Sample Complexity of Classification-based Policy Iteration Algorithms

Mohammad Ghavamzadeh

Adobe Research & INRIA Lille

November 19, 2014

Outline

Overview

Classification-based Policy Iteration Algorithms

Resource Allocation

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

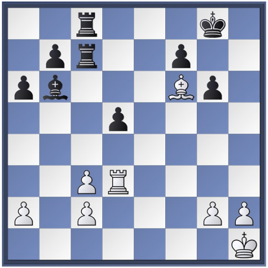
Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

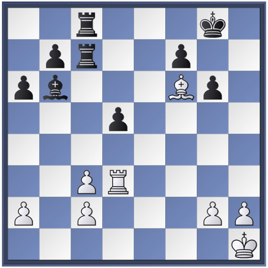
Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

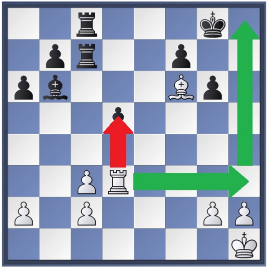
Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Computer Games

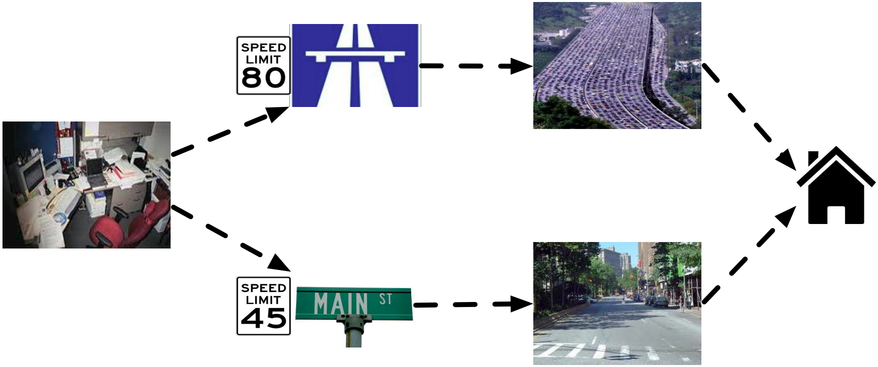


Computer Games

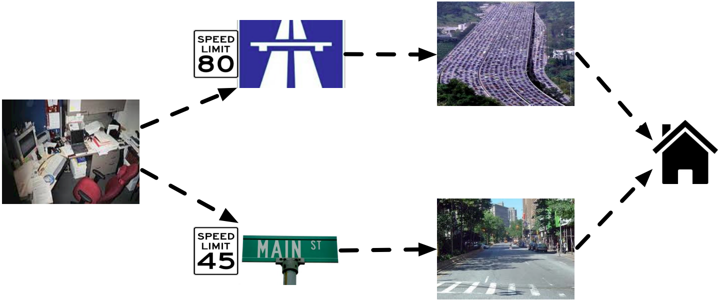




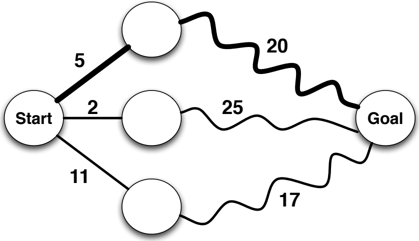
Routing & Traffic Control



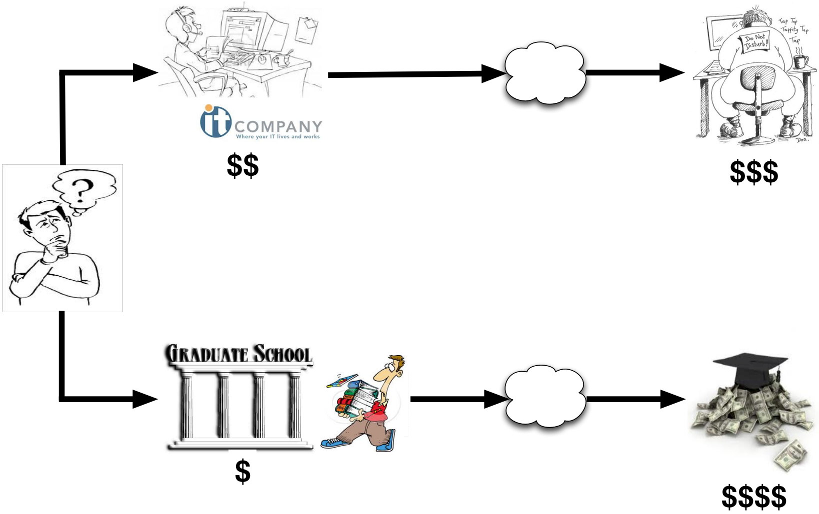
Routing & Traffic Control



shortest path problem



Career Decisions



Marketing & Finance

Which ad to show to this customer???

Which ad has the highest probability to be clicked by this customer???

one-shot decision

Sequential Decision-Making under Uncertainty

How Can I ... ?

