Fast Value Tracking for Deep Reinforcement Learning

Frank Shih and Faming Liang   Department of Statistics, Purdue University

Motivation

An ideal Deep RL algorithm should exhibit the following characteristics:

- **Uncertainty Quantification**
  Enhances policy robustness by addressing the stochastic nature of agent-environment interactions.
- **Nonlinear approximation**
  Increases the applicability of the algorithm by using DNN to approximate Q-functions.
- **Computational efficiency**
  Facilitates online learning by scaling with respect to model dimensions and training sample size.

Brief Recap of Reinforcement Learning

Q-Learning
Agents learn optimal policies via the state-action function (Q-function), defined as follows:

$$Q_\rho(s,a) = \mathbb{E}_\rho[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$$

where ρ represents the policy and γ is the discount factor. The Q-function can be learned by minimizing the Mean Squared Bellman Error (MSBE):

$$\min_\rho \mathbb{E}_{(s,a,r,s',a')\sim\rho}[(Q_\rho(s,a) - r - \gamma Q_\rho(s',a'))^2]$$

Deep Q-Network (DQN)
The DQN algorithm utilizes a deep neural network to approximate the Q-function as Q_θ, with weights θ. These weights can be updated via the semi-gradient of the Temporal Difference (TD) error:

$$\theta_t = \theta_{t-1} - \epsilon_t[Q_\theta(s,a) - r - \gamma \max_a Q_{\theta'}(s',a)]\nabla_\theta Q_\theta(s,a),$$

where θ' denotes the target network weights. The DQN algorithm is effective in solving deep reinforcement learning problems using optimization algorithms such as SGD, Adam, and RMSprop. However, due to the nature of these optimization algorithms, DQN may fail to accurately quantify uncertainties.

✔ Nonlinear approximation, computational efficiency.
✘ Uncertainty quantification.

Kalman Temporal Difference (KTD)
The KTD algorithm solves RL problems under the Kalman Filter paradigm through a state-space reformulation:

$$\begin{aligned}\theta_t &= \theta_{t-1} + w_t,\\ \boldsymbol{r}_t &= h(\boldsymbol{x}_t, \theta_t) + \eta_t,\end{aligned}$$

where $\boldsymbol{x}_t = (s, a, s', a')$ is the transition tuple generated by the policy at time t, and $h(\boldsymbol{x}_t, \theta_t) = Q_{\theta_t}(s,a) - \gamma Q_{\theta_t}(s',a')$. The KTD framework operates under the assumption of Gaussian distributions and linear measurement functions. To address the nonlinearity of deep neural networks, linearization techniques are required. However, KTD encounters several challenges:

(i) Accuracy for the true filtering distribution of θ_t is unknown
(ii) High computational complexity $O(np^2)$
(iii) High memory complexity, necessitating $O(p^2)$ additional space for the covariance matrix.

✔ Uncertainty quantification.
✘ Nonlinear approximation, computational efficiency.

State-Space Model Reformulation

To address the limitations of DQN and KTD, we reformulate RL using the following State-Space Model:

$$\begin{aligned}\theta_t &= \theta_{t-1} + \frac{\epsilon_t}{2}\nabla_\theta \log \pi(\theta_{t-1}) + w_t,\\ r_t &= h(x_t, \theta_t) + \eta_t,\end{aligned} \tag{1}$$

where $w_t \sim N(0, \epsilon_t I_p)$, $\pi(\theta)$ represents a prior density, and ϵ_t is the decaying learning rate. With the formulation (1) and our algorithm, θ_t converges to a proper distribution as $t \to \infty$, enabling the uncertainty to be properly quantified. Inclusion of the prior information in the state evolution equation generally robustifies the performance of the RL algorithm.

Langevinized Kalman Temporal Difference (LKTD)

The LKTD algorithm is designed to solve equation (1). We apply the variance splitting technique to convert the model (1) into a state-space model with a linear measurement equation, while allowing the state evolution equation to be nonlinear. WLOG, we assume $\eta_t \sim N(0, \sigma^2 I_n)$ for each stage t. By the state augmentation approach, we define

$$\varphi_t = \begin{pmatrix}\theta_t\\ \xi_t\end{pmatrix}, \quad \xi_t = h(\boldsymbol{x}_t;\theta_t) + u_t, \quad u_t \sim N(0, \alpha\sigma^2 I_n),$$

