SEMI-OFFLINE REINFORCEMENT LEARNING FOR PORTFOLIO OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce semi-offline reinforcement learning (RL), a new formalization of the sequential decision-making problem for portfolio optimization. Unlike the standard and the fully-offline RL settings, the unique challenge of semi-offline RL is the limited access to an actively evolving environment. Therefore, existing online/offline RL approaches are incapable of handling the distributional shift between the fixed observations in the training set and those in an out-of-distribution test domain. In this paper, we propose a novel off-policy RL algorithm named *stationarity-constrained MDP* (SC-MDP), which decouples the previously-collected training observations into two streams of *stationary* and *non-stationary* latent variables through a probabilistic inference framework. We demonstrate that in this way, the learned policies can be persistently profitable despite rapidly-changing environment dynamics. Our approach remarkably outperforms the existing online RL algorithms, advanced offline RL methods, and state-of-the-art stock prediction models on three real-world financial datasets.

1 INTRODUCTION

Portfolio optimization is a typical real-world application of reinforcement learning (RL), which requires the agents to identify and capitalize on promising trading opportunities. However, it presents two practical challenges that may violate the basic assumptions of standard RL approaches:

1. RL agents can only learn from a fixed set of historical data, *e.g.*, daily stock prices and trading volumes, without further opportunity to fully explore the environment.
2. Policies are learned and executed in evolving domains with non-stationary dynamics and rewards, in the sense that the assumption of stationarity of standard RL no longer holds.

To learn from the offline data, the advanced off-policy RL approaches, such as BCQ (Fujimoto et al., 2019) and CQL (Kumar et al., 2020), are mainly focused on tackling the overestimation problem of Q values, which is induced by the distributional shift between the learned policy $\hat{a} \sim \pi_\theta(a|s)$ and that in the previously-collected dataset $a \sim \mathcal{D}$. However, we rethink the challenge of Q-overestimation and find that it is not significant in our setup. Because unlike offline RL, in portfolio optimization, agents can interact with offline data to try new actions and collect rewards according to the investment returns. We formalize this problem setting with *semi-offline reinforcement learning*.

In semi-offline RL, as shown in Figure 1(a-b), there are two kinds of states. The offline states, denoted by s', are restricted in the static dataset with state transitions independent of the actions. For example, they may reflect the global dynamics of the historical financial markets. Other parts of the states (termed as *online states*), denoted by s, can be updated in an interactive manner by exploring new actions. In our setup, they may reflect the account balance and current assets. In this paper, we demonstrate that due to the distributional shift[1] induced by the non-stationarity of s', existing online/offline RL algorithms, such as SAC (Haarnoja et al., 2018), BCQ (Fujimoto et al., 2019), and CQL (Kumar et al., 2020), are incapable of learning consistently profitable policies $\hat{a} \sim \pi_\theta(a|s, s')$ across the evolving domains, making them ineffective for the semi-offline RL setting.

[1]Since we have very limited access to the feasible state space of s' in the offline dataset, the performance of the RL algorithms is severely affected by the distributional shift of s' across domains.

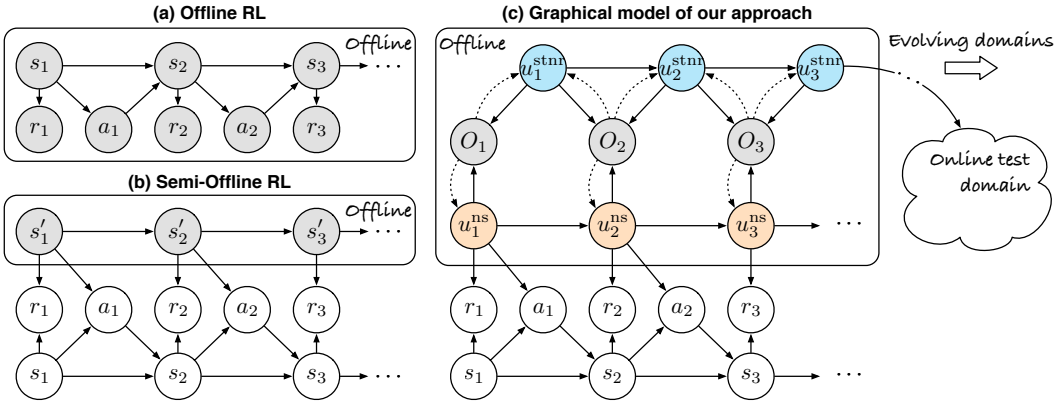

Figure 1: (a-b) In comparison with the offline RL setting, in *semi-offline RL*, the agent can interact with the offline dataset by trying new actions, updating parts of the states, and receiving corresponding rewards in a pre-defined form. (c) The graphical model of *stationarity-constrained MDP* (SC-MDP) with two streams of stationary and non-stationary latent variables that are inferred from a fixed set of observations. The key idea is to make the learned variables and corresponding policies more robust to the distributional shift of the dynamics in an actively evolving environment.

