DeepTOP: Deep Threshold-Optimal Policy for MDPs and RMABs

Khaled Nakhleh, I-Hong Hou

Texas A&M University, College Station

Neural Information Processing Systems (NeurIPS) 2022

Overview

- Threshold Policies for MDPs
- Notations
- Threshold Policy Gradient Theorem
- Extension to RMABs and its Policy Gradient
- MDP Results
- RMAB Results
- Conclusion and Future Work

Motivating examples

- Charging an Electric Vehicle (EV) given changing electricity prices.

Motivating examples

- Charging an Electric Vehicle (EV) given changing electricity prices.
- AC system deciding whether to cool a building or not.

Motivating examples

- Charging an Electric Vehicle (EV) given changing electricity prices.
- AC system deciding whether to cool a building or not.
- Central bank deciding whether to raise interest rate given the current inflation rate or not.

Motivating examples

- Charging an Electric Vehicle (EV) given changing electricity prices.
- AC system deciding whether to cool a building or not.
- Central bank deciding whether to raise interest rate given the current inflation rate or not.
- In all cases, the agent has a state v_t at time t. State contains info: current fuel level, number of people in the building, current inflation rate, etc.

Motivating examples

- Charging an Electric Vehicle (EV) given changing electricity prices.
- AC system deciding whether to cool a building or not.
- Central bank deciding whether to raise interest rate given the current inflation rate or not.
- In all cases, the agent has a state v_t at time t. State contains info: current fuel level, number of people in the building, current inflation rate, etc.
- Electricity price, temperature, or inflation rates are represented as λ_t.

Exploiting the inherent structure in those problems

- Suppose we have a *threshold function* $\mu : \mathcal{V} \rightarrow \mathcal{R}$.

Exploiting the inherent structure in those problems

- Suppose we have a *threshold function* $\mu : \mathcal{V} \to \mathcal{R}$.
- In the previous examples, it would be optimal to *activate* (i.e. $a_t = 1$) if the assigned value $\mu(v_t) > \lambda_t$.

Exploiting the inherent structure in those problems

- Suppose we have a *threshold function* $\mu : \mathcal{V} \to \mathcal{R}$.
- In the previous examples, it would be optimal to *activate* (i.e. $a_t = 1$) if the assigned value $\mu(v_t) > \lambda_t$.
- intuitively, it is optimal to charge the car if its current state (e.g. low fuel level and large distance until destination). The policy deterministically picks the action $a_t = \mathbb{1}(\mu(v_t) > \lambda_t)$.

Exploiting the inherent structure in those problems

- Suppose we have a *threshold function* $\mu : \mathcal{V} \to \mathcal{R}$.
- In the previous examples, it would be optimal to *activate* (i.e. $a_t = 1$) if the assigned value $\mu(v_t) > \lambda_t$.
- intuitively, it is optimal to charge the car if its current state (e.g. low fuel level and large distance until destination). The policy deterministically picks the action $a_t = \mathbb{1}(\mu(v_t) > \lambda_t)$.
- Threshold policies actions are *monotone*. E.g. If the HVAC turns on for a certain temperature, then it would also turn on for higher temperatures.

Related Work

[Hegde2011] Nidhi Hegde, Laurent Massoulié, Theodoros Salonidis, et al. Optimal control of residential energy storage under price fluctuations. Energy, 2011.

[Yao2021] Guidan Yao, Ahmed M. Bedewy, and Ness B. Shroff. 2021. Battle between Rate and Error in Minimizing Age of Information. In Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21).

[Borkar2021] Arghyadip Roy, Vivek S. Borkar, Abhay Karandikar, and Prasanna Chaporkar. Online reinforcement learning of optimal threshold policies for markov decision processes. IEEE Transactions on Automatic Control, pages 1–1, 2021.

Objective and contribution

Deep RL algorithms would asymptotically learn the optimal policy, however they require more transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t=1}^{B}$ stored into a memory $\mathcal{M}$. We can have a more efficient algorithm by learning the optimal threshold function and in return, the optimal threshold policy.

Objective and contribution

Deep RL algorithms would asymptotically learn the optimal policy, however they require more transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t=1}^{B}$ stored into a memory $\mathcal{M}$. We can have a more efficient algorithm by learning the optimal threshold function and in return, the optimal threshold policy.

Contributions

- For MDPs that admit a threshold policy, we find a simple expression for the threshold policy gradient.