where ξ_t is an n-dimensional vector, and $0 < \alpha < 1$ is a pre-specified constant. Suppose that θ_t has a prior distribution $\pi(\theta)$ as specified previously, the joint density function of $\varphi_t = (\theta_t^\top, \xi_t^\top)^\top$ is given by $\pi(\varphi_t) = \pi(\theta_t)\pi(\xi_t|\theta_t)$, where $\xi_t|\theta_t \sim N(h(\boldsymbol{x}_t;\theta_t), \alpha\sigma^2 I)$. Based on Langevin dynamics, we can reformulate (1) as the following model:

$$\begin{aligned}\varphi_t &= \varphi_{t-1} + \frac{\epsilon_t}{2}\frac{n}{\mathcal{N}}\nabla_\varphi \log \pi(\varphi_{t-1}) + \tilde{w}_t,\\ \boldsymbol{r}_t &= H_t\varphi_t + v_t,\end{aligned}$$

where $\mathcal{N} > 0$, $\tilde{w}_t \sim N(0, \frac{n}{\mathcal{N}}B_t)$, $B_t = \epsilon_t I_{\tilde{p}}$, $\tilde{p} = p + n$ is the dimension of φ_t; $H_t = (\mathbf{0}, I_n)$ such that $H_t\varphi_t = \xi_t$; $v_t \sim N(0, (1-\alpha)\sigma^2 I_n)$, which is independent of $\tilde{w}_t$ for all t. We call $\mathcal{N}$ the pseudo-population size, which scales uncertainty of the estimator of the system.

Training Algorithm

Algorithm 1. *(Langevinized Kalman temporal difference for nonlinear approximation)*

0. *(Initialization) Draw $\theta_0^a \in \mathbb{R}^p$ drawn from the prior distribution $\pi(\theta)$. For each stage $t = 1, 2, \dots, T$, do the following steps 1-4:*
1. *(Sampling) With policy $\rho_{\theta_{t-1}^a}$, generate a set of n transition tuples, denoted by $\boldsymbol{z}_t = (\boldsymbol{r}_t, \boldsymbol{x}_t) := \{r_t^{(j)}, x_t^{(j)}\}_{j=1}^n$, where $x_t^{(j)} = (s_t^{(j)}, a_t^{(j)}, s_{t+1}^{(j)}, a_{t+1}^{(j)})^T$. For each iteration $k = 1, \dots, \mathcal{K}$, do the following steps 2-4:*
2. *(Presetting) Set $B_{t,k} = \epsilon_{t,k} I_{\tilde{p}}$, $R_t = 2(1-\alpha)\sigma^2 I$, and Kalman gain $K_{t,k} = B_{t,k}H_t^\top(H_t B_{t,k} H_t^\top + R_t)^{-1}$*
3. *(Forecast) Draw $\tilde{w}_{t,k} \sim N_p(0, \frac{n}{\mathcal{N}}B_{t,k})$ and calculate*

$$\varphi_{t,k}^f = \varphi_{t,k-1}^a + \frac{\epsilon_{t,k}}{2}\frac{n}{\mathcal{N}}\nabla_\varphi \log \pi(\varphi_{t,k-1}^a) + \tilde{w}_{t,k},$$

where $\varphi_{t,0}^a = (\theta_{t-1,\mathcal{K}}^{a\;\top}, \boldsymbol{r}_t^\top)^\top$ if $k = 1$, and the gradient term is given by

$$\nabla_\varphi \log \pi(\varphi_{t,k-1}^a) = \begin{pmatrix}\nabla_\theta \log \pi(\theta_{t,k-1}) + \frac{1}{\alpha\sigma^2}\frac{\mathcal{N}}{n}\nabla_\theta h(\boldsymbol{x}_t;\theta_{t,k-1})(\xi_{t,k-1} - h(\boldsymbol{x}_t;\theta_{t,k-1}))\\ -\frac{1}{\alpha\sigma^2}(\xi_{t,k-1} - h(\boldsymbol{x}_t;\theta_{t,k-1}))\end{pmatrix}.$$

4. *(Analysis) Draw $v_{t,k} \sim N_n(0, \frac{n}{\mathcal{N}}R_t)$ and calculate*

$$\varphi_{t,k}^a = \varphi_{t,k}^f + K_{t,k}(\boldsymbol{r}_t - H_t\varphi_{t,k}^f - v_{t,k}) = \varphi_{t,k}^f + K_{t,k}(\boldsymbol{r}_t - \boldsymbol{r}_{t,k}^f).$$

Theory

As shown in our paper, Algorithm 1 is an preconditioned SGLD algorithm. To establish the convergence theory, it suffices to prove the convergence of the following SGLD sampler in the RL context:

$$\theta_k = \theta_{k-1} + \epsilon_k G(\theta_{k-1}, \boldsymbol{z}_k) + \sqrt{2\beta^{-1}\epsilon_k}\mathfrak{e}_k, \tag{2}$$

where $\mathfrak{e}_k \sim N(0, I_d)$, β is the inverse temperature, and $\boldsymbol{z}_k$ is the transition tuple.