The goal of this work is to develop a semi-offline RL approach that can cope with the non-stationarity of the offline states over the training dataset and the online test environment. To this end, we introduce the *stationarity-constrained Markov decision process* (SC-MDP), which decouples the offline state dynamics with two sequences of latent variables, as shown in Figure 1(c). Both variables are presented as Gaussian distributions. Specifically, the **non-stationary variable** u^{ns} is inferred from the observed *daily technical indicators* of stock prices and trading volumes. As the number of assets[2] increases, the above observations may rapidly change over time, which indicates a larger distributional shift across domains. To ease such non-stationarity, we draw upon ideas from the probabilistic inference to derive u^{ns} that can reason about the stationary component in the offline observations, *i.e.*, the *covariance matrix* between the technical indicators of each asset over the past year. On the other side, we infer the **stationary variable** u^{stnr} from the covariance matrix and keep it from learning static representations by forcing it to reason about the daily offset of future observations.

We evaluate the SC-MDP algorithm on three trading benchmarks from the NASDAQ and Chinese stock markets, as well as the cryptocurrencies market. Our approach consistently outperforms existing RL and stock prediction models. In summary, the contributions of this paper are as follows:

- We present semi-offline RL, a new problem setting in which the key challenge is to mitigate the distributional shift in state space between a previously-collected dataset and evolving test domains.
- We propose SC-MDP, a novel off-policy algorithm, to address the challenge of non-stationarity in the semi-offline setting. The decoupling of the stationary and non-stationary latent states enables the learned policy to be persistently effective for highly dynamic environments.

We focus on finding an RL solution to portfolio optimization. Nevertheless, the semi-offline RL is a general setting. The key idea of SC-MDP can also be applied to other decision-making problems with highly non-stationary dynamics and limited data collection, such as recommendation systems.

2 Problem Setup

2.1 Portfolio Optimization Problem Definition

As a MDP problem, the portfolio optimization involves observing change of stock market, and taking an action to maximize rewards under a trading strategy, written as $M := (S, A, r, T, \rho, \gamma)$, including a state space S, action space A, Markovian transition kernel $T(s'|s, a)$, reward function $r(s, a)$, initial state distribution $\rho(s_0)$, and discount factor $\gamma \in (0, 1)$. We give the definitions of state space, action space and reward function in our semi-offline RL framework as following.

[2]We perform portfolio optimization over nearly 100 stocks in our experiments.

State space. We use three types of information that the trading agent receives on the stock datasets: (i) the technical indicators $O_t^{\text{tech}} \in \mathbb{R}^{N \times K}$, where N is the number of concurrent time series and K is the number of statistics that capture the 146 dynamics of the time series, in which we randomly select half of the time series and mask their input statistics by zero; (ii) the covariance matrix $O_t^{\text{cov}} \in \mathbb{R}^{N \times N}$ between sequences of daily close prices of all concurrent time series over a fixed period of time before t; (iii) the account state $s_t \in \mathbb{R}^{N+1}$, representing the amount and the balance of each holding stock.

Action space. After analyzing various information, the trading agent needs to interact with the environment by executing a trade. We use $a_t \in \mathbb{R}^N$ in a continuous action space to denote the action at each time step, and then discretized into multiple discrete intervals of daily trading signals, indicating the amount of buying, holding, or selling shares on each trading target.

Reward function. The reward $r_t \sim \mathcal{R}(s_t, a_t)$ is defined as the daily portfolio return ratios.

2.2 Offline and Semi-offline Reinforcement Learning

In the offline RL, the environment is not available during the training phase, and we have access only to a fixed dataset $D_{env} = \{(s_t, a_t, r_t, s_{t+1})\}$, which pre-collected by a behavior policy π_β. The distribution induced by D_{env} is called as the behavioral distribution. The goal in offline RL is to find the best policy given this particular offline dataset D. However, standard off-policy deep reinforcement learning algorithms such as SAC and DDPG are incapable of learning well in offline setting due to overestimation of values induced by the distributional shift between the behavioral policy and the learned policy.

Semi-offline RL allows the agent to interact with offline data to take new actions, which decreases distributional shift by constantly collecting new data. As shown in Figure 1, the states include offline states s' and online states s. At each time step, we take O_t^{tech} and O_t^{cov} as the input states, and learn stationary and non-stationary latent variables $\{u_t^{\text{stnr}}, u_t^{\text{ns}}\}$ from them to make up offline states s'. After combining s' and s, we can better train a policy in an interactive environment.

