Online Reinforcement Learning in Large and Structured Environments

Sayak Ray Chowdhury

CISE Postdoctoral Fellow
Boston University

sayak@bu.edu

January 28, 2022

Reinforcement learning

Reinforcement learning is concerned with learning to take actions to maximize rewards, by trial and error, in environments that can evolve in response to actions

- Traditional Search Goal: Find a policy with high total reward using as few interactions with the environment as possible

Reinforcement learning

Reinforcement learning is concerned with learning to take actions to maximize rewards, by trial and error, in environments that can evolve in response to actions

- Traditional Search Goal: Find a policy with high total reward using as few interactions with the environment as possible
- Optimization Goal: Maximize total reward / Minimize regret (shortfall) in total reward compared to an optimal policy

Reinforcement learning

Reinforcement learning is concerned with learning to take actions to maximize rewards, by trial and error, in environments that can evolve in response to actions

- Traditional Search Goal: Find a policy with high total reward using as few interactions with the environment as possible
- Optimization Goal: Maximize total reward / Minimize regret (shortfall) in total reward compared to an optimal policy
  - **Applications:** Recommendation systems, Sequential investment, Dynamic resource allocation ....
  - No separate budget to purely exploring the environment
  - Exploration and Exploitation must be carefully balanced

Outline

Reinforcement learning algorithms for Regret Minimization in large and structured (unknown) environments

Outline

Reinforcement learning algorithms for Regret Minimization in large and structured (unknown) environments

- **Part 1:** Online learning in large scale **Multi-armed Bandits**
- **Part 2:** Online learning in large **Markov Decision Processes**

Outline

Reinforcement learning algorithms for Regret Minimization in large and structured (unknown) environments

- **Part 1:** Online learning in large scale **Multi-armed Bandits**
- **Part 2:** Online learning in large **Markov Decision Processes**

Main Challenge: Generalizing learned knowledge across unseen states and actions

Part 1: Online Learning in large-scale Multi-armed Bandits[1]

[1] S. R. Chowdhury and A. Gopalan, "*On kernelized multi-armed bandits*", **ICML**, 2017.

Background: Multi-armed Bandit

1 **2** **3** ••• **N**

N arms with **unknown** parameters $\mu_1, \ldots, \mu_N$

(Think Bernoulli)

Background: Multi-armed Bandit

Time 1

1 $\boxed{\mathbf{2}}$ **3** ••• **N**

$y_1 \sim \mu_2$

Background: Multi-armed Bandit

Time 2

$\boxed{\mathbf{1}}$ $\mathbf{2}$ $\mathbf{3}$ ••• $\mathbf{N}$

$y_2 \sim \mu_1$

Background: Multi-armed Bandit

Time 3

$\mathbf{1} \quad \mathbf{2} \quad \boxed{\mathbf{3}} \quad \cdots \quad \mathbf{N}$

$$y_3 \sim \mu_3$$

Background: Multi-armed Bandit

Time 4

$$\mathbf{1} \qquad \boxed{\mathbf{2}} \qquad \mathbf{3} \qquad \bullet\bullet\bullet \qquad \mathbf{N}$$

$$y_4 \sim \mu_2$$

Background: Multi-armed Bandit

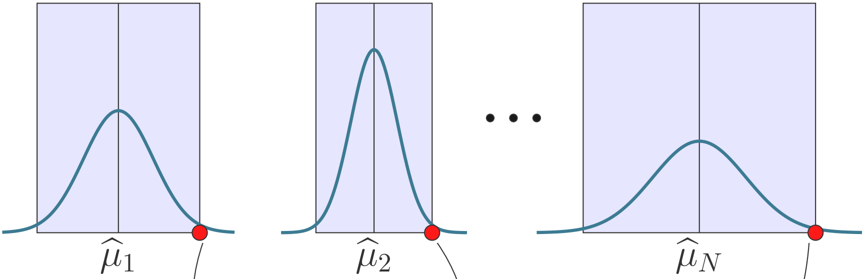
Time T

$\mathbf{1}$ $\mathbf{2}$ $\mathbf{3}$ ••• $\boxed{\mathbf{N}}$

$y_T \sim \mu_N$

Upper Confidence Bound algorithm [Auer et al.'02]

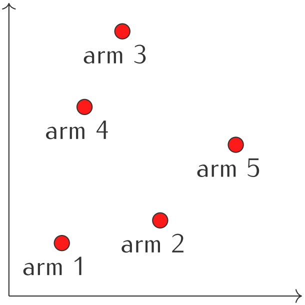
Idea: Be optimistic under uncertainty !