Theorem 1. *Consider the SGLD sampler (2) with a polynomailly-decay learning rate $\epsilon_k = \frac{\epsilon_0}{k^\varpi}$ for some $\varpi \in (0,1)$. Suppose the environment is stationary and some mild conditions hold. If $\mathbb{E}(G(\theta_{k-1}, z_k)) = g(\theta_{k-1})$ holds for any stage $k \in \{1, \dots, K\}$, $\beta \geq 1 \vee \frac{2}{m_U}$, then there exist constants (C_0, C_1, C_2, C_3) independent of the learning rates such that for all $K \in \mathbb{N}$, the 2-Wasserstein distance between μ_K and $\nu_{\mathcal{N}}$ can be upper bounded by*

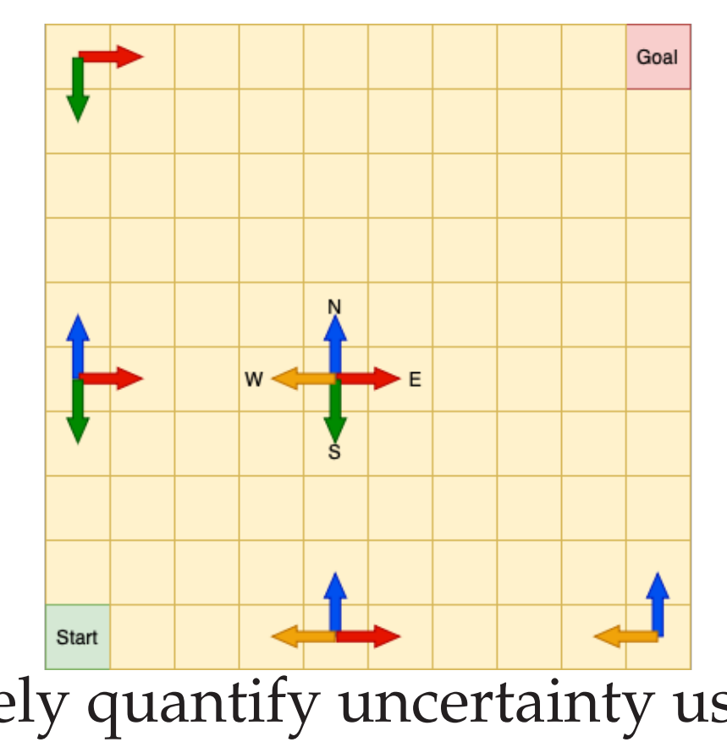
$$\begin{aligned}\mathcal{W}_2(\mu_K, \nu_{\mathcal{N}}) \leq{}& (12 + C_2\epsilon_0(\frac{1}{1-\varpi}K^{1-\varpi}))^{\frac{1}{2}} \cdot [(C_1\epsilon_0^2(\frac{2\varpi}{2\varpi-1}) + \delta C_0(\frac{\epsilon_0}{1-\varpi}K^{1-\varpi}))^{\frac{1}{2}}\\ &+ (C_1\epsilon_0^2(\frac{2\varpi}{2\varpi-1}) + \delta C_0(\frac{\epsilon_0}{1-\varpi}K^{1-\varpi}))^{\frac{1}{4}}] + C_3\exp\left(-\frac{1}{\beta c_{LS}}(\frac{\epsilon_0}{1-\varpi}K^{1-\varpi})\right),\end{aligned}$$

where $\mu_K(\theta)$ denotes the probability law of θ_K, $\nu_{\mathcal{N}}(\theta) \propto \exp(-\beta\mathcal{G}(\theta))$, $\mathcal{G}(\theta) = O(\mathcal{N})$ is the anti-derivative of $g(\theta)$, i.e., $\nabla_\theta\mathcal{G}(\theta) = g(\theta)$, and c_{LS} denotes a logarithmic Sobolev constant satisfied by the $\nu_{\mathcal{N}}$.

