Kernel-Based Function Approximation for Average Reward Reinforcement Learning: An Optimist No-Regret Algorithm

Sattar Vakili (MediaTek Research)

Julia Olkhovskaya (TU Delft)

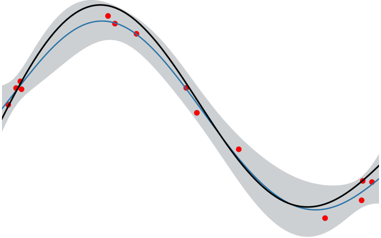
Oxford University, OxCSML Seminar
February 2025

Overview

Kernel Based Regression

Provided a dataset of t observation:

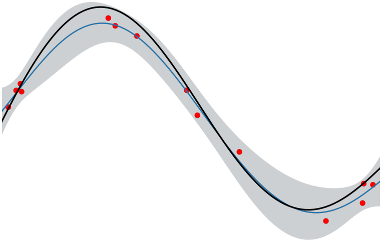
$$\left\{(z_j, Y(z_j))\right\}_{j=1}^{t}, Y(z_j) = f(z_j) + \varepsilon_j$$



Kernel-Based Regression

Predictor:

$$\hat{f}(z) = \boldsymbol{\kappa}_t^\top(z)(\mathbf{K}_t + \rho I)^{-1}\mathbf{y}_t$$

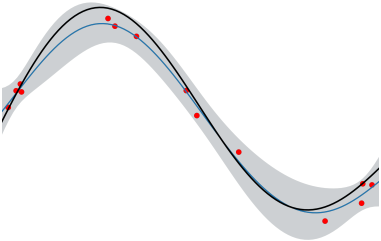


- $\boldsymbol{\kappa}_t(z) = [\kappa(z_1, z), \kappa(z_2, z), \cdots, \kappa(z_t, z)]$
- $\mathbf{K}_t = [\kappa(z_i, z_j)]_{i,j=1}^t$
- $\mathbf{y}_t = [Y(z_1), Y(z_2), \cdots, Y(z_t)]$

Kernel-Based Regression

Uncertainty estimator:

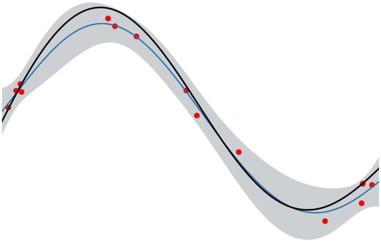
$$(\sigma_t(z))^2 = \kappa(z,z) - \boldsymbol{\kappa}_t^\top(z)(\mathbf{K}_t + \rho I)^{-1}\boldsymbol{\kappa}_t(z)$$



Kernel-Based Regression

Uncertainty estimator:

$$(\sigma_t(z))^2 = \kappa(z, z) - \boldsymbol{\kappa}_t^\top(z)(\mathbf{K}_t + \rho I)^{-1}\boldsymbol{\kappa}_t(z)$$

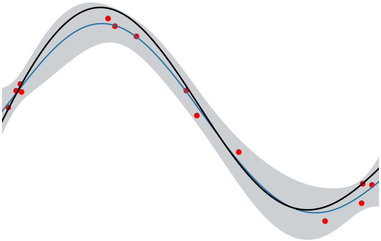


Closed from expressions for prediction and uncertainty quantification!

Kernel-Based Regression

Uncertainty estimator:

$$(\sigma_t(z))^2 = \kappa(z, z) - \boldsymbol{\kappa}_t^\top(z)(\mathbf{K}_t + \rho I)^{-1}\boldsymbol{\kappa}_t(z)$$



Closed from expressions for prediction and uncertainty quantification!