Play arm $i_t = \underset{1 \leqslant i \leqslant N}{\operatorname{argmax}} \left\{ \widehat{\mu}_i + \sqrt{\frac{2 \log t}{k_i}} \right\}$

Variation: Linear Bandits [Dani et al.'08, ...]

- Assumption in MAB: Arms' rewards are independent
- Often, more structure / coupling is present



Problem setting

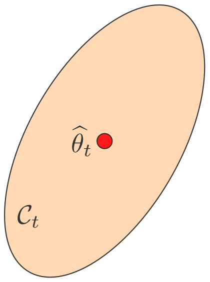
- Each arm i is a vector $x_i \in \mathbb{R}^d$
- Playing arm i_t gives reward

$$y_t = \theta^\top x_{i_t} + \varepsilon_t$$

- $\theta \in \mathbb{R}^d$ is unknown and $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ is noise

Variation: Linear Bandits [Dani et al.'08, ...]

- Assumption in MAB: Arms' rewards are independent
- Often, more structure / coupling is present

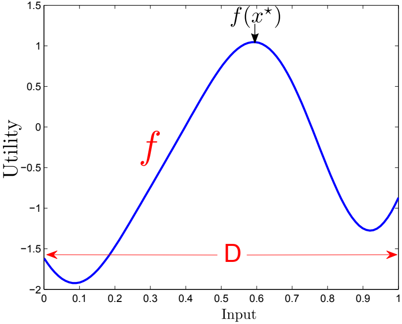


LinUCB algorithm

- Build a point estimate (least squares estimate) $\widehat{\theta}_t$ and a confidence region (ellipsoid) $\mathcal{C}_t$
- Play the most optimistic action w.r.t. this ellipsoid

$$i_t = \underset{1 \leqslant i \leqslant N}{\operatorname{argmax}} \max_{\theta \in \mathcal{C}_t} \left(\theta^\top x_i \right)$$

Generalization: Black-box optimization



Problem setting:

- Maximize an unknown utility function $f : D \to \mathbb{R}$
- Sequentially query f with points $x_1, x_2, \ldots, x_T$
- Noisy point evaluations: $y_t = f(x_t) + \varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ is the noise

Goal: Maximize cumulative (expected) utility: $\sum_{t=1}^{T} \mathbb{E}[y_t]$

or, equivalently,

Minimize cumulative regret: $\sum_{t=1}^{T} \left(f(x^\star) - f(x_t) \right)$

Application: Hyperparameter tuning

Hyperparameters in DeepNN training

- Learning rate
- Regularizer
- Number of hidden layers
- Number of units in each layer
- Optimizer (SGD, Adagrad, Adam, ...)
- Nonlinearity (Relu, Softmax, ...)

DeepNN training as Black-box optimization

- D : all possible hyperparameter configurations
- $f(x)$: training error for configuration x
- $x^\star$: the best set of hyperparameters

Huge (possibly infinite) set of hyperparameters to choose from !!!

Possible approaches

Grid search, Random search

- Do not use information from previous searches
- Not good when point evaluations are expensive

Need to make an educated decision about where to search next

Possible approaches

Grid search, Random search

- Do not use information from previous searches
- Not good when point evaluations are expensive

Need to make an educated decision about where to search next

Bayesian optimization

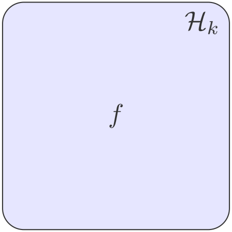
1. Learn a (probabilistic) model for f
2. Use model predictions and uncertainty to select query x_t
3. Update model with data (x_t, y_t) and repeat.

Regularity assumption

- f is an element of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ associated with a kernel $k : D \times D \to \mathbb{R}$

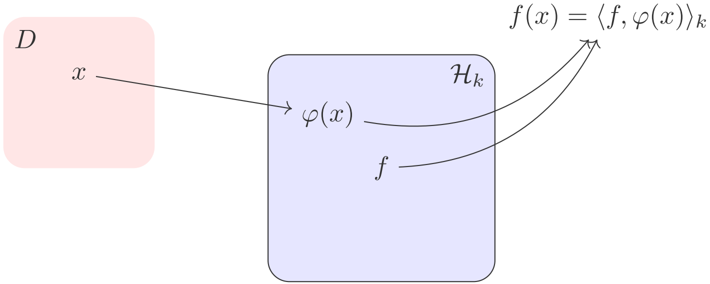
Regularity assumption

- f is an element of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ associated with a kernel $k : D \times D \to \mathbb{R}$



