with 224 steps.

B. Long-only environment

In the Long-only environment, Agents can only choose to long, but they can control the size of the position. It's same as classical trading where you can only buy some assets and sell if you have assets. For example, if you want to sell some bitcoins, you should already have some bitcoins in the long-only environment. The cons of this env market is that agents cannot earn anything if the market is bear market.

C. Short-available environment

In the short-available environment, agents can take a short position. This allows agents to make profit even if the market is down trended. For example, you can sell some bitcoins even if you have any, then earn some profits when price dropped by buying back what you previously sold. However, this simultaneously means the agent can lose from both direction, even if the market soars. If you short 1BTC at 40000$ and close your position at the 45000$, you lose 5000$. In Fig2, this possibility of losing money in bull market is illustrated.

D. 10x Leverage environment

In the leverage-available env, agents not only take both position but also can enjoy 10x leverage. For example, the agent could buy 10 BTC if it have a cash amounts to 1BTC. In other words, if the agent is sure that the market will go up, it could mobilize ten times the budget it actually has. This could result in huge profit uncomparable in classical trading in a single trade. In fact, Moon et al.(2016) mentioned that the only way algorithmic trading could beat baseline return is to use leverage. However, using leverage comes with a risk of liquidation. If the prices goes opposite side varying more than 10 percent, the agent would lose all the money it invested, and could never get back this money even after the price changes the direction in agent's favor later. For example, if you buy BTC at \$40000 with full leverage and the price dropped to \$35000, you're liquidated and have nothing left. It doesn't help anything if the BTC soars to \$50000 after the liquidation. Therefore, risk management plays key role when using leverage. Agent could lose all of its money at a single trade no matter how successful it previously did before.

V. Experiments

A. Hyperparameters

In all the experiments, we use Adam optimizer with a learning rate of 0.001. The discount factor of reward γ is 0.96. For the soft update of target networks, we use τ= 0.001. The neural networks use instance normalization at the first layer and use ReLU as an activation. The transformer encoder layer's embedding dimension is 512, 8 heads and use dropout ratio 0.1. There are two hidden fully connected layers with 512 and 64 units respectively. 32 τ is sampled from U[0,1] and we take a minibatch of 256.

B. Comparing short-available strategy and long-only strategy

In this experiment, we compare the return when the agent learns optimal trading policy using only long position and using short position also. Agent is rewarded based on the yield rate compared to the initial budget. Long-only agent could only make profit when base price rises, while short-available agent could make profit regardless of the market trend theoretically.

Ideally, short-available agent could always make plus profit taking short position when the price is dropping, and vice versa. This is illustrated in Fig4, where both agent achieves smiliar level of yield and short-available agent sometimes makes profit while long-only agent only achieving near 0 percent yield.

However, observing only OHLCV data, it is not always possible to predict the future price accurately. Since the agent tries to generalize the pattern rather than memorizing, it could fail to capture the idealistic action from time to time. This could lead to loss even in the bull market, and it is illustrated in the 80-90 episode in the Fig.4, short-available agent losing while long-only agent is earning the highest yield ever. This is so-called bi-directional risk which means allowing short position could result in loss no matter the price trends toward up or down. In summary, the agent should learn to manage the risk if it to fully enjoy the ability of short positions.

C. Comparing short-available strategy and risk-aware short-available strategy

At the previous experiment, we could see that agent suffers from bi-directional risk, and we conclude agent needs risk management skills To realize that, we introduced CVaR distortion. Previously, τ was sampled from U[0,1]. After distorting CVaR, we now sample τ from U[0,1] * η In this experiment, we use η=-1 because it showed robust performance than the other options. This distortion allows