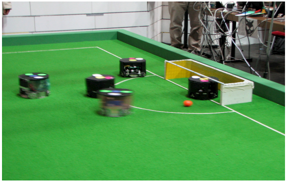
Play and Win a Game

Multi-Player Games

Computer Games

Two-Player Games

Move around in the Physical World (e.g. driving, navigation)



. . . and many more

Power Management

Factory Optimization

Information Retrieval

Medical Diagnosis & Treatment

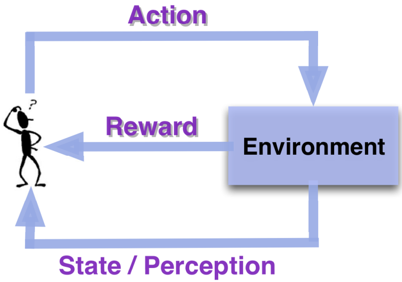
Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

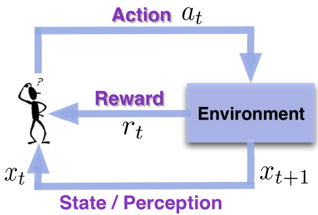
Reinforcement Learning (RL)



- **RL:** A class of learning problems in which an agent interacts with a dynamic, stochastic, and incompletely known environment

- **Goal:** Learn an action-selection strategy, or policy, to optimize some measure of its long-term performance

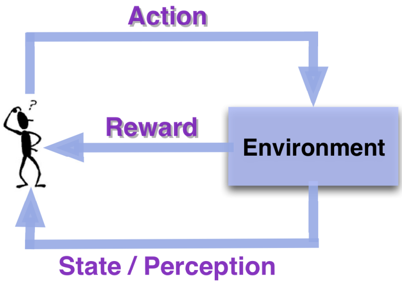
Reinforcement Learning (RL)



Agent's Life $\quad x_0\ a_0\ r_0\ x_1\ a_1\ r_1\ \dots\ \underbrace{x_t\ a_t\ r_t\ x_{t+1}}_{\text{unit of experience}}\ \dots$

- **Agent** has incomplete knowledge about its environment
- **Agent** chooses actions so as to optimize some measure of its long-term performance

Reinforcement Learning (RL)



- **RL:** A class of learning problems in which an agent interacts with a dynamic, stochastic, and incompletely known environment
- **Goal:** Learn an action-selection strategy, or policy, to optimize some measure of its long-term performance
- **Interaction:** Modeled as a MDP or a POMDP

Markov Decision Process

MDP

- An MDP $\mathcal{M}$ is a tuple $\langle \mathcal{X}, \mathcal{A}, r, p, \gamma \rangle$.
- The state space $\mathcal{X}$ is a bounded closed subset of $\mathbb{R}^d$.
- The set of actions $\mathcal{A}$ is finite ($|\mathcal{A}| < \infty$).
- The reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is bounded by $R_{\max}$.
- The transition model $p(\cdot|x, a)$ is a distribution over $\mathcal{X}$.
- $\gamma \in (0, 1)$ is a discount factor.

- **Policy:** a mapping from states to actions $\pi(x) \in \mathcal{A}$

Value Function

For a policy π

- **Value function** $V^\pi : \mathcal{X} \to \mathbb{R}$

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X_t, \pi(X_t)\big) \mid X_0 = x,\ \pi\right]$$

- **Action-value function** $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

$$Q^\pi(x,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t) \mid X_0 = x,\ A_0 = a,\ \pi\right]$$

Optimal Value Function and Optimal Policy

- **Optimal value function**

$$V^*(x) = \sup_\pi V^\pi(x) \qquad \forall x \in \mathcal{X}$$

- **Optimal action-value function**

$$Q^*(x,a) = \sup_\pi Q^\pi(x,a) \qquad \forall x \in \mathcal{X},\ \forall a \in \mathcal{A}$$

- A policy π is **optimal** if

$$V^\pi(x) = V^*(x) \qquad \forall x \in \mathcal{X}$$

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Dynamic Programming Algorithms

Policy Iteration

- start with an arbitrary policy π_0
- at each iteration k
  - Policy Evaluation: Compute Q^{π_k}
  - Policy Improvement: Compute the *greedy* policy w.r.t. Q^{π_k}

$$\pi_{k+1}(x) = (\mathcal{G}\pi_k)(x) = \arg\max_{a \in \mathcal{A}} Q^{\pi_k}(x, a) \qquad \forall x \in \mathcal{X}$$

* $\mathcal{G}$ is called the ***greedy policy operator***

Dynamic Programming Algorithms

Policy Iteration

- start with an arbitrary policy π_0
- at each iteration k
  - Policy Evaluation: Compute Q^{π_k}
  - Policy Improvement: Compute the *greedy* policy w.r.t. Q^{π_k}

$$\pi_{k+1}(x) = (\mathcal{G}\pi_k)(x) = \arg\max_{a \in \mathcal{A}} Q^{\pi_k}(x, a) \qquad \forall x \in \mathcal{X}$$

* $\mathcal{G}$ is called the ***greedy policy operator***

* the new policy resulted from the application of $\mathcal{G}$ is no worse than the old one

$$\pi_{k+1} = \mathcal{G}\pi_k \quad \longrightarrow \quad V^{\pi_{k+1}} \geq V^{\pi_k}$$

What will happen if we cannot compute Q^{π_k}?