Confidence interval $|f(z) - \hat{f}_t(z)| \leq \beta(\delta)\sigma_t(z)$, w.p. $1 - \delta$

Bayesian and Frequentist Interpretations

Bayesian: Posterior mean (maximum likelihood estimation) assuming a prior centered Gaussian process distribution $\mathcal{GP}(\mathbf{0}, \kappa)$ and Gaussian noise

Bayesian and Frequentist Interpretations

Bayesian: Posterior mean (maximum likelihood estimation) assuming a prior centered Gaussian process distribution $\mathcal{GP}(\mathbf{0}, \kappa)$ and Gaussian noise

Frequntist: Regularized Least Squares Error Estimation

$$\hat{f} = \arg\min_{g \in \mathcal{H}_\kappa} \sum_{j=1}^{t} \left(Y(z_j) - g(z_j)\right)^2 + \lambda \|g\|^2_{\mathcal{H}_\kappa}$$

Reproducing Kernel Hilbert Space

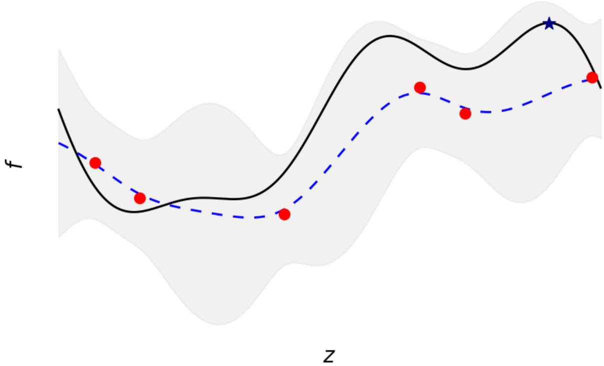
RKHS:

$$\mathcal{H}_\kappa = \{f(\cdot) = \sum_{m=1}^{\infty} w_m \phi_m(\cdot)\}$$

- Inner product $\langle f, g \rangle_{\mathcal{H}_\kappa} = \boldsymbol{w}_f^\top \boldsymbol{w}_g$
- $\|f\|_{\mathcal{H}_\kappa} = \|\mathbf{w}\|$
- $\phi_m = \sqrt{\lambda_m}\varphi_m$ form an orthonormal basis

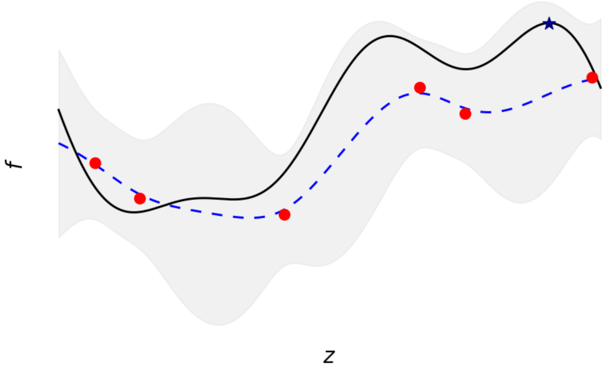
Overview

Bandits (Bayesian Optimization)



Procedure: Sequentially select points $z_1, z_2, \cdots, z_T$

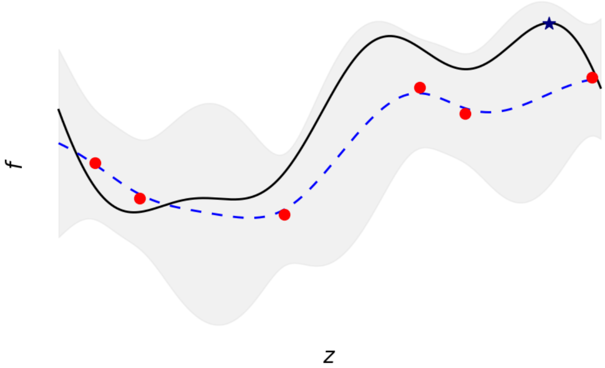
Bandits (Bayesian Optimization)



Procedure: Sequentially select points $z_1, z_2, \cdots, z_T$

Performance measure: $\text{Regret}(T) = \sum_{t=1}^{T} \left(f(z^\star) - f(z_t)\right)$.

Bandits (Bayesian Optimization)



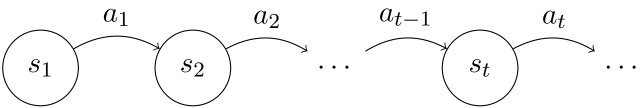
Procedure: Sequentially select points $z_1, z_2, \cdots, z_T$

Performance measure: $\text{Regret}(T) = \sum_{t=1}^{T} \left(f(z^\star) - f(z_t)\right)$.