3 Stationarity-Constrained Markov Decision Process

As aforementioned, we have formulated portfolio optimization problem as a semi-offline RL task, that is, the RL agent can collect more (a_t, r_t) pairs under the finite state set $\mathcal{S}$ besides the tuples collected by expert policy. Unlike the difficulty of overestimating the values in the offline setting, the new challenge in our task lies in the distribution shift of state between the training and test sets. To tackle this challenge, we formulate the portfolio optimization problem as an stationarity-constrained Markov decision process (SC-MDP), which corresponds to a sequence of stationary MDPs, represented in a latent space with the help of deep neural networks.

3.1 Stationarity-Constrained State Representation

State distribution mismatch. Through experimental validation, we find that using the trajectories collected by expert policy as our initial buffer will improve the performance of the RL algorithms. Similarly to the offline RL, the datasets from behavior policy and the learned policy has a certain distributional shift at the start of training. As a result, the mainstream offline RL algorithms like BCQ (Fujimoto et al., 2019) and CQL (Kumar et al., 2020) or imitation learning can be used as a kind of method for this problem. However, the key challenge existed in the portfolio optimization semi-offline RL task is the distributional shift of the observation due to the complexity and uncertainty in investment markets, while the available viable solutions mentioned above are unable to tackle this difficulty. (Experimental verification of the points in this paragraph can be checked in section 4.1.) By analyzing the two subparts the covariance matrix O_t^{cov} and common technical indicators O_t^{tech} included in the currently defined RL observation – O_t, the issues arisen from them show in two aspects. On the one hand, the non-stationary characteristic in O_t^{tech} mainly causes the distributional shift of O_t, which will bring about a degradation in the performance of the trained RL models on the test set. On the other hand, the time-invariant nature of the O_t^{cov} poses difficulties for learning the distribution of actions based on different states. Therefore, we propose two state learners in which

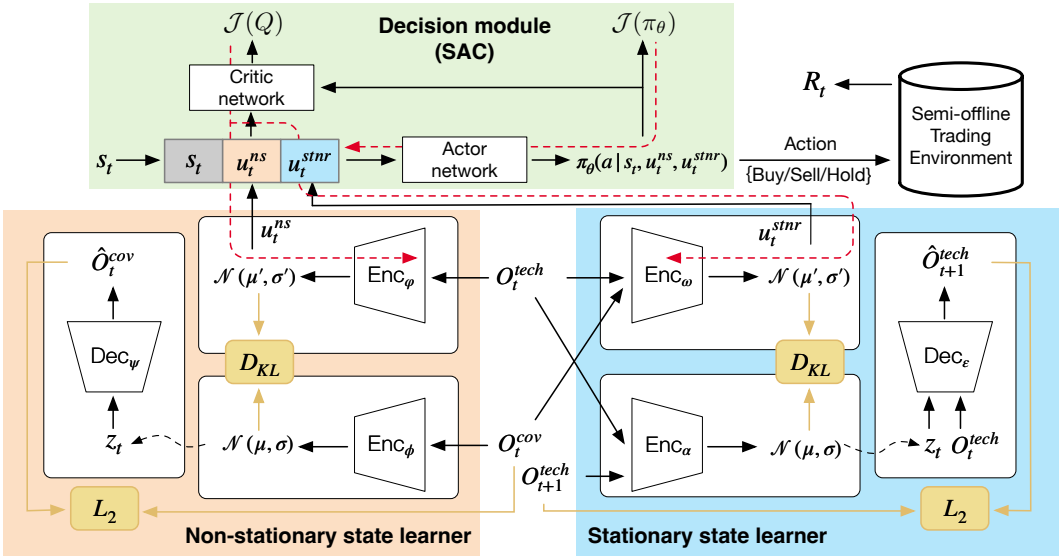

Figure 2: Unlike previous RL-for-finance methods, our method builds the decision module upon learned representations provided by the non-stationary state learner and stationary state learner. In each state learner, there exists three modules: a prior learner, a posterior learner and a decoder. In particular, the critic propagates its gradients of state values back into two state learners.

any subpart of the state takes another as a supervised constraint and learns a kind of representation that encapsulates the data characteristics of both sides under the hidden space.