Compute $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$ instead

What will happen if we cannot compute Q^{π_k}?

Compute $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$ instead

Why?

What will happen if we cannot compute Q^{π_k}?

Compute $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$ instead

Why?

- state space $\mathcal{X}$ and/or action space $\mathcal{A}$ are **large** or **infinite**
- not enough **time** to compute Q^{π_k}
- **model** of the system ***(transitions*** p ***and rewards*** r***)*** is unknown
- not enough **samples** to compute Q^{π_k}

Approximate Dynamic Programming & Reinforcement Learning

Approximate Dynamic Programming Algorithms

Approximate Policy Iteration

- start with an arbitrary policy π_0
- at each iteration k
  - Policy Evaluation: Compute $\widehat{Q}^{\pi_k}$ $\qquad \widehat{Q}^{\pi_k} \approx Q^{\pi_k}$
  - Policy Improvement: Compute the *greedy* policy w.r.t. $\widehat{Q}^{\pi_k}$

$$\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}} \widehat{Q}^{\pi_k}(x, a) \qquad \forall x \in \mathcal{X}$$

Approximate Dynamic Programming Algorithms

Approximate Policy Iteration

- start with an arbitrary policy π_0
- at each iteration k
  - Policy Evaluation: Compute $\widehat{Q}^{\pi_k}$ $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$
  - Policy Improvement: Compute the *greedy* policy w.r.t. $\widehat{Q}^{\pi_k}$

$$\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}} \widehat{Q}^{\pi_k}(x,a) \qquad \forall x \in \mathcal{X}$$

$$\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}} \widehat{Q}^{\pi_k}(x,a) \neq (\mathcal{G}\pi_k)(x) \longrightarrow V^{\pi_{k+1}} \overset{?}{\geq} V^{\pi_k}$$

☹

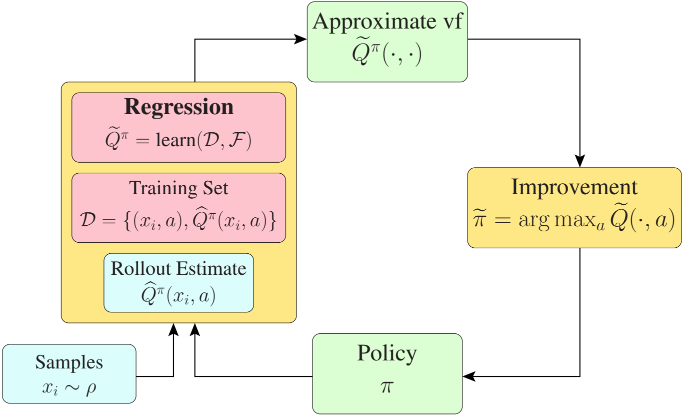
Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

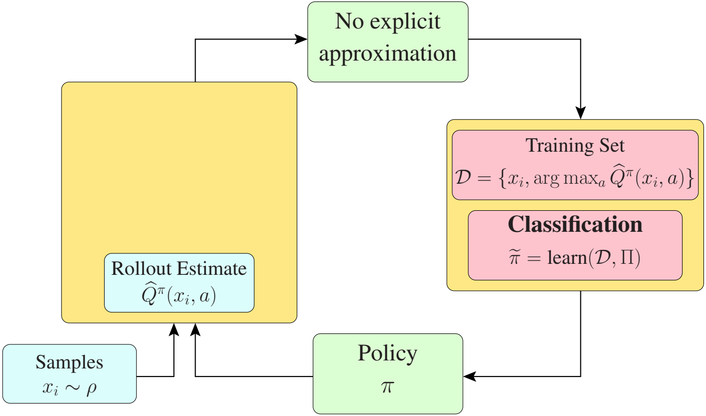
Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Value-based (Approximate) Policy Iteration



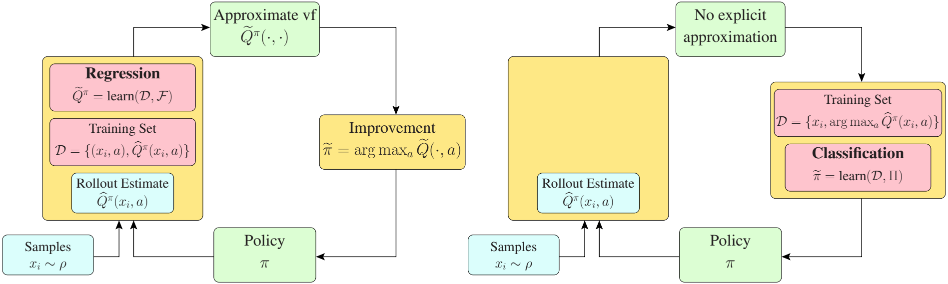
* We use Monte-Carlo estimation for illustration purposes