Objective and contribution

Deep RL algorithms would asymptotically learn the optimal policy, however they require more transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t=1}^{B}$ stored into a memory $\mathcal{M}$. We can have a more efficient algorithm by learning the optimal threshold function and in return, the optimal threshold policy.

Contributions

- For MDPs that admit a threshold policy, we find a simple expression for the threshold policy gradient.
- We extend the optimal threshold policy gradient theorem to the Restless multi-armed bandits (RMAB) framework.

Objective and contribution

Deep RL algorithms would asymptotically learn the optimal policy, however they require more transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t=1}^{B}$ stored into a memory $\mathcal{M}$. We can have a more efficient algorithm by learning the optimal threshold function and in return, the optimal threshold policy.

Contributions

- For MDPs that admit a threshold policy, we find a simple expression for the threshold policy gradient.
- We extend the optimal threshold policy gradient theorem to the Restless multi-armed bandits (RMAB) framework.
- We use these gradient expressions to have two off-policy, model-free deep RL algorithms.

Definitions for the MDP case

Define a Markov Decision Process as $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$.

Definitions for the MDP case

Define a Markov Decision Process as $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$.

Binary action space $\mathcal{A} = \{0, 1\}$. State space $\mathcal{S} = \mathbb{R} \times \mathcal{V}$ with the scalar state $\lambda_t \in \mathbb{R}$ and $v \in \mathcal{V}$ being a *discrete* set of vectors. Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Omega$ with Ω being the set of random variables. Discount factor $\gamma \in [0, 1)$.

Definitions for the MDP case

Define a Markov Decision Process as $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$.

Binary action space $\mathcal{A} = \{0, 1\}$. State space $\mathcal{S} = \mathbb{R} \times \mathcal{V}$ with the scalar state $\lambda_t \in \mathbb{R}$ and $v \in \mathcal{V}$ being a *discrete* set of vectors. Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Omega$ with Ω being the set of random variables. Discount factor $\gamma \in [0, 1)$.

MDP generates a reward r_t according to an unknown random variable $\mathcal{R}(s_t, a_t)$. Denote $\bar{r}(\lambda, v, a) = \mathbb{E}\big[\mathcal{R}((\lambda, v), a)\big]$ to be the unknown expected one-step reward.

The MDP objective function

Define the state-action value function under threshold function μ as

$$Q_\mu\Big(\lambda, v, \mathbb{1}(\mu(v) > \lambda)\Big) := \sum_{v' \in \mathcal{V}} \int_{\lambda'=-M}^{\lambda'=+M} \rho_\mu(\lambda', v', \lambda, v) \bar{r}\big(\lambda', v', \mathbb{1}(\mu(v') > \lambda')\big). \tag{1}$$

With $\rho_\mu(\lambda', v', \lambda, v)$ being the discounted state distribution.

The MDP objective function

Define the state-action value function under threshold function μ as

$$Q_\mu\Big(\lambda, v, \mathbb{1}(\mu(v) > \lambda)\Big) := \sum_{v' \in \mathcal{V}} \int_{\lambda'=-M}^{\lambda'=+M} \rho_\mu(\lambda', v', \lambda, v) \bar{r}\big(\lambda', v', \mathbb{1}(\mu(v') > \lambda')\big). \tag{1}$$

With $\rho_\mu(\lambda', v', \lambda, v)$ being the discounted state distribution.

Learn the optimal threshold function parametrized by ϕ that maximizes the objective function

$$K(\mu^\phi) := \int_{\lambda=-M}^{\lambda=+M} \sum_{v \in \mathcal{V}} Q_{\mu^\phi}\Big(\lambda, v, \mathbb{1}(\mu^\phi(v) > \lambda)\Big) d\lambda. \tag{2}$$

The MDP objective function

$$K(\mu^{\phi}) := \int_{\lambda=-M}^{\lambda=+M} \sum_{v \in \mathcal{V}} Q_{\mu^{\phi}} \Big(\lambda, v, \mathbb{1}(\mu^{\phi}(v) > \lambda)\Big) d\lambda.$$

- Computing the gradient of $K(\mu^{\phi})$ involves an integral over $\lambda \in [-M, +M]$.

The MDP objective function

$$K(\mu^{\phi}) := \int_{\lambda=-M}^{\lambda=+M} \sum_{v \in \mathcal{V}} Q_{\mu^{\phi}} \Big(\lambda, v, \mathbb{1}(\mu^{\phi}(v) > \lambda)\Big) d\lambda.$$

- Computing the gradient of $K(\mu^{\phi})$ involves an integral over $\lambda \in [-M, +M]$.
- Exploit the actions' monotone property of threshold policies to find a simple expression.

