

Additional Supplementary Material

Overview

In this additional supplementary material, we provide several components omitted in our main technical appendix. It includes: (1) The theoretical analysis of the robust TD-learning. (2) The further breakdown of the parameter setting and the grid-search strategy to optimal the optimal parameter. (3) The other implementation details that are related to our robust reinforcement learning setting.

1 Convergence of Robust TD-Learning

Theorem (Convergence of Robust TD-Learning) If the uncertainty set is defined as the ℓ_p -ellipse uncertainty set, then the worst-case transition probability $P_+(\cdot|s, a)$ can be represented as

$$P_+(\cdot|s, a) = P_0(\cdot|s, a) + u^*$$

where β is the radius of the uncertainty set and u^* is solved in Theorem 3.3 or Theorem 3.4.

Proof We apply the following TD-learning update rule:

$$V(s) \leftarrow V(s) + \eta \left(\underbrace{r(s, a) + \gamma V(s') - V(s)}_{\text{stand. TD err. under } P_0} + \gamma V^\top u^* \right). \quad (1)$$

It is easy to observe that this update rule is equivalent to the non-robust TD-learning over the worst-case transition probability:

$$\begin{aligned} & \mathbb{E}_{s' \sim P_0(s'|s, a)} [r(s, a) + \gamma V(s')] + \gamma \langle u^*, V \rangle \\ &= r(s, a) + \gamma \sum_{s', a} P_0(s'|s, a) \pi(a|s) V(s') + \gamma \sum_{s'} u^*(s') V(s') \\ &= r(s, a) + \gamma \sum_{s', a} P_0(s'|s, a) \pi(a|s) V(s') + \gamma \sum_{s', a} \pi(a|s) u^*(s') V(s') \\ &= r(s, a) + \gamma \sum_{s', a} \pi(a|s) [P_0(s'|s, a) + u^*(s')] V(s') \\ &\stackrel{(i)}{=} r(s, a) + \gamma \sum_{s', a} \pi(a|s) P_+(s'|s, a) V(s') \\ &= r(s, a) + \gamma \mathbb{E}_{s' \sim P_+(s'|s, a)} V(s'), \end{aligned}$$

where (i) applies the worst-case transition probability we have derived. Therefore, by applying existing TD-learning convergence analysis, we obtain that the convergence rate is also $\frac{1}{\sqrt{K}}$.

2 Parameter Setting and Grid-Search

We apply grid-search to select hyperparameters to balance learning performance, computational efficiency, and robustness to uncertainty. In this section, we detail the architectural and training hyperparameters used across our three model variants: standard RL, elliptic uncertainty robust RL, and ball-shaped uncertainty robust RL.

2.1 Model Architecture

All models share the same actor-critic network architecture, which consists of:

- **Feature Extractor:** Three fully-connected layers with ReLU activation functions, each containing 256 neurons.
- **Actor Network:** Two fully-connected layers (256 and 128 neurons) with the final layer using a tanh activation function to bound actions within $[-1, 1]$.
- **Critic Network:** Two fully-connected layers (256 and 128 neurons) with a linear output layer.
- **Weight Initialization:** Orthogonal initialization with a gain of $\sqrt{2}$ for linear layers to improve training stability.

This architecture is selected by default and we did not further tune the model architecture.

2.2 Training Parameters

We employ the Proximal Policy Optimization (PPO) algorithm with the following hyperparameters:

Table 1: PPO Training Hyperparameters. The PPO clip parameter ϵ regulates the range of policy updates to ensure stability. Policy update epochs indicate how many times each batch is reused for learning. Batch size is the number of trajectories used per update.

Parameter	Standard RL	Robust RL (Elliptic)	Robust RL (Ball)
Learning Rate	3×10^{-4}	3×10^{-4}	3×10^{-4}
Discount Factor (γ)	0.99	0.99	0.99
PPO Clip Parameter (ϵ)	0.2	0.2	0.2
Policy Update Epochs	10	10	10
Batch Size	64	64	64

We apply the grid-search on the learning rate and the PPO clip parameter; however, the grid-search is not applied to improve the performance. Instead, we apply the grid-search to identify a default parameter group to ensure the algorithm convergence. In the remaining experiments, we will use the same parameter in the robust RL method for fair comparison.

2.3 Robustness Parameters

For our robust RL implementations, we include another group of hyper-parameters.

We grid-search the parameter β from $[0.1, 0.01, 0.001, 0.0001, 0.00001]$ and use the best result on the training period with adopting the early stopping. Then the best model is evaluated in the testing period to obtain the final performance.

Table 2: Robustness Parameters. The robust type is the string used in our codes to determine the different types of uncertainty sets. The parameter β is used to determine the size of uncertainty set.

Parameter	Robust RL (Elliptic)	Robust RL (Ball)
Robust Type	P1N2	P1
Uncertainty Set Parameter (β)	1×10^{-4}	1×10^{-4}
Uncertainty Dimension	3	3
Epsilon	1×10^{-3}	1×10^{-3}

2.4 Implementation Details

The learning rate scheduler employs a ReduceLROnPlateau strategy with a factor of 0.5 and patience of 5 episodes. This adaptive approach reduces the learning rate when performance plateaus. Additionally, gradient clipping with a threshold of 0.5 is applied to prevent exploding gradients and ensure stable training.

The choice of $\beta = 1 \times 10^{-4}$ for both robust methods represents a balance between model performance and robustness. Too large a value would overly emphasize worst-case scenarios, while too small a value would not provide sufficient robustness against uncertainty. Through empirical testing, this value demonstrated the best trade-off between trading performance and resilience to market fluctuations.