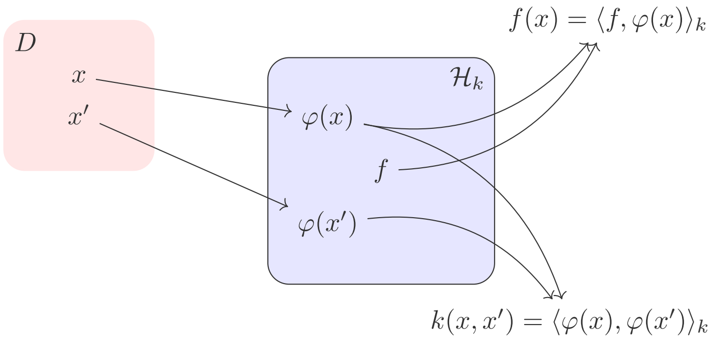
Regularity assumption

- f is an element of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ associated with a kernel $k : D \times D \to \mathbb{R}$



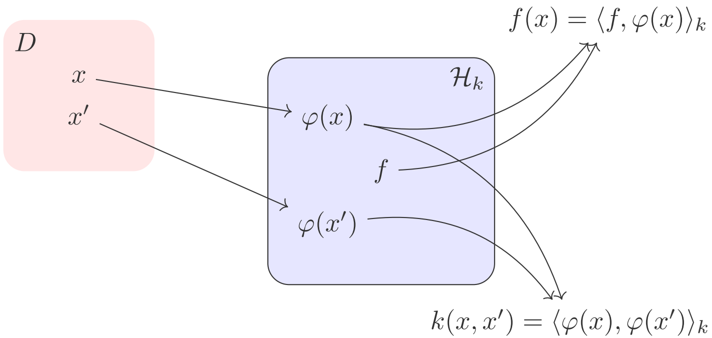
Regularity assumption

- f is an element of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ associated with a kernel $k : D \times D \to \mathbb{R}$



Regularity assumption

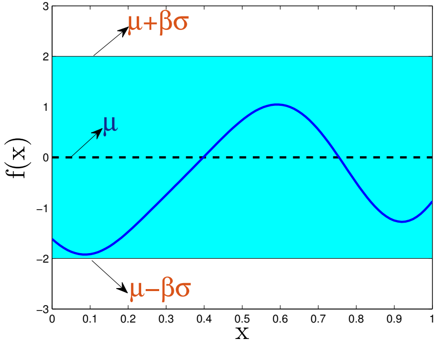
- f is an element of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ associated with a kernel $k : D \times D \to \mathbb{R}$



- Induces smoothness: $|f(x) - f(x')| \leqslant \|f\|_k \, \|\varphi(x) - \varphi(x')\|_k$

Algorithm Design

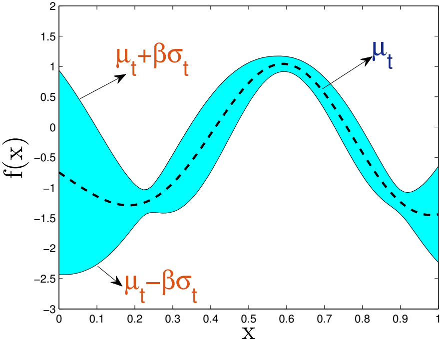
Idea: Gaussian Process (GP) regression



- Unknown utility function f modeled by Gaussian processes, $f \sim \mathcal{GP}\left(0, k(x, x')\right)$

Algorithm Design

Idea: Gaussian Process (GP) regression



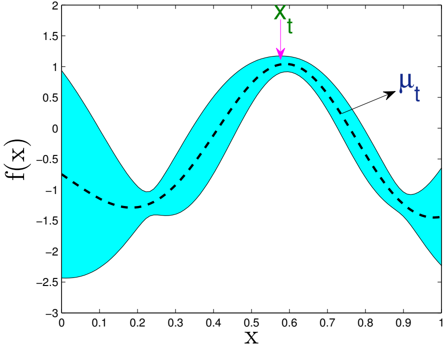
- Unknown utility function f modeled by Gaussian processes, $f \sim \mathcal{GP}\left(0, k(x, x')\right)$
- Posterior of f given t noisy observations $\mathcal{H}_t = (x_\tau, y_\tau)_{\tau=1}^t$ is a GP

$$f|\mathcal{H}_t \sim \mathcal{GP}(\mu_t(x), k_t(x, x'))$$

Posterior mean and covariance:

$$\begin{aligned} \mu_t(x) &= k_t(x)^\top (K_t + \sigma^2 I)^{-1} y_{1:t} \\ k_t(x, x') &= k(x, x') - k_t(x)^\top (K_t + \sigma^2 I)^{-1} k_t(x') \end{aligned}$$