Theorem 1: threshold policy gradient theorem for MDPs

Given the parameter vector ϕ, let $\bar{\rho}(\lambda, v)$ be the discounted state distribution when the initial state is chosen uniformly at random under the threshold policy. If all vector states $v \in \mathcal{V}$ have distinct values of $\mu^{\phi}(v)$, then,

$$\nabla_{\phi} K(\mu^{\phi}) = 2M|\mathcal{V}| \sum_{v \in \mathcal{V}} \bar{\rho}(\mu^{\phi}(v), v)\Big(Q_{\mu^{\phi}}(\mu^{\phi}(v), v, 1) - Q_{\mu^{\phi}}(\mu^{\phi}(v), v, 0)\Big)\nabla_{\phi}\mu^{\phi}(v). \tag{3}$$

Proof sketch for theorem 1

Let $\bar{\rho}_t(\lambda, v)$ be the distribution that the state at time t is (λ, v) when the initial state is chosen uniformly at random. Let $\mathbb{M}^0 = +M$, $\mathbb{M}^n = \mu^\phi(v^n)$, for all $1 \leq n \leq |\mathcal{V}|$, and $\mathbb{M}^{|\mathcal{V}|+1} = -M$. For any vector state v, the threshold policy would take the same action under all $\lambda \in (\mathbb{M}^{n+1}, \mathbb{M}^n)$, and we use $\pi^{n+1}(v)$ to denote this action.

Proof sketch for theorem 1

$$\nabla_\phi K(\mu^\phi) = \nabla_\phi \int_{\lambda=-M}^{\lambda=+M} \sum_{v\in\mathcal{V}} Q_{\mu^\phi}(\lambda, v, \mathbb{1}(\mu^\phi(v) > \lambda))d\lambda$$

$$= \sum_{v\in\mathcal{V}} \nabla_\phi \int_{\lambda=-M}^{\lambda=+M} Q_{\mu^\phi}(\lambda, v, \mathbb{1}(\mu^\phi(v) > \lambda))d\lambda \quad \text{(Fubini-Tonelli theorem)}$$

$$= \sum_{v\in\mathcal{V}} \sum_{n=0}^{|\mathcal{V}|} \nabla_\phi \int_{\lambda=\mathbb{M}^{n+1}}^{\lambda=\mathbb{M}^n} Q_{\mu^\phi}(\lambda, v, \pi^{n+1}(v))d\lambda$$

$$= \sum_{v\in\mathcal{V}} \sum_{n=0}^{|\mathcal{V}|} \Big(Q_{\mu^\phi}\left(\mathbb{M}^n, v, \pi^{n+1}(v)\right)\nabla_\phi \mathbb{M}^n - Q_{\mu^\phi}\left(\mathbb{M}^{n+1}, v, \pi^{n+1}(v)\right)\nabla_\phi \mathbb{M}^{n+1}$$

$$+ \int_{\lambda=\mathbb{M}^{n+1}}^{\lambda=\mathbb{M}^n} \nabla_\phi Q_{\mu^\phi}(\lambda, v, \pi^{n+1}(v))d\lambda \Big), \quad \text{(Leibniz integral rule)} \quad (4)$$

DeepTOP-MDP algorithm steps

Simplify the first term by

$$\sum_{v\in\mathcal{V}}\sum_{n=0}^{|\mathcal{V}|}\left(Q_{\mu^\phi}\left(\mathbb{M}^n, v, \pi^{n+1}(v)\right)\nabla_\phi \mathbb{M}^n - Q_{\mu^\phi}\left(\mathbb{M}^{n+1}, v, \pi^{n+1}(v)\right)\nabla_\phi \mathbb{M}^{n+1}\right)$$

$$=\sum_{v\in\mathcal{V}}\sum_{n=1}^{|\mathcal{V}|}\Big(Q_{\mu^\phi}\left(\mu^\phi(v^n), v, \mathbb{1}(v\in\mathbb{V}^{n+1})\right) - Q_{\mu^\phi}\left(\mu^\phi(v^n), v, \mathbb{1}(v\in\mathbb{V}^{n})\right)\Big)\nabla_\phi\mu^\phi(v^n)$$

$$=2M|\mathcal{V}|\sum_{v\in\mathcal{V}}\bar{\rho}_1(\mu^\phi(v), v)\Big(Q_{\mu^\phi}\left(\mu^\phi(v), v, 1\right) - Q_{\mu^\phi}\left(\mu^\phi(v), v, 0\right)\Big)\nabla_\phi\mu^\phi(v). \tag{5}$$

Expanding the second term in the same way gives the gradient expression.