Non-stationary state learner. As shown in Figure 2 (Left), the non-stationary state learner includes three parts. The posterior leaner takes the O_t^{cov} as input and we will get a latent variable z_t sampled from its output distribution $N(\mu, \sigma)$. The target of the decoder is to decode the O_t^{cov} given the z_t. On top of that, the prior learner takes the non-stationary observation O_t^{tech} as input and learns a distribution that is as close as possible to the output of the posterior learner by using the KL-divergence as a constraint. The process can be formulated as follows:

$$\begin{aligned} \mu, \sigma &= \text{Enc}_\phi(O_t^{\text{cov}}) \\ z_t &\sim \mathcal{N}(\mu, \sigma) \\ \hat{O}_t^{\text{cov}} &= \text{Dec}_\psi(z_t) \\ \mu', \sigma' &= \text{Enc}_\varphi(O_t^{\text{tech}}) \end{aligned} \tag{1}$$

And the loss function can be represented as:

$$\mathcal{J}_t^{\text{ns}} = \ell_2(\hat{O}_t^{\text{cov}}, O_t^{\text{cov}}) - \beta D_{\text{KL}}(q_\phi(z_t|O_t^{\text{cov}}) \| p_\varphi(z_t|O_t^{\text{tech}})) \tag{2}$$

Under the constraint of the loss function, the distribution $\mathcal{N}(\mu', \sigma')$ learned from O_t^{tech} will gradually move closer to the $\mathcal{N}(\mu, \sigma)$ with the stationary property, which will mitigating the distribution differences between the training and test sets. Therefore, we define $u_t^{\text{ns}} = \text{concat}(\mu', \sigma')$ as part of the state used in decision module.

Stationary state learner. The structure of the stationary state learner is similar to that of the non-stationary state learner. However, considering the stationary property in the covariance matrix, we add O_t^{tech} as an additional condition in all three subparts of the state learner and reconstruct the technical indicators at the next time step (O_{t+1}^{tech}). Under the constraint of L2 loss and KL-divergence, the distribution from prior learner will not only has the property of non-stationary, but also reflects the distribution of technical indicators in the future. We define $u_t^{\text{stnr}} = \text{concat}(\mu'', \sigma'')$ as a second

Algorithm 1 Stationarity-Constrained Markov Decision Process (SC-MDP)

Input: env, α_Q, α_π, α_{ns}, α_{stnr}
1: Randomly initialize θ_Q, θ_π, θ_{ns}, θ_{stnr}
2: Load replay buffer $\mathcal{D}$ collected by the expert policy
3: **for** i=1, 2, ... **do**
4: Get state s_t, O_t^{cov}, O_t^{tech} from the env, get $u_t^{\text{ns}} = \text{Enc}_\phi(O_t^{\text{cov}})$, $u_t^{\text{stnr}} = \text{Enc}_\omega(O_t^{\text{tech}})$, then $s = \text{concat}(s_t, u_t^{\text{ns}}, u_t^{\text{stnr}})$
5: Collect trajectory τ^i with $\pi_\theta(a|s)$ and update replay buffer $\mathcal{D}[i] \leftarrow \tau^i$ with observed data
6: **for** j=1, 2, ..., N **do**
7: Sample a batch of steps E from $\mathcal{D}$
{Update actor and critic}
$\theta_Q \leftarrow \theta_Q - \alpha_Q \nabla_{\theta_Q} \mathcal{J}_Q$, $\theta_\pi \leftarrow \theta_p i - \alpha_\pi \nabla_{\theta_\pi} \mathcal{J}_\pi$
{Update non-stationary state learner}
$\theta_{\text{ns}} \leftarrow \theta_{\text{ns}} - \alpha_{\text{ns}} \nabla_{\theta_{\text{ns}}} (\mathcal{J}_{L2}^{\text{ns}} + \mathcal{J}_{\text{KL}}^{\text{ns}} + \mathcal{J}_Q)$
{Update stationary state learner}
$\theta_{\text{stnr}} \leftarrow \theta_{\text{stnr}} - \alpha_{\text{stnr}} \nabla_{\theta_{\text{stnr}}} (\mathcal{J}_{L2}^{\text{stnr}} + \mathcal{J}_{\text{KL}}^{\text{stnr}} + \mathcal{J}_Q)$
8: **end for**
9: **end for**

source of the state s_t. The process and the loss function can be represented as follows:

$$\begin{aligned} \mu, \sigma &= \text{Enc}_\phi(O_t^{\text{cov}}) \\ z_t &\sim \mathcal{N}(\mu, \sigma) \\ \hat{O}_t^{\text{cov}} &= \text{Dec}_\psi(z_t) \\ \mu', \sigma' &= \text{Enc}_\varphi(O_t^{\text{tech}}) \end{aligned} \tag{3}$$

$$\mathcal{J}_t^{\text{stnr}} = \ell_2(\hat{O}_{t+1}^{\text{tech}}, O_{t+1}^{\text{tech}}) - \beta D_{\text{KL}}(q_\alpha(z_t|O_{t+1}^{\text{tech}}, O_t^{\text{tech}}) \| p_\omega(O_t^{\text{cov}}, O_t^{\text{tech}})) \tag{4}$$