Classification-based Policy Iteration



* The idea first introduced by ***Lagoudakis & Parr (2003)*** and ***Fern et al. (2004)***

Value-based vs Classification-based Policy Iteration



Appealing Properties

- **Property 1.** More important to have a policy with a performance similar to the greedy policy w.r.t. Q^{π_k} than an accurate approximation of Q^{π_k}

- **Property 2.** In some problems good policies are easier to represent and learn than their corresponding value functions

Tetris

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

for $j = 1$ to M **do**

Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

end for

$\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M}\sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

end for

$\pi_{k+1} = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

  Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N, \; x_i \overset{\text{iid}}{\sim} \rho$

  for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

    for $j = 1$ to M **do**

      Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

      with $x^t \sim p\big(\cdot \,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

    end for

    $\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

  end for

  $\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

  Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

  for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

    for $j = 1$ to M **do**

      Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

      with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

    end for

    $\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

  end for

  $\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

for $j = 1$ to M **do**

Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

end for

$\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

end for

$\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

* How to select the sampling distribution ρ?

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N,\ x_i \overset{\text{iid}}{\sim} \rho$

for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

for $j = 1$ to M **do**

Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

end for

$\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M}\sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

end for

$\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

* How to select the sampling distribution ρ?

** Can we use the same set of samples for all iterations?

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

for $j = 1$ to M **do**

Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

end for

$\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M}\sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

end for

$\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

* How to select the sampling distribution ρ?

** Can we use the same set of samples for all iterations? **yes** ***(more complex analysis)***

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

for $j = 1$ to M **do**

Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

with $x^t \sim p\big(\cdot \,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

end for

$\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

end for

$\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

  Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, $x_i \overset{\text{iid}}{\sim} \rho$

  for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

    for $j = 1$ to M **do**

      Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

      with $x^t \sim p\big(\cdot \,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

    end for

    $\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

  end for

  $\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,; \pi)$ **(classifier)**

end for

* rollouts are allocated ***uniformly*** over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$. Other possibilities?

Template of the Algorithm

Input: policy space Π, state distribution ρ, number of rollout states N, number of rollouts per state-action pair M, rollout horizon H

Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy

for $k = 0, 1, 2, \dots$ **do**

  Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N,\ x_i \overset{\text{iid}}{\sim} \rho$

  for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**

    for $j = 1$ to M **do**

      Perform a rollout according to policy π_k and return

$$R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big),$$

      with $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$

    end for

    $\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M}\sum_{j=1}^M R_j^{\pi_k}(x_i, a)$

  end for

  $\pi_{k+1} = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,;\pi)$ **(classifier)**

end for

Gap-based Loss

- **Empirical Gap-based Error**

$$\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi) = \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}} \widehat{Q}^{\pi_k}(x_i,a) - \widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\Big]$$

Gap-based Loss

- **Empirical Gap-based Error**

$$\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi) = \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}} \widehat{Q}^{\pi_k}(x_i,a) - \widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\Big]$$

* $\widehat{\rho}$: empirical distribution induced by $\mathcal{D}_k$

Gap-based Loss

- **Empirical Gap-based Error**

$$\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi) = \frac{1}{N}\sum_{i=1}^{N}\left[\max_{a\in\mathcal{A}}\widehat{Q}^{\pi_k}(x_i,a) - \widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\right]$$

* $\widehat{\rho}$: empirical distribution induced by $\mathcal{D}_k$

** $\widehat{Q}^{\pi_k}(x_i,a)$: rollout estimation of $Q^{\pi_k}(x_i,a)$

Gap-based Loss

- **True Gap-based Error**

$$\mathcal{L}_{\pi_k}(\rho;\pi) = \mathbb{E}_{x\sim\rho}\left[\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a) - Q^{\pi_k}\big(x,\pi(x)\big)\right]$$

Gap-based vs. Mistake-based Errors

- **Gap-based Error** ***(weighted loss)***

$$\mathcal{L}_{\pi_k}(\rho;\pi) = \mathbb{E}_{x\sim\rho}\left[\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a) - Q^{\pi_k}\big(x,\pi(x)\big)\right]$$

$$= \int_{\mathcal{X}} \underbrace{\mathbb{I}\Big\{\pi(x) \neq \arg\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a)\Big\}}_{\textbf{mistake}} \underbrace{\left[\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a) - Q^{\pi_k}\big(x,\pi(x)\big)\right]}_{\textbf{cost/regret}} \rho(dx)$$

- **Mistake-based Error** ***(0/1 loss)***

$$\mathcal{L}_{\pi_k}(\rho;\pi) = \mathbb{E}_{x\sim\rho}\left[\mathbb{I}\Big\{\pi(x) \neq (\mathcal{G}\pi_k)(x)\Big\}\right]$$

$$= \int_{\mathcal{X}} \underbrace{\mathbb{I}\Big\{\pi(x) \neq \arg\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a)\Big\}}_{\textbf{mistake}} \rho(dx)$$