DeepTOP-MDP algorithm design

DeepTOP-MDP as an off-policy, model-free algorithm.

We maintain four neural network parameters: actor ϕ, actor-target ϕ', critic θ, and critic-target θ'.

Critic network learns the parametrized action-value function $Q_\mu^\theta(\lambda, v, a)$ under threshold function μ using a minibatch of transitions from memory $\mathcal{M}$.

The estimated actor gradient is

$$\hat{\nabla}_\phi K(\mu^\phi) := \frac{1}{B}\sum_{k=1}^{B}\left(Q_{\mu^\phi}^\theta\left(\mu^\phi(v_{t_k}), v_{t_k}, 1\right) - Q_{\mu^\phi}^\theta\left(\mu^\phi(v_{t_k}), v_{t_k}, 0\right)\right)\nabla_\phi \mu^\phi(v_{t_k}). \tag{6}$$

Revisiting the EV charging example

You have N EVs parked at a charging station and you can only select V EVs for charging.

Revisiting the EV charging example

You have N EVs parked at a charging station and you can only select V EVs for charging.

Decompose the problem into N sub-problems and find the optimal threshold function for each MDP (or arm).

Revisiting the EV charging example

You have N EVs parked at a charging station and you can only select V EVs for charging.

Decompose the problem into N sub-problems and find the optimal threshold function for each MDP (or arm).

Form an alternative control problem for the Restless Multi-Armed Bandits (RMABs).

Objective function for the RMAB case

Form an alternative control problem where the agent pays an *activation cost* $\lambda \in [-M, +M]$ if it activates the arm i. Net-reward at time t is $r_{i,t} - \lambda a_{i,t}$. For each arm i, define the objective function of maximizing

$$K_i(\mu_i^{\phi_i}) := \int_{\lambda=-M}^{\lambda=+M} \sum_{s_i \in \mathcal{S}_i} Q_{i,\lambda}\big(s_i, \mathbb{1}(\mu_i^{\phi_i}(s_i) > \lambda)\big) d\lambda. \qquad (7)$$

Optimal threshold policy for RMABs

Theorem 2: threshold policy gradient theorem for RMABs

Given the parameter vector ϕ_i, let $\bar{\rho}_\lambda(s_i)$ be the discounted state distribution when the initial state is chosen uniformly at random and the activation cost is λ. If all states $s_i \in \mathcal{S}_i$ have distinct values of $\mu_i^{\phi_i}(s_i)$, then,

$$\nabla_{\phi_i} K_i(\mu_i^{\phi_i}) = |\mathcal{S}_i| \sum_{s_i \in \mathcal{S}_i} \bar{\rho}_{\mu_i^{\phi_i}(s_i)}(s_i) \Big(Q_{i,\mu_i^{\phi_i}(s_i)}(s_i, 1) - Q_{i,\mu_i^{\phi_i}(s_i)}(s_i, 0) \Big) \nabla_{\phi_i} \mu_i^{\phi_i}(s_i). \tag{8}$$

DeepTOP-RMAB algorithm design

Two main differences from DeepTOP-MDP

DeepTOP-RMAB algorithm design

Two main differences from DeepTOP-MDP

- Gradient update steps for the critic and actor are done independently for each arm $i = 1, 2, \ldots, N$.
- Value of λ is an artificial value that exists in the alternative problem. Sampled for each arm from the range $[-M, +M]$.