Optimistic Algorithm: $z_t = \arg\max_z \left(\hat{f}_{t-1}(z) + \beta(\delta)\sigma_{t-1}(z)\right)$

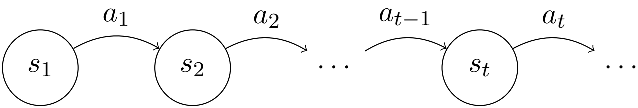
Overview

Markov Decision Process



Procedure: Observe $s_t \sim P(\cdot|s_{t-1}, a_{t-1})$, select $a_t = \pi(s_t)$

Markov Decision Process



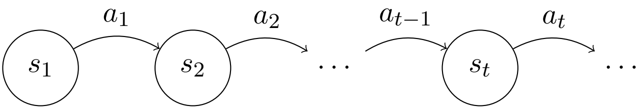
Procedure: Observe $s_t \sim P(\cdot|s_{t-1}, a_{t-1})$, select $a_t = \pi(s_t)$

Performance measure: $\text{Regret}(T) = \sum_{t=1}^{T} (J^\star - r(s_t, a_t))$.

$$J^\star = \max_\pi \liminf_{T\to\infty} \frac{1}{T}[\sum_{t=1}^{T} r(s_t, a_t)]$$

$$a_t = \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)$$

Markov Decision Process



Procedure: Observe $s_t \sim P(\cdot|s_{t-1}, a_{t-1})$, select $a_t = \pi(s_t)$

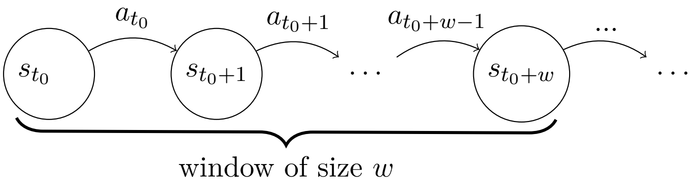
Performance measure: $\text{Regret}(T) = \sum_{t=1}^{T} (J^\star - r(s_t, a_t))$.

$$J^\star = \max_\pi \liminf_{T\to\infty} \frac{1}{T}[\sum_{t=1}^{T} r(s_t, a_t)]$$

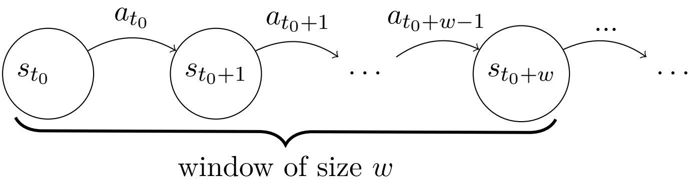
$$a_t = \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)$$

Weekly Communicating MDP

Value Function



Value Function

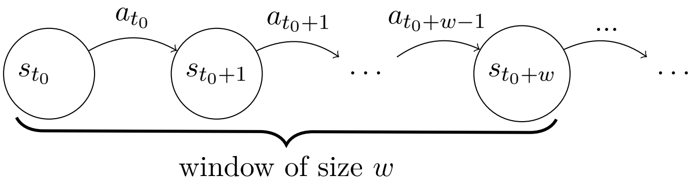


- The value of a policy π over a window w

$$v_w^\pi(s) = \mathbb{E}\left[\sum_{t=t_0}^{t_0+w-1} r(s_t, a_t)\right]$$

$$s = s_{t_0},\ a_t = \pi(s_t),\ s_{t+1} \sim P(\cdot|s_t, a_t)$$

Value Function



- The value of a policy π over a window w

$$v_w^\pi(s) = \mathbb{E}\left[\sum_{t=t_0}^{t_0+w-1} r(s_t, a_t)\right]$$

$$s = s_{t_0},\ a_t = \pi(s_t),\ s_{t+1} \sim P(\cdot|s_t, a_t)$$

$$q_w^\pi(s, a) = \mathbb{E}\left[\sum_{t=t_0}^{t_0+w-1} r(s_t, a_t)\right]$$

$$s = s_{t_0}, a = a_{t_0},\ a_{t+1} = \pi(s_{t+1}),\ s_{t+1} \sim P(\cdot|s_t, a_t)$$

Overview

Known Model

Dynamic Programming

Known Model

Dynamic Programming

- $v_{t_0+w} \leftarrow \mathbf{0}$

Known Model

Dynamic Programming

$$[Pv](s,a) := \mathbb{E}_{s'\sim P(\cdot|s,a)}[v(s')] = \int_{s'\in\mathcal{S}} v(s')P(s'|s,a)ds'$$