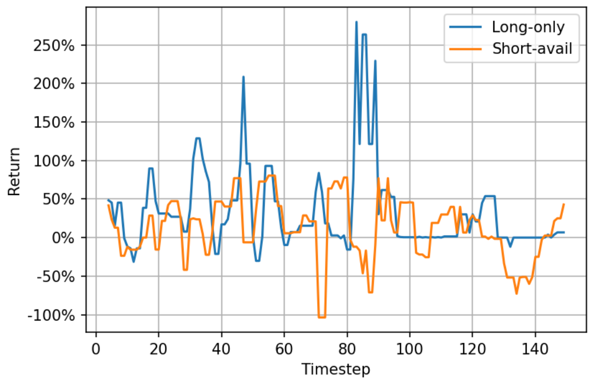

Fig. 3. Comparison of the return between long-only strategy and short-available strategy. Short-available strategy sometimes outperform long-only strategies but total expected return is smaller.

the agent to make decision based on worst estimation of each actions. In other words, it chooses maxmin action. In Fig. 4, we find that the agent learned to use risk-aware strategy, making huge profit while risk-neutral agent suffers from huge losses. As risk-aware strategy without CVaR distortion is exactly same as risk-neutral strategy, we can see CVaR distortion indeed helps. The agent could make stable and sizable profit because it could spare its budget during the uncertain phase in the market, and take full position if it's sure.

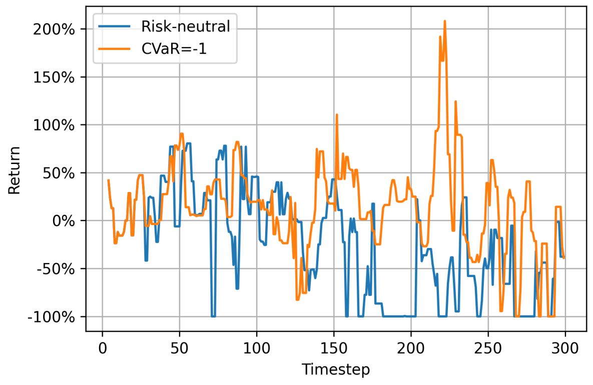

Fig. 4. Comparison of the return between long-only strategy and short-available strategy. Short-available strategy sometimes outperform long-only strategies but total expected return is smaller.

D. Leverage available strategy

We showed the agent could manage its risks and make profit from both price direction. Now we further exploit this risk management ability using leverage to maximize the profit. However, using leverage is often not recommended for investors because of the risk of liquidation. Once liquidated, even if the price turns to favor direction later, you cannot get back your money. No matter how successful agent did at the past, one single liquidation could lead to losing everything.

In this experiment, we assumes 10x leverage, so if the price deviates more than 10 percent from the targeted position, the agent will be swapped out. This extreme risk of liquidation turns down people using leverage. However, at the same time, leverage allows uncomparable opportunity to make huge profit. However, using leverage is often not recommended for investors because of the risk of liquidation. Once liquidated, even if the price turns to favor direction later, you cannot get back your money. No matter how successful agent did at the past, one single liquidation could lead to losing everything. In this experiment, we assumes 10x leverage, so if the price deviates more than 10 percent from the targeted position, the agent will be swapped out. This extreme risk of liquidation turns down people using leverage. However, at the same time, leverage allows uncomparable opportunity to make huge profit. Under spot trading environment, even if the agent is so sure that the price would drastically rise, all of things it can do is to invest all the budget it has. However, under leverage-available environment, agent could mobilize external money and make 10x more profit. This could be inspected in Fig.5, leverage agent could make profit up to 1000% while long-only agent and baseline return maximum was under 400% Therefore, we conclude that if the agent cuold manage the risk, leverage is the best option to maximize the return.

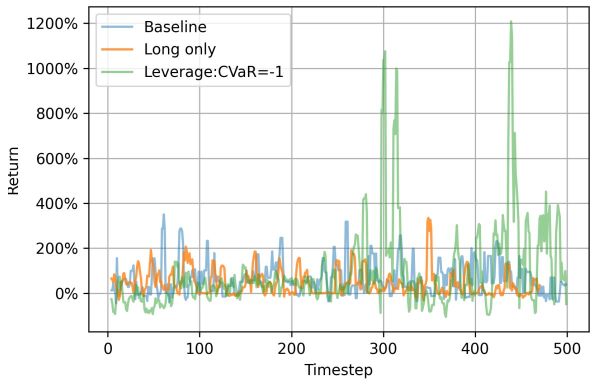

Fig. 5. Comparison of the return between the baseline, long-only strategy and risk-aware leverage strategy.