Numerical Results

Escape Environment

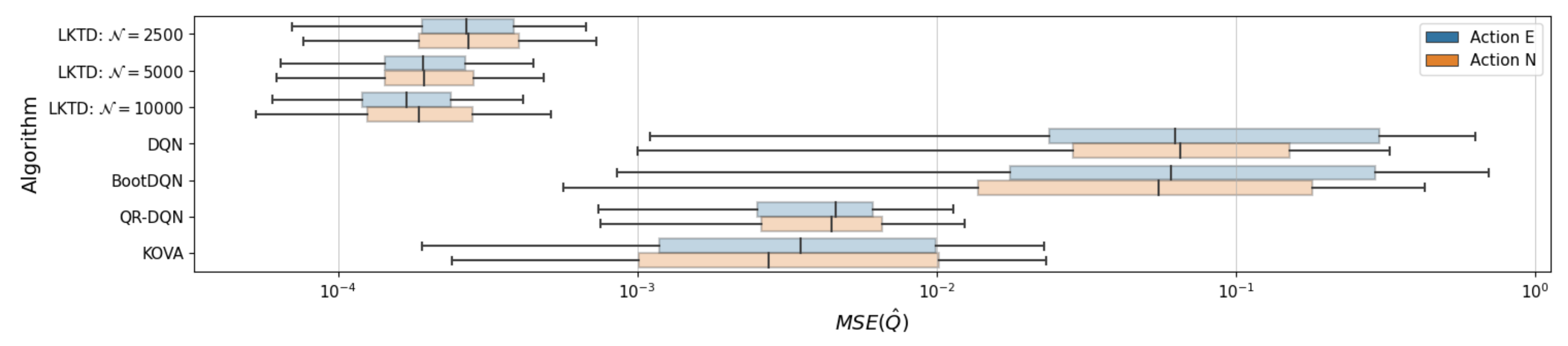
- State space: $\mathcal{S} = \{(x,y) : x, y \in \{1, \dots, 10\}\}$
- Action space: $\mathcal{A} = \{N, E, S, W\}$
- Reward: $\mathcal{N}(-1, 0.01)$
- Discount factor: $\gamma = 0.9$

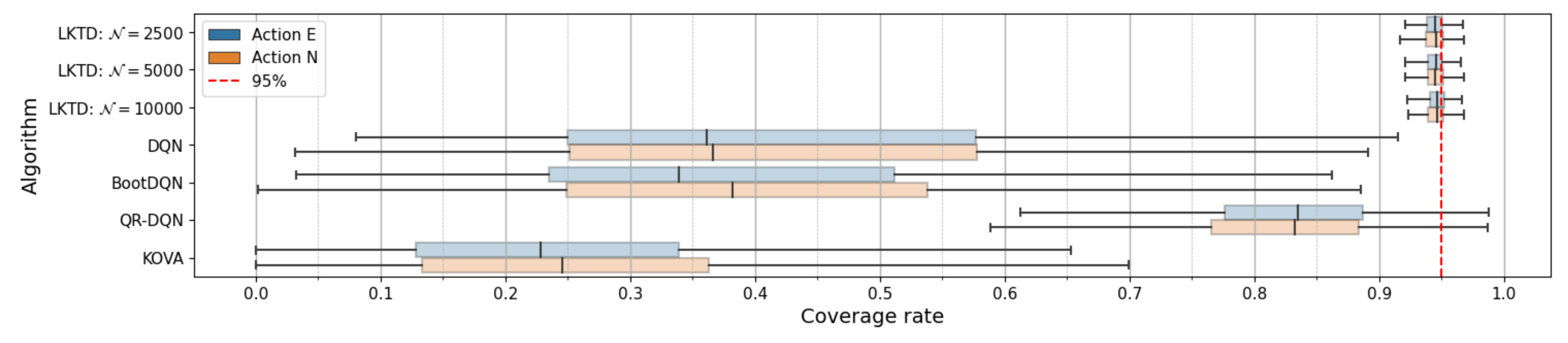


In this toy example, we demonstrate the ability to accurately quantify uncertainty using the following metrics:

- MSE: Measures the accuracy of Q-value estimation.
- CR: Indicates the coverage rate of the 95% prediction interval.

The figures below illustrate that the LKTD algorithm outperforms competing algorithms in terms of estimation accuracy and uncertainty quantification. These experiments were replicated 100 times and are represented in the boxplot.