- $v_{t_0+w} \leftarrow \mathbf{0}$
- Recursively $t = t_0 + w - 1, t_0 + w - 2, \cdots, t_0$:

$$q_t(s,a) = r(s,a) + [Pv_{t+1}](s,a)$$
$$v_t(s) = \max_a q_t(s,a)$$

Known Model

Dynamic Programming

$$[Pv](s,a) := \mathbb{E}_{s' \sim P(\cdot|s,a)}[v(s')] = \int_{s' \in \mathcal{S}} v(s') P(s'|s,a) ds'$$

- $v_{t_0+w} \leftarrow \mathbf{0}$
- Recursively $t = t_0 + w - 1, t_0 + w - 2, \cdots, t_0$:

$$q_t(s,a) = r(s,a) + [Pv_{t+1}](s,a)$$
$$v_t(s) = \max_a q_t(s,a)$$

- $\pi_t^{\star,w}(s) = \arg\max_a q_t(s,a)$

Unknown Model : RL

r and P are unknown in RL

$$f_t = [Pv_{t+1}]$$
$$\text{UCB} = \hat{f}_t + \beta(\delta)\sigma_t$$

- $v_{t_0+w} \leftarrow \mathbf{0}$
- Recursively $t = t_0 + w - 1, t_0 + w - 2, \cdots, t_0$:

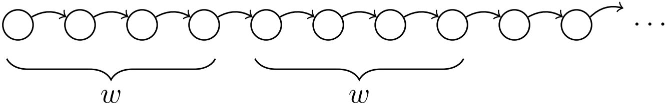
$$q_t(s,a) = r(s,a) + [Pv_{t+1}](s,a)$$

$$q_t(s,a) = r(s,a) + \text{UCB}([Pv_{t+1}])$$

$$v_t(s) = \max_a q_t(s,a)$$

- $\pi_t(s) = \arg\max_a q_t(s,a)$

KUCB-RL Algorithm



- Fix a window size w
- For each window $t \in [t_0, t_0 + w - 1]$, compute q_t and v_t, recursively, starting from $v_{t_0+w} = \mathbf{0}$
- Unroll the policy $a_t = \arg\max_a q_t(s_t, a)$ over this window

This is equivalent to solving a w-step look ahead MDP

Overview

Confidence Intervals

- How to create confidence intervals for $f = [Pv]$

$$\left| \hat{f}_t(s,a) - [Pv](s,a) \right| \leq \beta(\delta)\sigma_t(s,a).$$

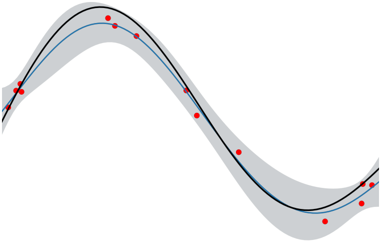
Confidence Intervals

- How to create confidence intervals for $f = [Pv]$

$$\left|\hat{f}_t(s,a) - [Pv](s,a)\right| \leq \beta(\delta)\sigma_t(s,a).$$

- For a fixed $f \in \mathcal{H}_\kappa$ with non-adaptive inputs $z_1, \ldots, z_t$,

$$\beta(\delta) \approx \|f\|_{\mathcal{H}_\kappa} + \frac{w}{\sqrt{\rho}}\sqrt{d\log(\tfrac{t}{\delta})}$$



Confidence Intervals

Challenge 1: Inputs $(s_1, a_1), \ldots, (s_t, a_t)$ are adaptive!

Solution: Self-normalized concentration inequalities for vector-valued martingales extended to kernel setting (Abbasi-Yadkori 2013; Whitehouse et al. 2023):

$$\beta(\delta) \approx \|f\|_{\mathcal{H}_\kappa} + \frac{w}{\sqrt{\rho}}\sqrt{\gamma(t;\rho) + \log(\tfrac{1}{\delta})}$$

- Maximum information gain: $\gamma(t;\rho) = \sup_{\{z_i\}_{i=1}^t \subset \mathcal{Z}} \frac{1}{2} \log\det\left(I + \frac{K_t}{\rho}\right)$

Confidence Intervals

Challenge 2: $f = [Pv]$ is not fixed, but v depends on past data!