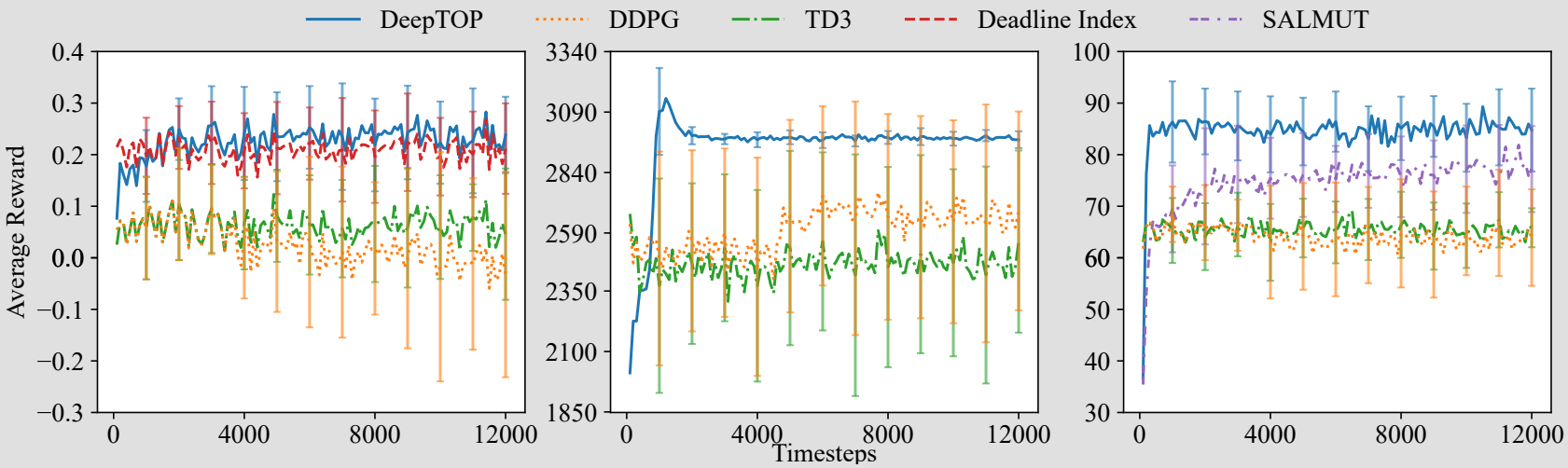
Problems' descriptions

EV charging: station decides whether it is optimal to charge the EV or not based on its state at time t.

Inventory management: manager decides if it is optimal to buy additional goods based on the season's fluctuations and in-lead times in order.

Make-to-stock production: system that produces m items with W demand classes and buffer size s. System determines if it will accept class orders or not.

MDP results



(a) EV charging. (b) Inventory management. (c) Make-to-stock production.

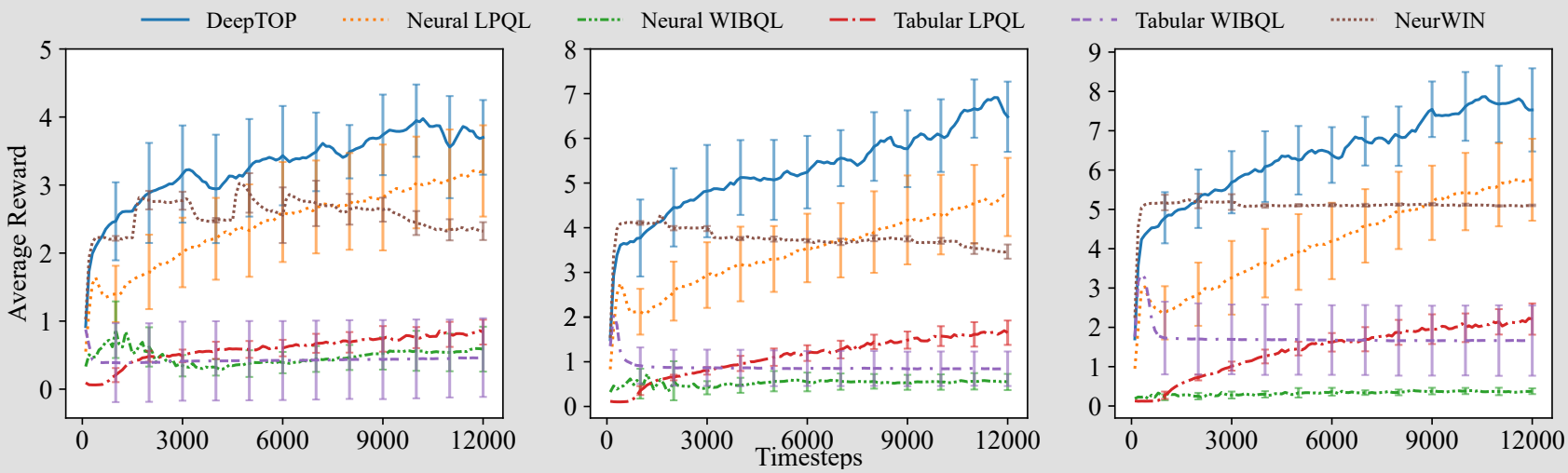
Figure: Average reward results for the MDP problems.