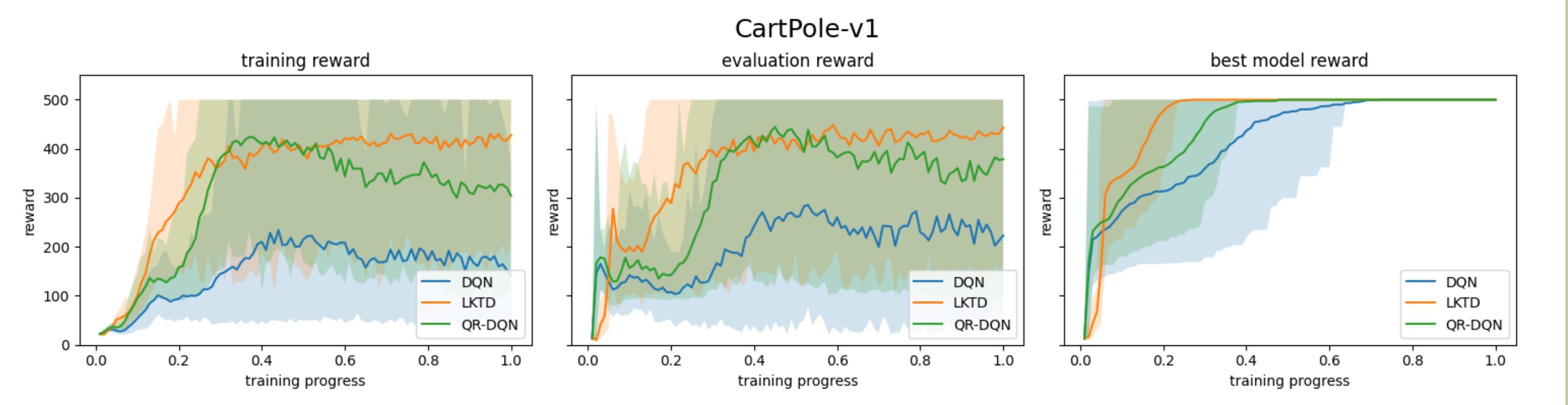


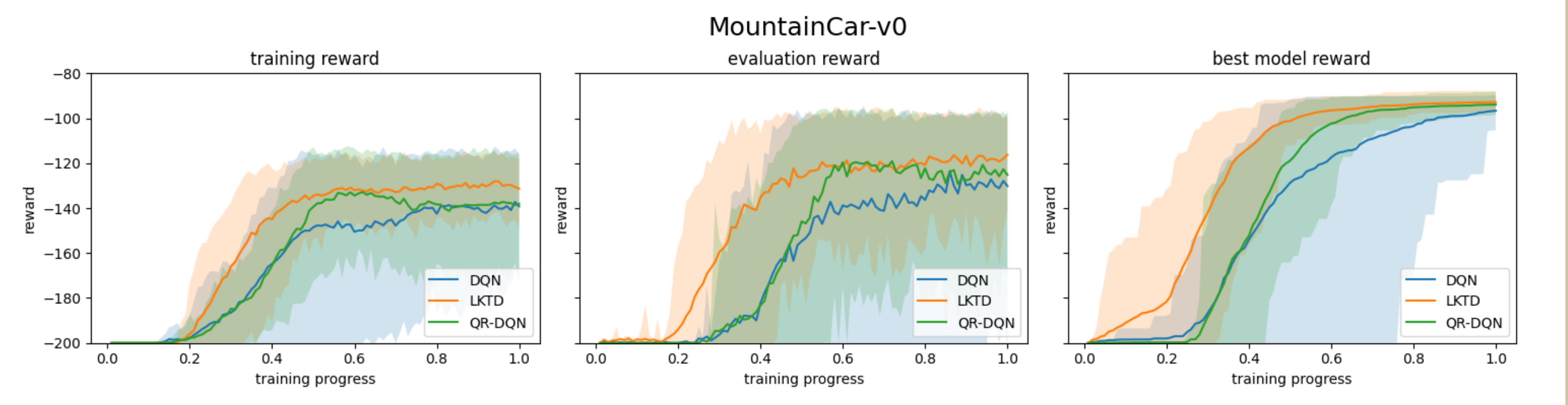
Classic Control Problem (OpenAI Gym)

This experiment showcases the performance of LKTD compared to DQN and Quantile Regression DQN (QR-DQN) across three metrics:

1. Training reward: The reward score achieved through training phase.
2. Evaluation reward: The reward evaluated for the current model.
3. Best model reward: The reward evaluated for the best-learned model.

The figures below demonstrates that LKTD achieves higher training and evaluation rewards compared to DQN and QR-DQN. In terms of robustness, LKTD exhibits a higher lower bound for reward curves and achieves optimal policies faster than the other algorithms.


CartPole-v1
training reward
evaluation reward
best model reward
reward
training progress
DQN
LKTD
QR-DQN


MountainCar-v0
training reward
evaluation reward
best model reward
reward
training progress
DQN
LKTD
QR-DQN