In Fig.6., we compared the return using risk-neutral strategy and risk-averse strategy. We can see that the area under 0% is much wider when using risk-neutral strategy than risk-averse strategy. This diminishing loss behavior is really desirable thing in leverage environment. Because in leverage environment, if you could just manage to survive, usually you will end up beating the baseline return. Moreover, while risk-averse strategy succeeded in diminishing the loss, it doesn't fall behind in terms of the overall return at the latter part of the episode.

E. *Real-world trading using leverage*

In this section, we show the real-world BTC trading results on binance. Both long and short positions are available, and used 3x leverage. The seed money was 1000$ and trading was done from 2021-06-08 to 2021-06-13. We could see that the agent successfully managed the risk, cutting of the some loss and learn to satisfy on the small amount of money.

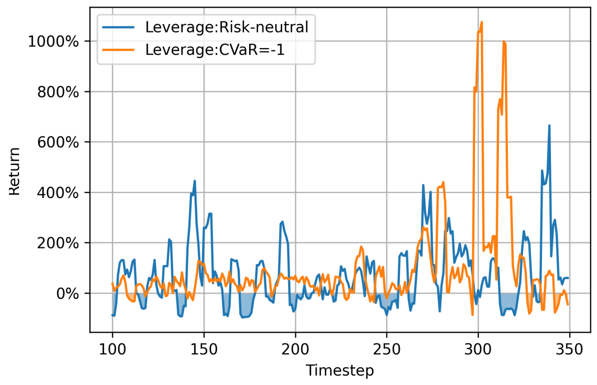

Fig. 6. Comparison of the return using risk-neutral strategy and risk-averse strategy. The area under 0% is much wider when using risk-neutral strategy than risk-averse strategy.

TABLE I
RESULTS ON REAL-TRADING ON BINANCE
2021-06-08 TO 2021-06-13

Date	Pair	Profit and Loss
2021-06-13	BTCUSDT	134.05496000
2021-06-13	BTCUSDT	3.24112000
2021-06-13	BTCUSDT	40.82325000
2021-06-13	BTCUSDT	48.15050000
2021-06-09	BTCUSDT	54.23129252
2021-06-09	BTCUSDT	164.33725006
2021-06-09	BTCUSDT	65.73490002
2021-06-09	BTCUSDT	188.98783757
2021-06-09	BTCUSDT	148.29225006
2021-06-09	BTCUSDT	47.90790007
2021-06-08	BTCUSDT	31.04080875
2021-06-08	BTCUSDT	-12.45155906

VI. DISCUSSION AND CONCLUSIONS

Observing only OHLCV data, it is not always possible to predict the future price accurately. We bypass this problem in this paper, but additional data more than OHLCV could help predict the future price and imporve the model more accurately.

We only used value-based approach in the work, but policy-gradient based approach could be also tried with constrained policy optimization. Since policy-gradient allows continuous actions, the agent could use more flexible policy and this could lead to improvement. For example, DDPG + CPO worths try.

We have proposed adaption of recent work basedaround using IQN to learn the optimal policy under risk of liquidation. Our adaption leads to a siginificant improvement in the return in the leverage-available environment, restricting the risk of liquidation and maximize the return with the leverage. Finally, we show substantial gains in the real trading experiments, successfully making profit while controlling the loss.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] DABNEY, Will, et al. Implicit quantile networks for distributional reinforcement learning. In: International conference on machine learning. PMLR, 2018. p. 1096-1105.

[2] VASWANI, Ashish, et al. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

[3] Sutton, R. S., and Barto, A. G. 1998.Reinforcement Learn-ing: An Introduction. The MIT press, Cambridge MA.

[4] Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015.Prioritized experience replay. InProc. of ICLR.

[5] DULAC-ARNOLD, Gabriel; MANKOWITZ, Daniel; HESTER, Todd. Challenges of real-world reinforcement learning. arXiv preprint arXiv:1904.12901, 2019.

[6] Byunro, Moon. 2016. Metric Studio., p.56–60.