3.2 Policy Optimization

We formulate RL-based portfolio optimization as a new problem which we called SC-MDP. Instead of learning the policy directly from the observed data, we takes a combined state space $\mathcal{S}$ that involves the learned representations from the two state learner branches and account related state. The final state used in decision module can be formulated by $s_t = \text{concat}(u_t^{\text{ns}}, u_t^{\text{stnr}}, s_t')$. On the experiment part, the RL algorithm exploits soft actor-critic (SAC) (Haarnoja et al., 2018) to learn the trading policy. The parametric soft Q-function in the critic network is trained by minimizing the following soft Bellman residual. Note that we omit the subscript t for simplicity.

$$\begin{aligned} L(\phi_i, \mathcal{D}) &= \underset{(s,a,r,s_n,d)\sim\mathcal{D}}{\text{E}} \left[\left(Q_{\phi_i}(s,a) - y(r, s_n, d) \right)^2 \right], \\ \text{with} \quad y(r, s_n, d) &= r + \gamma(1-d) \left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s_n, \tilde{a}_n) - \alpha \log \pi_\theta(\tilde{a}_n \mid s_n) \right) \\ \tilde{a}_n &\sim \pi_\theta(\cdot \mid s_n), \end{aligned} \tag{5}$$

where s and s_n are integrated state at the current and the next time step, $\tilde{a}_n$ is sampled from the action distribution given s_n, and d is the indicator for the end of the episode. To avoid the over-estimation problem, we take the clipped double-Q trick used in SAC. As for the actor network, the training objective can be shown as follows:

$$\max_\theta \underset{\xi\sim\mathcal{D}}{\text{E}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi) \mid s) \right], \tag{6}$$

where ξ is a noise sampled from normal Gaussian distribution using the re-parameterization trick. The entropy loss term in each objective encourages the decision module to take more stochastic actions for better exploration. Our method takes an end-to-end architecture and the critic network also propagates the analytic gradients of state values back into the two state learner branches. Algorithm 1 shows the whole process of our algorithm.

Table 1: Statistical details of the financial investment datasets.

Market	# Stocks / Cryptocurrencies	# Train Days	# Test Days
CSI-300	95	1935	728
NASDAQ-100	86	2000	756
Cryptocurrency	27	1108	258

Table 2: Results on the NASDAQ-100 dataset.

Method	Traininig Set				Test Set			
	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
Imitation learning	0.298	0.302	1.000	0.214	0.100	0.106	0.364	0.470
Vanilla SAC (offline)	-	-	-	-	0.521	0.263	1.278	0.240
Vanilla SAC (Haarnoja et al., 2018)	0.330	0.045	2.282	**0.064**	0.398	0.199	**1.342**	0.212
BCQ-SAC	0.299	0.041	1.807	0.088	0.442	0.228	1.324	**0.199**
CQL-SAC	0.324	0.044	1.654	0.129	0.449	0.194	0.947	0.283
SC-MDP	**0.407**	**0.054**	**2.591**	<u>0.069</u>	**0.563**	**0.274**	1.306	0.273

4 EXPERIMENTS

4.1 DATASET

To reduce the impact of different training and test set lengths on model performance, we explicitly define the length of each episode length as 250 days (approximately the number of trading days in a year) and slice the test set into 10 episodes with the same steps and different start dates, and compare the performance by calculating the average performance of the model on ten different test sets. We use portfolio return (PR), annual return (AR), Sharpe ratio (SP), and maximum drawdown (MDD) as the evaluation metrics.

CSI-300 Stock Dataset. The Chinese stock dataset contains stock data from the CSI-300 Composite Index from 01/17/2011 to 12/30/2021. As shown in Table 1, the dataset is divided into the training and test splits containing the basic stock price-volume information in $1{,}935$ days and 728 trading days respectively. Furthermore, we follow the work from (Feng et al., 2019) to retain the stocks that have been traded on more than 98% training days since 01/17/2011. Our investment pool contains 95 stocks. If a stock in the training set is suspended from trading, we interpolate the missing data in using the daily rate of change of the CSI-300 Composite Index.

NASDAQ-100 Stock Dataset. For the NASDAQ stock dataset, we collect the daily price-volume records and related technical indicators between 01/17/2011 and 12/30/2021 from the website of Yahoo Finance[3]. We use the 98% criteria to filter stocks, which derives an investment pool of 86 stocks, and then fill in the missing data based on the daily rate of change of the NASDAQ 100 Index. We construct the training and test datasets that involve $2{,}000$ and 756 trading days respectively.

Cryptocurrency Market Dataset. Apart from the stock market datasets, we evaluate SC-MDP by using it to make trading decisions in the cryptocurrency market. With the 98% filtering criteria, we select 27 cryptocurrencies and collect their daily records between 11/01/2017 and 05/01/2022 from Yahoo Finance. We split the data into a training set of $1{,}108$ trading days and a test set of 258 trading days.