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Error at Each Iteration

Error at each Iteration (DPI)

Error at iteration k

$$||\pi_{k+1} - \mathcal{G}\pi_k||_{1,\rho} \leq f(B, \Pi, \delta) \qquad \text{w.p. } 1-\delta$$

Error at each Iteration (DPI)

Error at iteration k

$$||\pi_{k+1} - \mathcal{G}\pi_k||_{1,\rho} = \mathcal{L}_{\pi_k}(\rho\,;\pi_{k+1}) \leq f(B, \Pi, \delta) \qquad \text{w.p. } 1-\delta$$

Bound on the Error at each Iteration

Theorem

Let Π *be a policy space with* $h = VC(\Pi) < \infty$ *and* ρ *be a distribution over* $\mathcal{X}$. *Let* N *be the number of states in* $\mathcal{D}_k$ *drawn i.i.d. from* ρ, H *be the rollout horizon, and* M *be the number of rollouts per state-action pair. Let*

$$\pi_{k+1} = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}\,;\pi)$$

be the policy computed at the k*'th iteration of DPI . Then, for any* $\delta > 0$

$$\mathcal{L}_{\pi_k}(\rho\,;\pi_{k+1}) \leq \inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho\,;\pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max}),$$

with probability $1-\delta$, *where*

$$\epsilon_1 = 16Q_{\max}\sqrt{\frac{2}{N}\left(h\log\frac{eN}{h} + \log\frac{32}{\delta}\right)} \quad \textit{and} \quad \epsilon_2 = (1-\gamma^H)Q_{\max}\sqrt{\frac{2}{MN}\log\frac{4|\mathcal{A}|}{\delta}}\,.$$

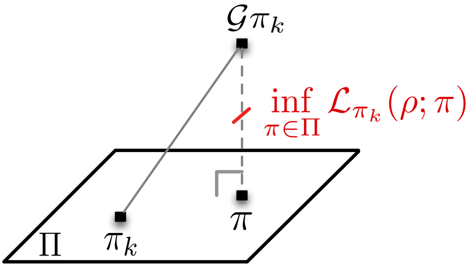
Remarks

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \leq \underbrace{\inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi)}_{\textbf{approximation error}} + 2\Big(\epsilon_1(N)+\epsilon_2(N,M,H)+\gamma^H Q_{\max}\Big)$$

Remarks

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \le \underbrace{\inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi)}_{\textbf{approximation error}} + 2\big(\epsilon_1(N)+\epsilon_2(N,M,H)+\gamma^H Q_{\max}\big)$$

- **approximation error:** depends on how well the policy space Π (***classifier***) can approximate the greedy policy $\mathcal{G}\pi_k$



Remarks

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \leq \underbrace{\inf_{\pi\in\Pi}\mathcal{L}_{\pi_k}(\rho;\pi)}_{\textbf{approximation error}} + \underbrace{2\Big(\epsilon_1(N) + \epsilon_2(N,M,H) + \gamma^H Q_{\max}\Big)}_{\textbf{estimation error}}$$

Remarks

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \le \underbrace{\inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi)}_{\textbf{approximation error}} + \underbrace{2\big(\epsilon_1(N) + \epsilon_2(N,M,H) + \gamma^H Q_{\max}\big)}_{\textbf{estimation error}}$$

- **estimation error**

$$\epsilon_1 = 16 Q_{\max}\sqrt{\frac{2}{N}\left(h\log\frac{eN}{h} + \log\frac{32}{\delta}\right)} \qquad \epsilon_2 = (1-\gamma^H)Q_{\max}\sqrt{\frac{2}{MN}\log\frac{4|\mathcal{A}|}{\delta}}$$

Remarks

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \leq \underbrace{\inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi)}_{\textbf{approximation error}} + \underbrace{2\big(\epsilon_1(N) + \epsilon_2(N,M,H) + \gamma^H Q_{\max}\big)}_{\textbf{estimation error}}$$

- **estimation error**

$$\epsilon_1 = 16 Q_{\max}\sqrt{\frac{2}{N}\left(h\log\frac{eN}{h} + \log\frac{32}{\delta}\right)} \qquad \epsilon_2 = (1-\gamma^H)Q_{\max}\sqrt{\frac{2}{MN}\log\frac{4|\mathcal{A}|}{\delta}}$$

- avoid overfitting (ϵ_1): take $N \gg h$
- fixed budget of rollouts $B = MN$: take $M = 1$ and $N = B$
- fixed budget $B = MNH$ and $M = 1$: take $O(\frac{\log B}{\log 1/\gamma})$ and $N = O(B/H)$

Proof

Main steps

- Bound on $\mathcal{L}_{\pi_k}(\rho\,;\pi_{k+1}) - \mathcal{L}_{\pi_k}(\widehat{\rho}\,;\pi_{k+1})$ using a VC-bound $\boldsymbol{\epsilon_1}$
- Replace $Q^{\pi_k}(x_i,a)$ with $Q_H^{\pi_k}(x_i,a)$ $\boldsymbol{\gamma^H Q_{\max}}$
- Bound on $\widehat{Q}^{\pi_k}(x_i,a) - Q_H^{\pi_k}(x_i,a)$ using Chernoff-Hoeffding $\boldsymbol{\epsilon_2}$
- π_{k+1} minimizes the empirical error $\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi)$