$v \in \mathcal{V}$: the set of all possible v_t appearing in the algorithm!

Confidence Intervals

Challenge 2: $f = [Pv]$ is not fixed, but v depends on past data!

$v \in \mathcal{V}$: the set of all possible v_t appearing in the algorithm!

Solution: Optimistic closure assumption

- For all $v \in \mathcal{V}$ and some $\kappa' : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, $\|v\|_{\mathcal{H}_{\kappa'}} \leq C_v = \mathcal{O}(w)$

$$\beta(\delta) \approx \|f\|_{\mathcal{H}_\kappa} + \frac{C_v}{\sqrt{\rho}}\sqrt{\gamma(t;\rho) + \log(\tfrac{1}{\delta})}$$

Confidence Intervals

Challenge 2: $f = [Pv]$ is not fixed, but v depends on past data!

$v \in \mathcal{V}$: the set of all possible v_t appearing in the algorithm!

Solution: Optimistic closure assumption

- For all $v \in \mathcal{V}$ and some $\kappa' : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, $\|v\|_{\mathcal{H}_{\kappa'}} \leq C_v = \mathcal{O}(w)$

$$\beta(\delta) \approx \|f\|_{\mathcal{H}_\kappa} + \frac{C_v}{\sqrt{\rho}}\sqrt{\gamma(t;\rho) + \log(\tfrac{1}{\delta})}$$

- Proof idea:
  - Write v in terms of the feature space of κ'
  - For M largest eigenvalues use standard confidence intervals (Whitehouse et al. 2023)
  - The rest is bounded using the eigendecay

Overview

Performance Guarantee

Assumption: $P(s|\cdot,\cdot)$ is in the RKHS $\mathcal{H}_k$ of a known kernel k

Performance Guarantee

Assumption: $P(s|\cdot,\cdot)$ is in the RKHS $\mathcal{H}_k$ of a known kernel k

$\implies \|[Pv]\|_{\mathcal{H}_k} \leq C_f = \mathcal{O}(w)$ for integrable $v : \mathcal{S} \rightarrow \mathbb{R}$

Performance Guarantee

Assumption: $P(s|\cdot,\cdot)$ is in the RKHS $\mathcal{H}_k$ of a known kernel k

$\implies \|[Pv]\|_{\mathcal{H}_k} \leq C_f = \mathcal{O}(w)$ for integrable $v : \mathcal{S} \to \mathbb{R}$

Assumption: For all $v \in \mathcal{V}$ and some $\kappa' : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$,
$\|v\|_{\mathcal{H}_{\kappa'}} \leq C_v = \mathcal{O}(w)$

Performance Guarantee

Assumption: $P(s|\cdot,\cdot)$ is in the RKHS $\mathcal{H}_k$ of a known kernel k

$\implies \|[Pv]\|_{\mathcal{H}_k} \leq C_f = \mathcal{O}(w)$ for integrable $v : \mathcal{S} \to \mathbb{R}$

Assumption: For all $v \in \mathcal{V}$ and some $\kappa' : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$,
$\|v\|_{\mathcal{H}_{\kappa'}} \leq C_v = \mathcal{O}(w)$

Theorem: Regret bound w.p. $1 - \delta$

$$\mathcal{R}(T) = \mathcal{O}\left(\underbrace{\frac{T}{w}}_{w\text{-step look ahead}} + \underbrace{\beta(\delta) \sum_{t=1}^{T} \sigma_{w\lfloor \frac{t-1}{w} \rfloor}(s_t, a_t)}_{\text{Uncertainties in values}} \right)$$

Performance Guarantee

Assumption: $P(s|\cdot,\cdot)$ is in the RKHS $\mathcal{H}_k$ of a known kernel k

$\implies \|[Pv]\|_{\mathcal{H}_k} \le C_f = \mathcal{O}(w)$ for integrable $v : \mathcal{S} \to \mathbb{R}$

Assumption: For all $v \in \mathcal{V}$ and some $\kappa' : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, $\|v\|_{\mathcal{H}_{\kappa'}} \le C_v = \mathcal{O}(w)$