Key Idea: Choose the point with highest Upper Confidence Bound

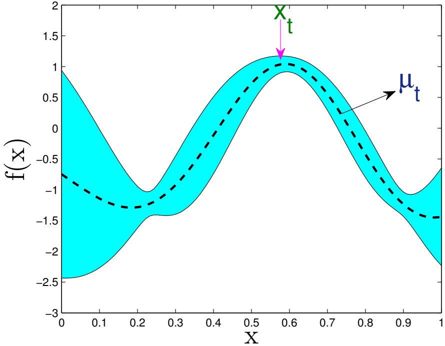


At each round t, choose the query point x_t using current GP posterior and a suitable parameter β_t:

$$x_t \in \operatorname*{argmax}_{x \in D} \mu_t(x) + \beta_t \sigma_t(x)$$

Algorithm 1: Improved GP-UCB

Key Idea: Choose the point with highest Upper Confidence Bound



At each round t, choose the query point x_t using current GP posterior and a suitable parameter β_t:

$$x_t \in \operatorname*{argmax}_{x \in D} \mu_t(x) + \beta_t \sigma_t(x)$$

First appeared as **GP-UCB** [Srinivas et al.'10] $\longrightarrow$ We provide a tighter regret bound ($O(\log T)$ improvement!)

IGP-UCB achieves sublinear regret [CG'17]

Regret bound: $O\left(\gamma_T\sqrt{T\log\left(\frac{1}{\delta}\right)}\right)$ with probability at least $1-\delta$

IGP-UCB achieves sublinear regret [CG'17]

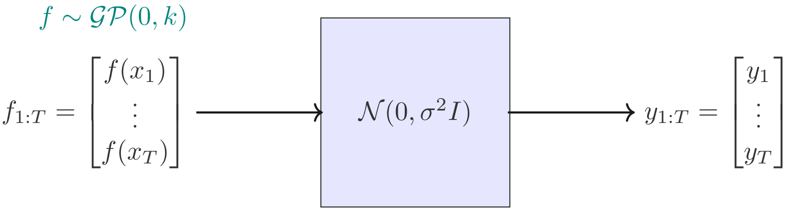
Regret bound: $O\left(\gamma_T \sqrt{T \log\left(\frac{1}{\delta}\right)}\right)$ with probability at least $1-\delta$

"Information Complexity": Captures reduction in uncertainty after observing noisy rewards (depends on the kernel function)

IGP-UCB achieves sublinear regret [CG'17]

Regret bound: $O\left(\gamma_T \sqrt{T \log\left(\frac{1}{\delta}\right)}\right)$ with probability at least $1-\delta$

"Information Complexity": Captures reduction in uncertainty after observing noisy rewards (depends on the kernel function)



$$\gamma_T = \max_{\{x_1, \ldots x_T\}} \mathtt{I}(y_{1:T}, f_{1:T})$$

Algorithm 2: Gaussian Process Thompson Sampling

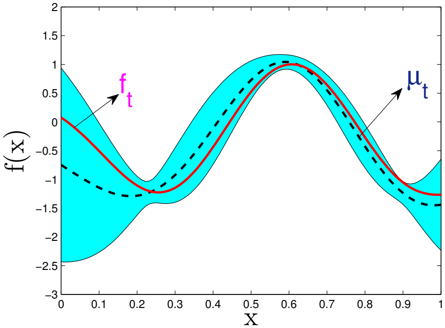
What would a "Bayesian" do?

- Sample a random function and choose its maximizer
- Prehistoric [Thompson'33]

Algorithm 2: Gaussian Process Thompson Sampling

What would a "Bayesian" do?