Error Propagation
&
Final Performance Bound

Final Performance Bound

Final Objective: Bound the error after K iteration of the alg.

$$||V^* - V^{\pi_K}||_{1,\mu} \leq f(B, \Pi, \delta, K) \qquad \text{w.p. } 1-\delta$$

π_K is the policy computed by the algorithm after K iterations

Final Performance Bound

Final Objective: Bound the error after K iteration of the alg.

$$||V^* - V^{\pi_K}||_{1,\mu} \leq f(B, \Pi, \delta, K) \qquad \text{w.p. } 1-\delta$$

π_K is the policy computed by the algorithm after K iterations

Error Propagation: How the error at each iteration $||\pi_{k+1} - \mathcal{G}\pi_k||_{1,\rho}$ propagates through the iterations of the algorithm

Pointwise Error Propagation

Lemma

Let π_k, π_{k+1}, and π_K be the policies learned by DPI at iterations k, $k+1$, and K, then we have

$$V^* - V^{\pi_K} \leq (\gamma P^*)^K (V^* - V^{\pi_0}) + \sum_{k=0}^{K-1} (\gamma P^*)^{K-k-1} E_k \, \ell_{\pi_k}(\pi_{k+1})$$

where $E_k = (I - \gamma P^{\pi_{k+1}})^{-1}$ and

$$\ell_{\pi_k}(x;\pi_{k+1}) = \max_{a \in \mathcal{A}} Q^{\pi_k}(x,a) - Q^{\pi_k}\big(x, \pi(x)\big), \qquad \forall x \in \mathcal{X}.$$

DPI Final Performance Bound

Theorem

Let Π *be a policy space with VC-dimension* h *and* π_K *be the policy generated by DPI after* K *iterations. Then, for any* $\delta > 0$

$$||V^* - V^{\pi_K}||_{1,\mu} \leq \frac{1}{(1-\gamma)^2} C_{\mu,\rho}\Big(d(\Pi, \mathcal{G}\Pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max})\Big) + 2\gamma^K Q_{\max} \quad \textbf{(A1)}$$

with probability $1-\delta$, *where*

$$\epsilon_1 = 16 Q_{\max}\sqrt{\frac{2}{N}\left(h \log\frac{eN}{h} + \log\frac{32K}{\delta}\right)} \qquad \textit{and}$$

$$\epsilon_2 = (1-\gamma^H) Q_{\max}\sqrt{\frac{2}{MN}\log\frac{4|\mathcal{A}|K}{\delta}} \; .$$

Concentrability Coefficient

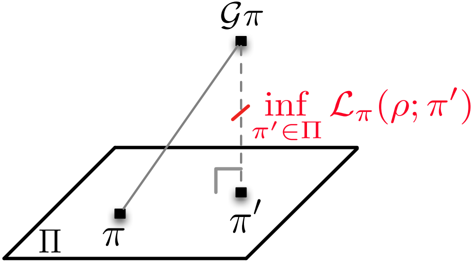
For any policy $\pi \in \Pi$ and any non-negative integers s and t, there exists a constant $C_{\mu,\rho}(s,t) < \infty$ such that

$$\mu(P^*)^s(P^\pi)^t \leq C_{\mu,\rho}(s,t)\ \rho$$

We define

$$C_{\mu,\rho} = (1-\gamma)^2 \sum_{s=0}^{\infty} \sum_{t=0}^{\infty} \gamma^{s+t} C_{\mu,\rho}(s,t)$$

Approximation Error



Inherent Greedy Error $\quad d(\Pi, \mathcal{G}\Pi) = \sup_{\pi \in \Pi} \inf_{\pi' \in \Pi} \mathcal{L}_\pi(\rho\,; \pi')$

An Open Question?

Q. rollouts are allocated ***uniformly*** over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$

An Open Question?

Q. rollouts are allocated ***uniformly*** over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$

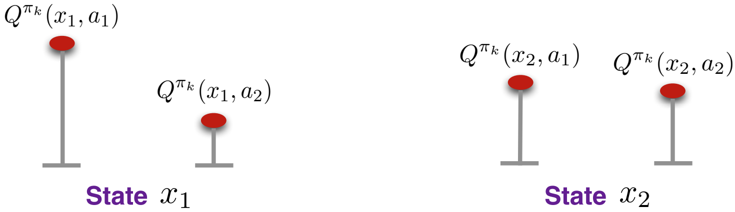
How to allocate a fixed budget of rollouts over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$ in order to have an accurate training set for the classifier???

An Open Question?