Advantage normalization is applied prior to the policy update to stabilize training and improve convergence. The PPO clip parameter $\epsilon = 0.2$ prevents excessively large policy updates, maintaining proximity to the previous policy while allowing sufficient exploration.

Discretization of Execution Prices To apply the robust RL framework, we apply the discretization to the execution price. We introduce two additional hyper-parameter to control the discretization level: $2N + 1$ represents the number of total discretization in the execution prices and δ represents the strength of each level. Given the execution price p , we have $2N + 1$ potential execution prices $[p - N\epsilon, \dots, p - \epsilon, p, p + \epsilon, \dots, p + N\epsilon]$ in total. This discretization approach allows us to directly apply the closed-form solution to solve the optimal u^* . The uncertainty dimension and the epsilon in the robustness parameters are used to determine the discretization level of execution prices. For example, when the discretization level is given by 3, we have three potential execution prices $[p + \epsilon, p, p - \epsilon]$, which also corresponds to the uncertainty dimension. In the nominal transition kernel, we assume the distribution over this discretized set as $[0.25, 0.5, 0.25]$, which provides sufficient flexibility for us to choose the uncertainty u .

3 Other Implementation Details

Her, we introduce the implementation details in calculating the worst-case uncertainty u^* .

Uncertainty Restriction to Execution Prices Instead of perturbing the whole transition probability, we restrict the perturbation on the execution price. The formulation of restriction is given as follows: we denote the state s_t as two components (p_t, f_t) , and we hope perturb the transition on p_t only. We re-write the transition probability using the conditional probability

$$\begin{aligned}\mathbb{P}(s_{t+1} \mid s_t, a_t) &= \mathbb{P}(p_{t+1}, f_{t+1} \mid s_t, a_t) \\ &= \mathbb{P}(p_{t+1} \mid s_t, a_t) \mathbb{P}(f_{t+1} \mid p_{t+1}, s_t, a_t).\end{aligned}$$

Then we set the uncertainty u as the perturbation on the kernel $\mathbb{P}(p_{t+1} \mid s_t, a_t)$ instead of the whole transition kernel $\mathbb{P}(s_{t+1} \mid s_t, a_t)$. Given that the transition of the execution price p_{t+1}

is discretized, the transition probability $\mathbb{P}(p_{t+1} \mid s_t, a_t)$ is a discrete distribution. For example, when taking the uncertainty dimension as 3, we in fact obtain: $\mathbb{P}(p_{t+1} = p + \epsilon \mid s_t, a_t) = 0.25 + u^1$, $\mathbb{P}(p_{t+1} = p \mid s_t, a_t) = 0.25 + u^2$, and $\mathbb{P}(p_{t+1} = p - \epsilon \mid s_t, a_t) = 0.25 + u^3$. By default, in the P1N2-type robust RL, we choose the shift parameter $u_1 = [u^1, u^2, u^3] = [0.1 - 1/3, -1/3, -1/3]$ if the action is buying a stock and $u_1 = [u^1, u^2, u^3] = [-1/3, -1/3, 0.1 - 1/3]$ if the action is selling a stock.

Solving the Worst-Case Uncertainty After restricting the domain to the discrete execution price, we turn the original worst-case Bellman equation into the following form:

$$\begin{aligned}
V^\pi(s) &= r(s, a) + \gamma \min_u \sum_{s'} \sum_a V^\pi(s') \pi(a|s) \mathbb{P}_u(s'|s, a) \\
&= r(s, a) + \gamma \min_u \sum_{s'} \sum_a V^\pi(s') \pi(a|s) \mathbb{P}_u(p' \mid s, a) \mathbb{P}(f' \mid p', s, a) \\
&= r(s, a) + \gamma \min_u \sum_{s'} \sum_a V^\pi(s') \pi(a|s) [\mathbb{P}_0(p' \mid s, a) + u_{s,a}] \mathbb{P}(f' \mid p', s, a) \\
&= r(s, a) + \gamma \sum_{s'} \sum_a V^\pi(s') \pi(a|s) \mathbb{P}_0(s'|s, a) + \gamma \min_u \sum_{s'} \sum_a u_{s,a} V^\pi(s') \pi(a|s) \mathbb{P}(f' \mid p', s, a).
\end{aligned}$$

Moreover, we use $\mathbb{P}(f' \mid p', s, a)$ as a deterministic transition. It is common when training in the historical data, as (1) the observed feature f' comes from the existing dataset and the action a does not change its value (in our implementation, the action a will only affect the execution price), and (2) many features are directly calculated based on the current state, action, and the execution price such as the remaining cash. As the result, we obtain

$$\begin{aligned}
V^\pi(s) &= r(s, a) + \gamma \sum_{s'} \sum_a V^\pi(s') \pi(a|s) \mathbb{P}_0(s'|s, a) + \gamma \min_u \sum_{p', s'=(p', f')} u(p') V^\pi(s') \\
&= r(s, a) + \gamma \sum_{s'} \sum_a V^\pi(s') \pi(a|s) \mathbb{P}_0(s'|s, a) + \gamma \min_u u^\top V^\pi.
\end{aligned}$$

Here we still write $u^\top V^\pi$ for convenience, while the value function V^π represents the $2N + 1$ dimensional vector with value taken on $s' = (p', f')$, where p' takes $2N + 1$ values and f' is deterministically determined by (p', s, a) . After making this modification, we can solve $\min_u u^\top V^\pi$ using Theorem 3.3 and Theorem 3.4, or existing ℓ_p -norm formula.