- Sample a random function and choose its maximizer
- Prehistoric [Thompson'33]



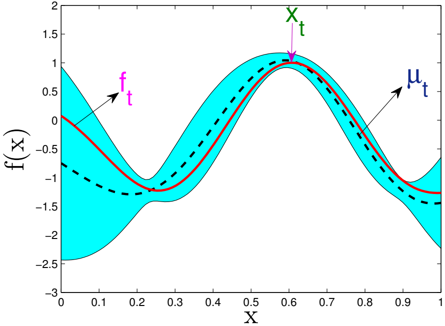
At each round t:

- Sample a function f_t from current GP posterior

Algorithm 2: Gaussian Process Thompson Sampling

What would a "Bayesian" do?

- Sample a random function and choose its maximizer
- Prehistoric [Thompson'33]



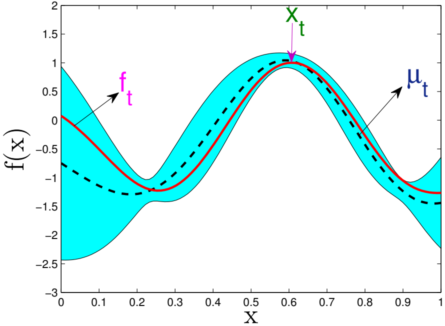
At each round t:

- Sample a function f_t from current GP posterior
- Choose the query point $x_t = \operatorname*{argmax}_{x \in D} f_t(x)$

Algorithm 2: Gaussian Process Thompson Sampling

What would a "Bayesian" do?

- Sample a random function and choose its maximizer
- Prehistoric [Thompson'33]



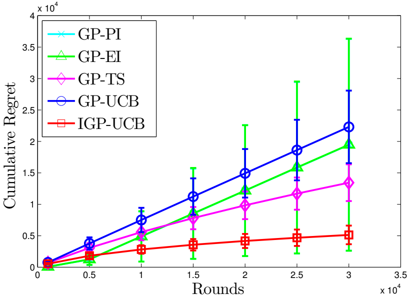
At each round t:

- Sample a function f_t from current GP posterior
- Choose the query point $x_t = \underset{x \in D}{\operatorname{argmax}} f_t(x)$

Regret bound: $O\left(\gamma_T \sqrt{dT \log\left(\frac{1}{\delta}\right)}\right)$ with probability at least $1-\delta$

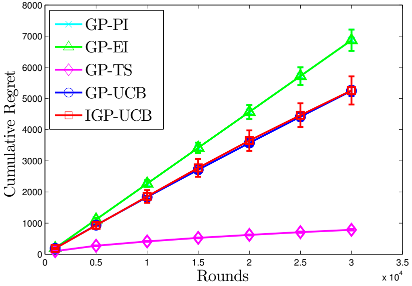
Numerics

f sampled from RKHS (RBF kernel)



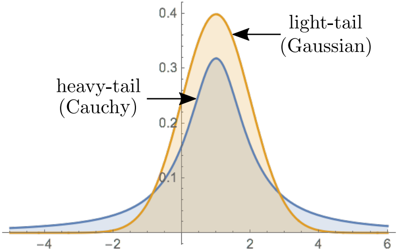
- IGP-UCB improves over GP-UCB ☺☺
- GP-TS fares well ☺

Temperature Sensor Data (Intel Berkeley Research lab)



- IGP-UCB performs similar to GP-UCB ✓
- GP-TS performs the best ☺

Variation: Heavy-tailed kernelized bandits[2]



eg. Student's-t, Pareto, Cauchy etc.

Motivation:

- Distribution of delays in communication networks
- Bursty traffic flow distributions
- Price fluctuations in financial and insurance data

- Heavy-tailed payoffs: $\mathbb{E}\left[|y_t|^2\right] < +\infty$
- **Results:** Regret upper and lower bounds of order $\approx \gamma_T \sqrt{T}$

Bounded second-moment sufficient for $O(\sqrt{T})$ regret!

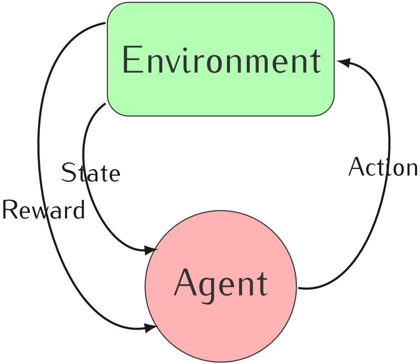
[2] S. R. Chowdhury and A. Gopalan, "*Bayesian Optimization under Heavy-tailed Payoffs*", **NeurIPS**, 2019.