4.2 PRELIMINARY FINDINGS

Does expert policy benefit OOD decision-making? Yes, but it generally improves SAC by a small margin. As shown in Table 2, we can see that modeling the semi-offline RL problem using the SC-MDP framework (SC-MDP) can better unleash the strength of the expert policy compared with the vanilla SAC (offline) on the test set.

[3] `https://github.com/ranaroussi/yfinance`

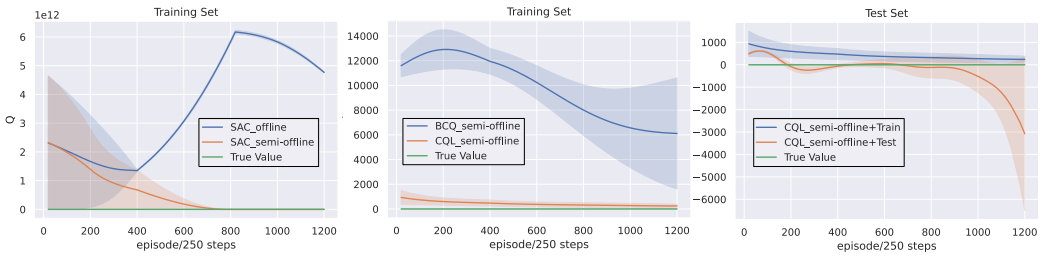

Figure 3: We examine the corresponding value estimate of SAC, BCQ-SAC and CQL-SAC on the NASDAQ training and test sets. Specifically, we validate the value estimate of SAC on the offline and semi-offline two settings respectively. An estimate of the true value of the semi-offline agent, evaluated by Monte Carlo returns, is represented as the green line.

Table 3: Results on the CSI-300 dataset.

Method	Traininig Set				Test Set			
	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
Imitation learning	**0.442**	**0.448**	**1.134**	0.241	0.090	0.091	0.379	0.201
Vanilla SAC ()	0.021	0.010	0.204	0.261	0.617	0.343	1.779	0.238
BCQ-SAC	0.099	0.045	0.614	**0.169**	0.511	0.295	1.717	**0.199**
CQL-SAC	0.004	0.002	0.116	0.180	0.561	0.329	1.705	0.214
SC-MDP	**0.119**	**0.054**	**0.680**	0.172	**0.677**	**0.345**	**2.015**	0.206

Use imitation learning to solve our task. As aforementioned, we take expert policy for the initialization of the buffer, which means we buy a stock each day that has the highest return in the next day and sell it on the third day. Inspired by this, we try to use imitation learning for expert policy, and we adopt the Transformer's architecture and formulate the problem of predict the stock with highest return ratio as a classification problem. However, from Table 2, it can be found that it is difficult to estimate expert actions via supervised learning as the performance behaves poor on the test set, also the accuracy can only reached 7.00%. Similar conclusions were obtained in the other two datasets. Based on the above findings, we decide to abstract the portfolio optimization problem as a semi-offline RL learning task with the help of expert policy as our initial buffer.

Offline RL methods cannot solve semi-offline RL problems. From the Figure 3 (The Left and middle one), it can be found that under the setting of semi-offline, the problem of over-estimation has been largely alleviated. Furthermore, from the right one in Figure 3 , CQL have a worse estimation of values (the orange line) on the test set compared with its estimation on the training set (the blue line), which means the main challenge exists in semi-offline setting is the state distributional shift problem. Therefore, offline RL methods like BCQ and CQL are not the best solutions for our task.

4.3 Main Results

Performance on three datasets. Table 3 shows qualitative results on the CSI-300 training and test set. Although the performance of SC-MDP doesn't perform as well as the imitation learning, it reaches the best on the test set. Compared with the dramatic performance degradation on the test set of imitation learning method, the core problem to be solved in this paper: the state distributional shift problem, can be demonstrated explicitly. Also, from the Table 2, we can see that SC-MDP achieves the best portfolio return on both training and test set. Furthermore, on the Cryptocurrency market (Table 4), most of the methods (like vanilla SAC and CQL-SAC) have a loss on the test set, which means the data property in this dataset shows uncertainty. Also it means the policy learning becomes more difficult, however, SC-MDP still achieves profitable average results which demonstrates the effectiveness of the state constraint mechanism in solving the problem.

Ablation study. To verify the performance of two state learners proposed in the SC-MLP framework. We compare two variants on the NASDAQ dataset and in each variant, we replace the con-

Table 4: Results on the cryptocurrencies dataset.