RMAB problems' descriptions

One-dimensional bandits: each arm has 100 states with the reward depending on the current state of arm i. If arm i is activated, next state is $\min\{s_{i,t}+1, 99\}$ with probability p_i. Otherwise, next state is $\max\{s_{i,t}-1, 0\}$ with probability q_i.

Recovering bandits: RMAB that captures the varying behavior of customers on advertisement links. If an arm is activated, the next state $s_{i,t}$ is reset to state 1. Otherwise, the state increases by 1.

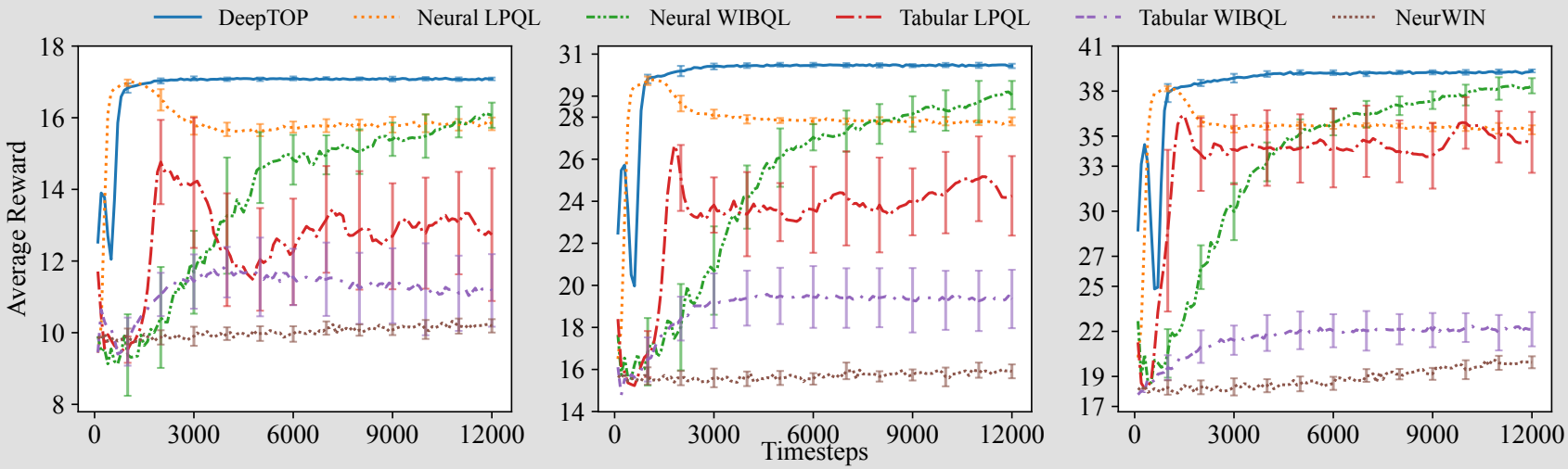
Results



(a) $N = 10.\ V = 3.$ (b) $N = 20.\ V = 5.$ (c) $N = 30.\ V = 6.$

Figure: Average reward results for the one-dimensional bandits.

Results



(a) $N = 10$. $V = 3$. (b) $N = 20$. $V = 5$. (c) $N = 30$. $V = 6$.

Figure: Average reward results for the recovering bandits.

- Presented a simple to compute gradient for threshold functions to obtain the optimal threshold policy.

- Presented a simple to compute gradient for threshold functions to obtain the optimal threshold policy.
- Designed two algorithms: DeepTOP-MDP and DeepTOP-RMAB for MDPs that admit a threshold policy and RMABs.

- Presented a simple to compute gradient for threshold functions to obtain the optimal threshold policy.
- Designed two algorithms: DeepTOP-MDP and DeepTOP-RMAB for MDPs that admit a threshold policy and RMABs.
- The two algorithms outperform the baselines.

- Presented a simple to compute gradient for threshold functions to obtain the optimal threshold policy.
- Designed two algorithms: DeepTOP-MDP and DeepTOP-RMAB for MDPs that admit a threshold policy and RMABs.
- The two algorithms outperform the baselines.
- Future direction is extending the threshold policy gradient theorem to the multi-action setup.

EOF.