Part 2: Online Learning in Large-scale Markov Decision Processes[3]

[3] S. R. Chowdhury, A. Gopalan and O.-A. Maillard, "*Reinforcement Learning in Parametric MDPs with Exponential Families*", **AISTATS**, 2021.

Markov Decision Process

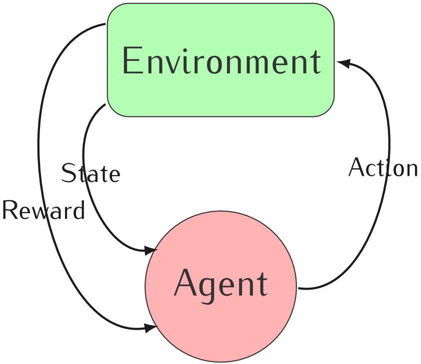
We consider learning in an episodic MDP $\{\mathcal{S}, \mathcal{A}, R, P, H\}$



- State space $\mathcal{S} \subset \mathbb{R}^m$
- Action space $\mathcal{A} \subset \mathbb{R}^n$
- Transition probabilities $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- Reward function $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$
- Finite episode length $H \in \mathbb{N}$

Markov Decision Process

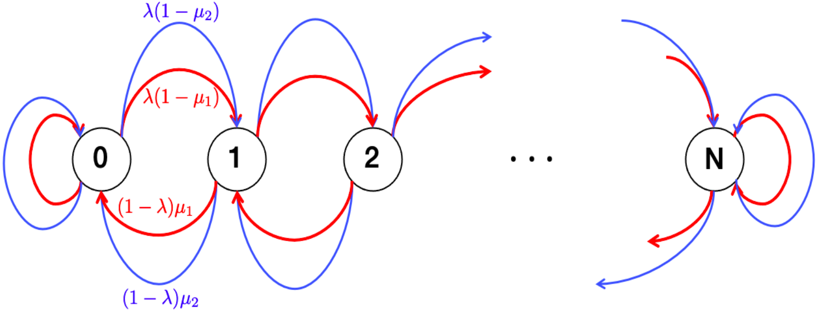
We consider learning in an episodic MDP $\{\mathcal{S}, \mathcal{A}, R, P, H\}$



- State space $\mathcal{S} \subset \mathbb{R}^m$
- Action space $\mathcal{A} \subset \mathbb{R}^n$
- Transition probabilities $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- Reward function $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$
- Finite episode length $H \in \mathbb{N}$

Parameterized MDP – Queuing System

Transitions are parameterized by θ – unknown to the agent a priori



- Single queue with N states, discrete-time
- `Bernoulli`(λ) arrivals at every state
- 2 actions: (`Bernoulli`) service rates $\{\mu_1, \mu_2\}$
- Assume service rates known, uncertainty in λ only

Online Reinforcement Learning

- MDP parameter θ is unknown to the decision maker a priori
- Must LEARN optimal policy – what action to take in each state to maximize the cumulative reward

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{h=1}^{H} R(s_{t,h}, a_{t,h})\right]$$

or, equivalently, minimize the cumulative regret

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{h=1}^{H} R(s_{t,h}, a^{\star}_{t,h})\right] - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{h=1}^{H} R(s_{t,h}, a_{t,h})\right] \; ,$$

- **Trade-off:** Explore the state space or Exploit existing knowledge to design good current policy?

Online reinforcement learning

- **Upper confidence-based approaches:** Build confidence intervals per state-action pair, be optimistic!
  - Rmax [Brafman-Tennenholtz'01]
  - UCRL2 [Jaksch et al.'07]
- **Key idea:** Maintain estimates + high-confidence sets for transition probabilities for every state-action pair
- "Wasteful" if transitions have **structure/relations**

Online reinforcement learning

- **Upper confidence-based approaches:** Build confidence intervals per state-action pair, be optimistic!
  - Rmax [Brafman-Tennenholtz'01]
  - UCRL2 [Jaksch et al.'07]
- **Key idea:** Maintain estimates + high-confidence sets for transition probabilities for every state-action pair
- "Wasteful" if transitions have **structure/relations**

Parameterized MDP: Transitions are described by a finite dimensional parameter – unknown to the agent a priori

Example: Linear quadratic regulator

<u>Berstekas'04:</u>

State transitions: $s' = As + Ba + \eta$

- $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times n}$ are unknown matrices
- Noise $\eta \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ with known variance