Method	Traininig Set				Test Set			
	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
Imitation learning	1.486	1.510	1.444	0.359	-0.414	-0.416	-0.575	0.589
Vanilla SAC ()	**4.857**	0.575	**2.53**	**0.316**	-0.080	0.081	0.149	**0.482**
BCQ-SAC	1.911	0.316	1.833	0.385	0.151	0.153	0.544	0.487
CQL-SAC	1.452	**0.696**	1.451	0.422	-0.105	-0.107	0.193	0.521
SC-MDP	3.134	0.440	2.050	0.403	**0.227**	**0.230**	**0.644**	0.504

Table 5: Ablation study on the NASDAQ dataset.

Method	Test Set			
	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
SC-MDP w/o ns state learner	0.278	0.122	1.022	**0.226**
SC-MDP w/o stnr state learner	0.399	0.191	1.183	0.256
SC-MDP	**0.509**	**0.495**	**1.212**	0.282

strained state u_t with the original input o_t. From Table 5, we can see that removing any state learner leads to a remarkable performance drop, and each branch makes essential contributions to the final performance of our final approach.

5 Related Work

5.1 Offline Reinforcement Learning

Offline RL (Riedmiller, 2005; Levine et al., 2020; Lange et al., 2012) has recently emerged as a prominent paradigm in control and decision-making learning, covering the domains including robotic manipulation (Singh et al., 2020), healthcare (Wang et al., 2018) and NLP (Jaques et al., 2020). The main challenge of offline RL is the insufficient coverage of the offline dataset due to lack of interaction with the environment, which results in the distributional shift between the learned policy and the behavior policy (Fujimoto et al., 2019). There are two mainstream methods to handle this challenge. Fisrt, the policy is regularized to avoid visiting the state-action pairs which are not in offline dataset (Wu et al., 2019b; Ghasemipour et al., 2021; Dadashi et al., 2021; Fujimoto & Gu, 2021). Second, it is a feasible way to learn conservative value functions which penalize the estimated values of the unseen state-action pairs (Kumar et al., 2020; Buckman et al., 2020). Drawing on these prior works, we study portfolio optimization in the semi-offline RL setting.

5.2 Reinforcement Learning for Portfolio Optimization

In the field of portfolio optimization, there have been many attempts (Théate & Ernst, 2021; Weng et al., 2020; Liang et al., 2018; Benhamou et al., 2020) to use RL methods to make trading decisions. The main differences among these approaches can be roughly summarized in three aspects: the definition of the input states (Zhong et al., 2020; Liu et al., 2021; Weng et al., 2020), the engineering of reward functions (Liang et al., 2018; Suri et al., 2021), and the RL algorithms (Benhamou et al., 2020; Hu & Lin, 2019; Huotari et al., 2020). Zhong *et al.* (Zhong et al., 2020) used Q-learning to learn a policy on an electronic exchange in discrete state and action spaces. Wang *et al.* (Wang et al., 2019) proposed AlphaStock which uses the RL method to optimize the parameters of a stock prediction model to reach a good performance under a fixed strategy. Liu *et al.* (Liu et al., 2021) presented a framework named FinRL that integrated multiple off-the-shelf RL algorithms such as Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and DDPG (Lillicrap et al., 2015). We use its SAC implementation as an important baseline model of SC-MDP. Notably, most of these approaches formulate the task as an MDP problem, which has limited information in their state space. To grasp more information, our approach learns a mapping from different kinds of hidden representations to integrated latent state space. Besides, FinRL (Liu et al., 2021) defines the states as a combination of

the covariance matrix of the close prices of all stocks and the MACD indicators, whose dimension will sharply increase with the growth of the number of stocks.

5.3 Stock Prediction Models

Mainstream stock prediction methods can be divided into three categories. CNN-based methods (Hoseinzade & Haratizadeh, 2019; Wen et al., 2019; Maqsood et al., 2020; Eapen et al., 2019) take the historical data of different stocks as a set of input feature maps of a convolutional neural network. In contrast, RNN-based methods (Li et al., 2018; Zhang et al., 2017; Long et al., 2020; Qin et al., 2017; Feng et al., 2018) are better at capturing the underlying sequential trends in stocks. Other network structures are used in the recurrent architecture to jointly model the long-term dynamics and the correlations between stocks, such as attention mechanisms (Hu et al., 2018; Li et al., 2018; Wu et al., 2019a), dilated convolutions (Wang et al., 2021; Cho et al., 2019), and graph neural networks (Feng et al., 2019; Wang et al., 2021; Patil et al., 2020). In this paper, we use stock prediction as a predictive coding strategy, extracting short-term and long-term future dynamics from noisy market observations into the state space of the RL agent.