Q. rollouts are allocated ***uniformly*** over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$

How to allocate a fixed budget of rollouts over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$ in order to have an accurate training set for the classifier???

uniform allocation can be wasteful

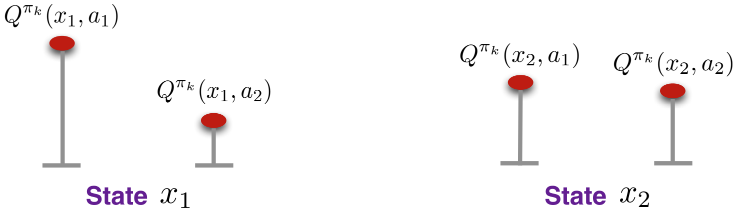


An Open Question?

Q. rollouts are allocated ***uniformly*** over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$

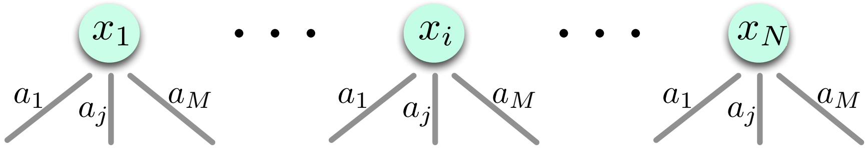
How to allocate a fixed budget of rollouts over $x \in \mathcal{D}_k$ and $a \in \mathcal{A}$ in order to have an accurate training set for the classifier???

uniform allocation can be wasteful

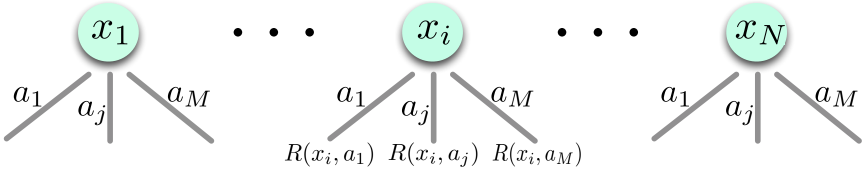


A. adaptive resource allocation


x_1
x_i
x_N

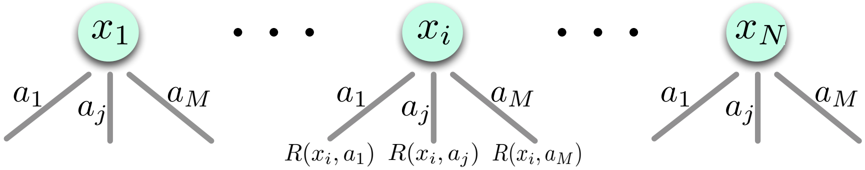

x_1
a_1
a_j
a_M
x_i
a_1
a_j
a_M
x_N
a_1
a_j
a_M

Given a fixed budget of rollouts B



$R(x_i, a_j)$ is a sample from a distribution whose mean value is $Q(x_i, a_j)$

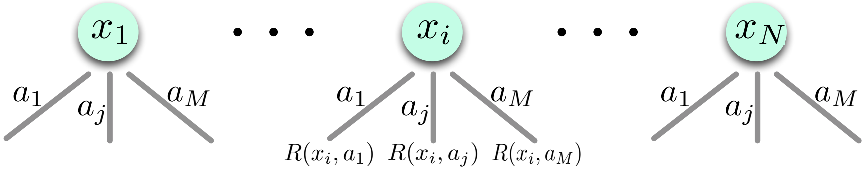
Given a fixed budget of rollouts B



$R(x_i, a_j)$ is a sample from a distribution whose mean value is $Q(x_i, a_j)$

each state x_i and action a_j has a distribution with the mean $Q(x_i, a_j)$

Given a fixed budget of rollouts B

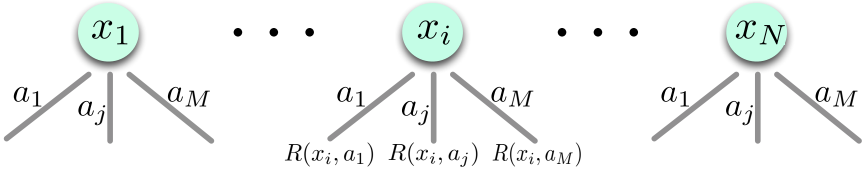


$R(x_i, a_j)$ is a sample from a distribution whose mean value is $Q(x_i, a_j)$

each state x_i and action a_j has a distribution with the mean $Q(x_i, a_j)$

How to allocate rollouts to maximize the probability of selecting the action with the highest mean value, Q, at each of these N states?

Given a fixed budget of rollouts B



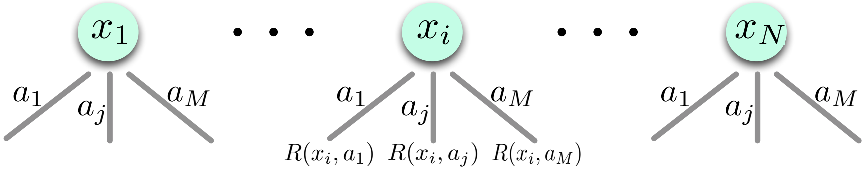
$R(x_i, a_j)$ is a sample from a distribution whose mean value is $Q(x_i, a_j)$

each state x_i and action a_j has a distribution with the mean $Q(x_i, a_j)$

How to allocate rollouts to maximize the probability of selecting the action with the highest mean value, Q, at each of these N states?