Rewards: $R(s,a) = s^\top Ps + a^\top Qa$

- $P \in \mathbb{R}^{m\times m}$ and $Q \in \mathbb{R}^{n\times n}$ are known matrices

Generic models: Linear MDPs

Yang and Wang'19: $P(s'|s,a) = \psi(s')^\top M \varphi(s,a)$

Jin et al.'19: $P(s'|s,a) = \theta^\top \nu(s', s, a)$

- Known feature functions: $\psi(s')$, $\varphi(s,a)$ and $\nu(s',s,a)$
- Unknown parameters: M (a matrix) and θ (a vector)
- **Special case:** Tabular finite-state, finite-action MDP

DO NOT cover several popular models:

- Linearly controlled systems [Bertsekas' 04]
- Factored MDPs [Kearns and Koller' 99]

Our Model: Linear Exponential Family [CGM'21]

log-transition probabilities are linear:

$$P_\theta(s'|s,a) = \exp\left(\theta^\top F(s',s,a) - Z_{s,a}(\theta)\right)$$

- $\theta \in \mathbb{R}^d$ is the unknown parameter of the model
- $F(s',s,a)$ are known features (sufficient statistic)
- $Z_{s,a}(\theta)$ is the log-partition function

(Captures a **wide range of distributions**, e.g., Gaussian, Bernoulli, Gamma, Chi-square, ...)

Algorithm: Exponential family UCRL (Exp-UCRL)

Key idea: Maintain estimates + high-confidence sets for θ

At each episode t:

1. Compute the penalized maximum-likelihood estimate (MLE) $\widehat{\theta}_t$ of θ from data $\{s_{\tau,h}, a_{\tau,h}, s_{\tau,h+1}\}_{\tau<t,h\leq H}$:

$$\widehat{\theta}_t \in \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \sum_{\tau<t,h\leq H} -\log P_\theta(s_{\tau,h+1}|s_{\tau,h}, a_{\tau,h}) + \frac{1}{2}\|\theta\|^2$$

2. Build a confidence set around the MLE:

$$\mathcal{C}_t = \left\{\theta \mid \sum_{\tau<t,h\leq H} \mathtt{KL}\Big(P_{\widehat{\theta}_t}(\cdot|s_{\tau,h}, a_{\tau,h}), P_\theta(\cdot|s_{\tau,h}, a_{\tau,h})\Big) + \frac{1}{2}\left\|\widehat{\theta}_t - \theta\right\|^2 \leq \beta_t\right\}$$

3. Compute the optimal policy w.r.t. $\mathcal{C}_t$ (by value iteration, simulation,...) and **play actions prescribed by that policy**

Exp-UCRL attains sublinear regret [CGM'21]

Concentration inequality: $\mathbb{P}\big[\forall t \in \mathbb{N},\ \theta \in \mathcal{C}_t\big] \geq 1 - \delta$

- A **general** result to design high-confidence sets for adaptive regression in conditional exponential families
- Generalize known results for linear bandits [Abbasi-Yadkori et al.'11] and GLM bandits [Filippi et al.'10]

Exp-UCRL attains sublinear regret [CGM'21]

Concentration inequality: $\mathbb{P}\left[\forall t \in \mathbb{N},\ \theta \in \mathcal{C}_t\right] \geq 1 - \delta$

- A **general** result to design high-confidence sets for adaptive regression in conditional exponential families
- Generalize known results for linear bandits [Abbasi-Yadkori et al.'11] and GLM bandits [Filippi et al.'10]

Regret Bound: $O\left(\frac{\beta}{\sqrt{\alpha}} H^2 d \sqrt{T \log(1/\delta)}\right)$ with probability $\geq 1-\delta$

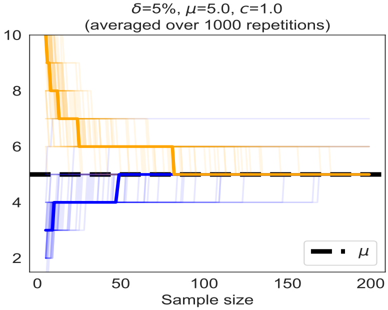
$$\alpha = \inf_{\theta,s,a} \lambda_{\min}\left(\mathcal{C}_\theta[\psi(s')|s,a]\right)$$

$$\beta = \sup_{\theta,s,a} \lambda_{\max}\left(\mathcal{C}_\theta[\psi(s')|s,a]\right)$$