Theorem: Regret bound w.p. $1-\delta$

$$\mathcal{R}(T) = \mathcal{O}\left(\underbrace{\frac{T}{w}}_{w\text{-step look ahead}} + \underbrace{\beta(\delta)\sum_{t=1}^{T} \sigma_{w\lfloor \frac{t-1}{w} \rfloor}(s_t, a_t)}_{\text{Uncertainties in values}} \right)$$

$$\mathcal{R}(T) = \mathcal{O}\left(\frac{T}{w} + \left(w + \frac{w}{\sqrt{\rho}}\sqrt{\gamma(T;\rho) + \log\left(\frac{T}{\delta}\right)} \right) \sqrt{\rho T \gamma(T;\rho) + \rho^2 w^2 \gamma(T;\rho)\gamma(T/w;\rho)} \right)$$

Result

Matérn ν kernel (Neural tangent kernel)

$$\mathcal{R}(T) = \tilde{\mathcal{O}}(T^{\frac{3\nu+4d}{4\nu+4d}})$$

Result

Matérn ν kernel (Neural tangent kernel)

$$\mathcal{R}(T) = \tilde{\mathcal{O}}(T^{\frac{3\nu+4d}{4\nu+4d}})$$

- $\mathcal{R}(T) = o(T)$, the algorithm converges to the optimal stationary policy $\implies$ ***no-regret!***

Result

Matérn ν kernel (Neural tangent kernel)

$$\mathcal{R}(T) = \tilde{\mathcal{O}}(T^{\frac{3\nu+4d}{4\nu+4d}})$$

- $\mathcal{R}(T) = o(T)$, the algorithm converges to the optimal stationary policy $\implies$ ***no-regret!***

- For comparison, the regret bound for UCB algorithm in BO: $\mathcal{R}(T) = \tilde{\mathcal{O}}(T^{\frac{\nu+2d}{2\nu+2d}})$ (Whitehouse et al. 2023)

Discussion and Technical Challenges

- The policy is updated every w step: delay in using samples

$$\sum_{t=1}^{T} \sigma_{w\lfloor \frac{t-1}{w} \rfloor}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T; \rho) + \rho^2 w^2 \gamma(T; \rho) \gamma(T/w; \rho)}$$

Elliptical potential lemma (Srinivas et al. 2010).

$$\sum_{t=1}^{T} \sigma_{t-1}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T; \rho)}$$

Discussion and Technical Challenges

- The policy is updated every w step: delay in using samples

$$\sum_{t=1}^{T} \sigma_{w\lfloor \frac{t-1}{w} \rfloor}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T;\rho) + \rho^2 w^2 \gamma(T;\rho)\gamma(T/w;\rho)}$$

Elliptical potential lemma (Srinivas et al. 2010).

$$\sum_{t=1}^{T} \sigma_{t-1}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T;\rho)}$$

- The algorithm follows a w-step look ahead policy: this seems to be the main factor contributing to the difference with BO $\mathcal{O}(\frac{T}{w} + ...)$

Discussion and Technical Challenges

- The policy is updated every w step: delay in using samples

$$\sum_{t=1}^{T} \sigma_{w\lfloor \frac{t-1}{w} \rfloor}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T;\rho) + \rho^2 w^2 \gamma(T;\rho)\gamma(T/w;\rho)}$$

Elliptical potential lemma (Srinivas et al. 2010).

$$\sum_{t=1}^{T} \sigma_{t-1}(s_t, a_t)) \leq \sqrt{\rho T \gamma(T;\rho)}$$

- The algorithm follows a w-step look ahead policy: this seems to be the main factor contributing to the difference with BO $\mathcal{O}(\frac{T}{w} + ...)$

- Confidence interval is not applied to prefixed functions (as in BO)

Summary

The first no-regret performance guarantee for an algorithm in the infinite-horizon average-reward RL setting with kernel-based modeling

Summary

The first no-regret performance guarantee for an algorithm in the infinite-horizon average-reward RL setting with kernel-based modeling

Open problem: Is our bound improvable? What is the minimum rate of regret growth with T?

References I

Y. Abbasi-Yadkori. Online learning for linearly parametrized control problems. *PhD Thesis, University of Alberta*, 2013.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 1015–1022, July 2010.

J. Whitehouse, A. Ramdas, and S. Z. Wu. On the sublinear regret of gp-ucb. *Advances in Neural Information Processing Systems*, 36, 2023.