Multi-bandit Best Arm Identification

Given a fixed budget of rollouts B



$R(x_i, a_j)$ is a sample from a distribution whose mean value is $Q(x_i, a_j)$

each state x_i and action a_j has a distribution with the mean $Q(x_i, a_j)$

How to allocate rollouts to maximize the probability of selecting the action with the highest mean value, Q, at each of these N states?

Multi-bandit Best Arm Identification

GapE and GapE-V algorithms ***(Gabillon, MGH, Lazaric, NIPS-2011)***

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Production Line

Production Line

Given a fixed budget of tests

- Allocate these tests over the production lines, *such that*
- Estimate their average performance **as accurate as possible**

Test: run a production line and measure its performance

Online Advertisement

Online Advertisement – Online Polling

Given a fixed budget of ads

- Allocate this budget over several types of ads (products or services), *such that*
- Estimate their average preference **as accurate as possible**

There is a cost each time an ad is presented (e.g., web banner) to a random customer and her feedback is collected (customer clicks or not)

Clinical Trial

Clinical Trial

Given

- a fixed budget of clinical trials
- a number of subpopulations (patients with a particular gene biomarker)
- a number of available treatments for subjects from each subpopulation

Objective: construct a rule (from clinical trials) that recommends the best treatment for each of the subpopulations

Uniform Strategy

Uniform strategy:

- may waste the budget and have the risk of finding a bad treatment for a subpopulation
- more resources might be needed to find the best treatment for one subpopulation than the other

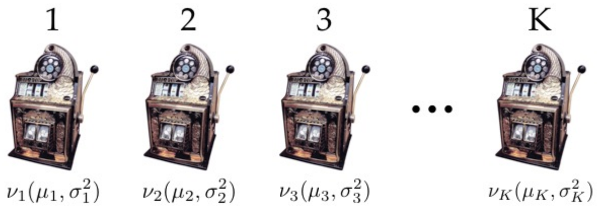
Outline

Overview
- Sequential Decision-making under Uncertainty
- Reinforcement Learning
- Reinforcement Learning Algorithms

Classification-based Policy Iteration Algorithms
- An Algorithm – Direct Policy Iteration (DPI)
- Finite-sample Performance Analysis of DPI

Resource Allocation
- Motivating Examples
- Resource Allocation as Stochastic Multi-armed Bandit

Stochastic Multi-Armed Bandits



Setting

- Number of arms $= K$, Total number of pulls $=$ budget $= n$
- each arm k is characterized by a distribution ν_k bounded in $[0, 1]$ with mean μ_k and variance σ_k^2
- at each round t, the algorithm pulls an arm $I(t)$ and observes a sample $X_{I(t)}(t) \sim \nu_{I(t)}$

Pure Exploration *(Bubeck et al. 2009; Audibert et al. 2010)*

Output: at the end of round n, the algorithm returns $J(n)$ some characteristics of the arms *(distributions)*

Objective: the returned characteristics of the arms *(distributions)* $J(n)$ to be as accurate as possible

Pure Exploration *(Bubeck et al. 2009; Audibert et al. 2010)*

Output: at the end of round n, the algorithm returns $J(n)$ some characteristics of the arms *(distributions)*

Objective: the returned characteristics of the arms *(distributions)* $J(n)$ to be as accurate as possible

In the **pure exploration** setting

- the algorithm is evaluated only based on its final output
- exploration phase and evaluation phase are separated

Best Arm Identification - Extensions

- m-**best arm identification:** finding the set of m-optimal arms
- (m, ϵ)-**best arm identification:** finding the set of (m, ϵ)-optimal arms
- **Fixed budget vs. Fixed confidence:** design a forecaster capable of
  - **Fixed budget:** finding a set of (m, ϵ)-optimal arms with the largest possible confidence, given the fixed budget of n rounds
  - **Fixed confidence:** stopping as soon as possible and returning a set of (m, ϵ)-optimal arms with a desired (fixed) confidence

Best Arm Identification - Extensions

- **m-best arm identification:** finding the set of m-optimal arms
- **(m, ϵ)-best arm identification:** finding the set of (m, ϵ)-optimal arms
- **Fixed budget vs. Fixed confidence:** design a forecaster capable of
  - **Fixed budget:** finding a set of (m, ϵ)-optimal arms with the largest possible confidence, given the fixed budget of n rounds
  - **Fixed confidence:** stopping as soon as possible and returning a set of (m, ϵ)-optimal arms with a desired (fixed) confidence

UGapEb and UGapEc algorithms ***(Gabillon, MGH, Lazaric, NIPS-2012)***

Thank you!!

we are looking for interns at Adobe Research

Mohammad Ghavamzadeh

mohammad.ghavamzadeh@inria.fr