6 Conclusions

In this work, we introduce the semi-offline RL, a new setting of the sequential decision-making problem. Unlike other RL settings, its unique challenge is the limited access to an actively evolving environment. To solve this problem, we propose a novel off-policy RL algorithm named SC-MDP. Our approach exploits a probabilistic inference framework to decouple the previously-collected training observations into two independent streams of *stationary* and *non-stationary* latent variables. SC-MDP remarkably outperforms the existing online RL algorithms, advanced offline RL methods, and state-of-the-art stock prediction models on three real-world financial datasets. We demonstrate that by decoupling and leveraging non-stationary variables, the learned policies can be persistently profitable despite rapidly-changing environment dynamics.

References

Eric Benhamou, David Saltiel, Jean Jacques Ohana, Jamal Atif, and Rida Laraki. Deep reinforcement learning (drl) for portfolio allocation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 527–531. Springer, 2020.

Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

Chun-Hung Cho, Guan-Yi Lee, Yueh-Lin Tsai, and Kun-Chan Lan. Toward stock price prediction using deep learning. In *UCC*, pp. 133–135, 2019.

Robert Dadashi, Shideh Rezaeifar, Nino Vieillard, Léonard Hussenot, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning with pseudometric learning. In *International Conference on Machine Learning*, pp. 2307–2318. PMLR, 2021.

Jithin Eapen, Doina Bein, and Abhishek Verma. Novel deep learning model with cnn and bi-directional lstm for improved stock market index prediction. In *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, pp. 0264–0270. IEEE, 2019.

Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. Enhancing stock movement prediction with adversarial training. *arXiv preprint arXiv:1810.09936*, 2018.

Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2): 1–30, 2019.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Ehsan Hoseinzade and Saman Haratizadeh. Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

Yuh-Jong Hu and Shang-Jen Lin. Deep reinforcement learning for optimizing finance portfolio management. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pp. 14–20. IEEE, 2019.

Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 261–269, 2018.

Tommi Huotari, Jyrki Savolainen, and Mikael Collan. Deep reinforcement learning agent for s&p 500 stock selection. *Axioms*, 9(4):130, 2020.

Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Human-centric dialog training via offline reinforcement learning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 3985–4003. Association for Computational Linguistics, 2020.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In Marco A. Wiering and Martijn van Otterlo (eds.), *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pp. 45–73. Springer, 2012.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input lstm. In *ACML*, pp. 454–469, 2018.

Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. *ICAIF*, 2021.

Jiawei Long, Zhaopeng Chen, Weibing He, Taiyu Wu, and Jiangtao Ren. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in chinese stock exchange market. *Applied Soft Computing*, 91:106205, 2020.

Haider Maqsood, Irfan Mehmood, Muazzam Maqsood, Muhammad Yasir, Sitara Afzal, Farhan Aadil, Mahmoud Mohamed Selim, and Khan Muhammad. A local and global event sentiment based efficient stock exchange forecasting using deep learning. *International Journal of Information Management*, 50:432–451, 2020.

Pratik Patil, Ching-Seh Mike Wu, Katerina Potika, and Marjan Orang. Stock market prediction using ensemble of graph theory, machine learning and deep learning models. In *Proceedings of the 3rd International Conference on Software Engineering and Information Management*, pp. 85–92, 2020.

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

Martin A. Riedmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge, and Luís Torgo (eds.), *Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3720 of *Lecture Notes in Computer Science*, pp. 317–328. Springer, 2005.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. COG: connecting new skills to past experience with offline reinforcement learning. *CoRR*, abs/2010.14500, 2020.

Karush Suri, Xiao Qi Shi, Konstantinos Plataniotis, and Yuri Lawryshyn. Trader: Practical deep hierarchical reinforcement learning for trade execution. *arXiv preprint arXiv:2104.00620*, 2021.

Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.

Heyuan Wang, Shun Li, Tengjiao Wang, and Jiayi Zheng. Hierarchical adaptive temporal-relational modeling for stock trend prediction. In *IJCAI*, 2021.

Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *KDD*, pp. 1900–1908, 2019.

Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In Yike Guo and Faisal Farooq (eds.), *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 2447–2456. ACM, 2018.

Min Wen, Ping Li, Lingfei Zhang, and Yan Chen. Stock market trend prediction using high-order information of time series. *IEEE Access*, 7:28299–28308, 2019.

Liguo Weng, Xudong Sun, Min Xia, Jia Liu, and Yiqing Xu. Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism. *Neurocomputing*, 402:171–182, 2020.

Jheng-Long Wu, Chi-Sheng Yang, Kai-Hsuan Liu, and Min-Tzu Huang. A deep learning model for dimensional valencearousal intensity prediction in stock market. In *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, pp. 1–6. IEEE, 2019a.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019b.

Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2141–2149, 2017.

Yueyang Zhong, YeeMan Bergstrom, and Amy Ward. Data-driven market-making via model-free learning. In *IJCAI*, pp. 4461–4468, 2020.