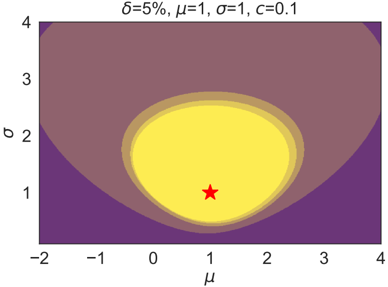
Model dependent constants — encode degree of **non-linearity**

Confidence Sets in Exponential Families [CSGM'22]

A general toolkit:[4] High probability confidence sets in exponential families — applications well beyond bandits and RL



Confidence sets for Chi-square distribution



Joint confidence sets for (μ, σ) for Gaussian distribution

[4] S. R. Chowdhury, P. Saux, A. Gopalan, O.-A. Maillard "*Bregman Deviations of Generic Exponential Families*", **arxiv**, 2022.

Variation: Privacy Concerns in RL

Reinforcement learning (RL) widely applied to personalized service:

- personalized healthcare
- virtual assistants
- social robots
- online recommendations
- ...

However, both states and rewards contain user's sensitive information

- healthcare: age, gender, treatment history
- virtual assistants: words, voice, sentences.
- social robots: facial expressions and scores on puzzles
- online recommendations: shopping habits

How to protect all these information in a rigorous way?

Differential privacy

Central Model:

- The learning agent has access to users' personal data
- Privacy protection: adversary cannot infer any particular user's data by observing the outputs the agent

Local Model:

- Each user protects her data at the local side
- The learning agent only has private data from users

Can we have a unified framework for both models in RL?

Privacy in Tabular MDPs

Private information:

- number of visits to a state-action pair $n(s,a)$ – bits $(0/1)$
- number of transitions to a given state $n(s'|s,a)$ – bits $(0/1)$
- rewards $r(s,a)$ – scalars $[0,1]$

The goal: release private counts with minimal amount of noise

Protection in local model:

- each user k, add independent noise, leads to sum of K noise

Protection in central model:

- Binary counting mechanism, only $\log K$ noise [Chan et al.'11]

Private Algorithms and Regret

Private Policy Optimization:[5]

- Central Model: $\widetilde{O}\left(\sqrt{S^2AH^3T} + S^2AH^3/\varepsilon\right)$
- Local Model: $\widetilde{O}\left(\sqrt{S^2AH^3T} + S^2A\sqrt{H^5T}/\varepsilon\right)$

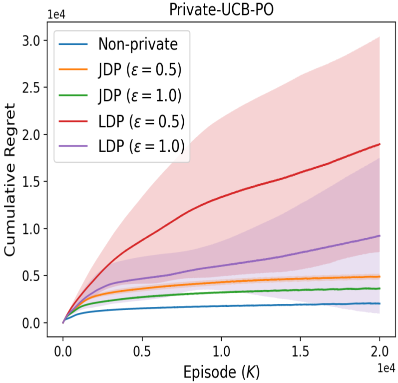
Private Value Iteration:

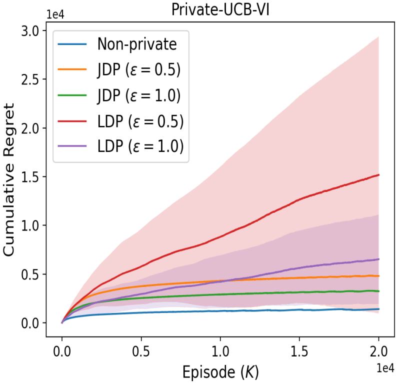
- Central Model: $\widetilde{O}\left(\sqrt{SAH^3T} + S^2AH^3/\varepsilon\right)$
- Local Model: $\widetilde{O}\left(\sqrt{SAH^3T} + S^2A\sqrt{H^5T}/\varepsilon\right)$

Regret increases if level of privacy increases (i.e., ε decreases)

[5] S. R. Chowdhury and X. Zhou, "*Differentially Private Regret Minimization in Episodic Markov Decision Processes*", **AAAI**, 2022.

Numerics [CZ'22]





Current/Future Work

Model selection in bandits and MDPs:

- Function class (in bandits) / class of transition distributions (in MDPs) unknown?
- How to identify the correct model class during learning?

Ethics/Privacy questions:

- How to protect users' sensitive information (e.g. recommendation systems, personalized treatment)?
- Fairness and transparency in policy selection?

Multi-agent systems:

- Decentralized peer-to-peer learning/ Federated learning?
- Impact of **network properties**, **delay in communication**, etc.?

Thank You