Linear Bayesian Reinforcement Learning

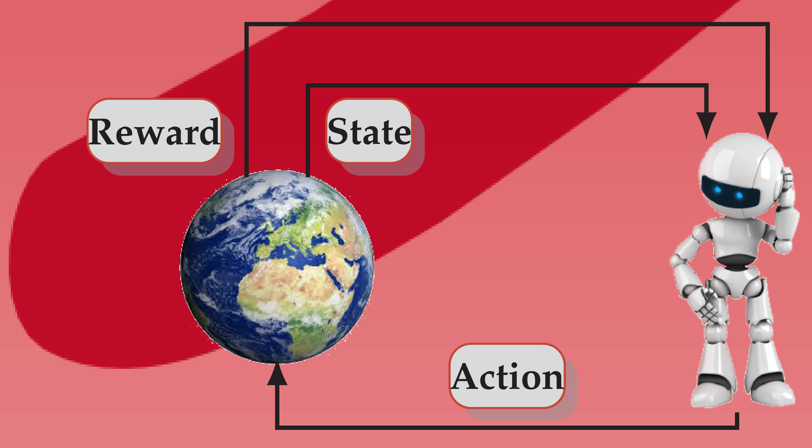
Nikolaos Tziortziotis[†], Christos Dimitrakakis[*] and Konstantinos Blekas[†]
[†]Department of Computer Science & Engineering, University of Ioannina, Greece
[*]Artificial Intelligence Laboratory, EPFL, Lausanne, Switzerland
email: {ntziorzi, christos.dimitrakakis}@gmail.com, kblekas@cs.uoi.gr

Abstract

✓ Simple linear Bayesian approach
✓ A Bayesian linear Gaussian model is used for estimating the system dynamics
✓ Policies are estimated by performing approximate dynamic programming on a sampled transition model
✓ Thompson sampling results in good exploration in unknown environments
✓ LBRL could be seen as a Bayesian generalisation of the LSPI

Reinforcement Learning (RL)

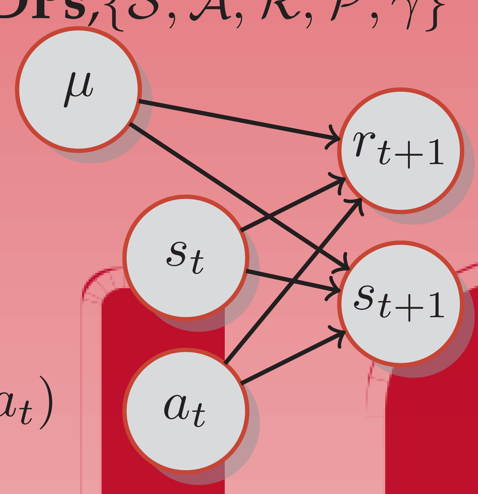
Learning to act in an unknown environment, by interaction and reinforcement.



RL tasks folmulated as **MDPs**, $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$

Environment μ; at time t:

1. Observe state $s_t \in \mathcal{S}$
2. Take action $a_t \in \mathcal{A}$
3. Receive $r_{t+1} = \mathcal{R}_\mu(s_t, a_t)$



Policy π: $\mathcal{S} \to \mathcal{A}$ (map states to actions)
Target: find π maximizing the expected utility

$$\mathbb{E}^\pi_\mu U = \mathbb{E}^\pi_\mu \sum_{t=0}^{\infty} \gamma^t r_t, \qquad (\mu \text{ is known})$$

Bayesian Reinforcement Learning

Decision-theoritic approach which use a *subjective* belief $\xi(\mu)$

$$\mathbb{E}^\pi_\xi U = \int_{\mathcal{M}} (\mathbb{E}^\pi_\mu U)\, \mathrm{d}\xi(\mu)$$

Handicaps:
- Future observations will alter our beliefs
- Planning must take into account future learning
- $\pi : (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^* \to \mathcal{A}$ must now map from *complete histories* to actions

Solutions:

✓ **Monte Carlo sampling**:

$$\mathbb{E}^\pi_\xi U \approx \frac{1}{K}\sum_{j=1}^{K} \mathbb{E}^\pi_{\mu_j} U, \qquad \mu_j \sim \xi.$$

✓ **Thompson sampling**:
Only a single MDP is sampled ($K = 1$)

Algorithmic Description

LBRL: **L**inear **B**ayesian **R**einforcement **L**earning

Input: Basis f, ADP parameters P, prior ξ_0
for *episode* k **do**
  $\mu^{(k)} \sim \xi_{t_k}(\mu)$ //Generate MDP from posterior
  $\pi^{(k)} = \mathtt{ADP}(\mu^{(k)}, P))$ //Get new policy
  for $t = t_k, \ldots, t_{k+1} - 1$ **do**
    $a_t \mid s_t = s \sim \pi^{(k)}(a \mid s)$ //Take action
    $\xi_{t+1}(\mu) = \xi_t(\mu \mid s_{t+1}, a_t, s_t)$//Update
  end
end

Predictive Linear Model

Assumption: $f : \mathcal{S} \to \mathcal{X}$
A separate linear model for each action $i \in \mathcal{A}$

$$s_{t+1} = \underbrace{A_i}_{\text{Design matrix}} x_t + \underbrace{\varepsilon_i}_{\text{noise}}, \quad x_t \triangleq f(s_t) = [s_t, 1]^\top$$

Probabilistic view of the model:

$$s_{t+1} \mid x_t = x, a_t = i \sim \mathcal{N}(A_i x, \underbrace{V_i}_{\text{covariance}}).$$

Conjugate priors:

$$A_i \mid V_i = V \sim \phi(A_i \mid \underbrace{M_i, C_i}_{\text{prior parameters}}, V) \quad (\textit{matrix-Normal})$$

$$V_i \sim \psi(V_i \mid \overbrace{W_i, n_i}) \quad (\textit{inverse-Wishart})$$

Posterior parameters: $M^t_i, W^t_i, n^t_i, C^t_i$

$$p_t(s_{t+1} \mid x_t = x, a_t = i) = \mathcal{S}t(M^t_i, W^t_i / z^t_i, 1 + n^t_i),$$

where $z^t_i = 1 - x^\top (C^t_i + x x^\top)^{-1} x$

✓ Closed-form calculation of posterior parameters

Approximate Policy Iteration

Value function $V^\pi_\mu : \mathcal{S} \to \mathbb{R}$, defined as:

$$V^\pi_\mu(s) \triangleq \mathbb{E}^\pi_\mu(U \mid s_t = s)$$
$$= \mathcal{R}(s) + \gamma \sum_{a \in \mathcal{A}} \int_{\mathcal{S}} V^\pi_\mu(s)\, \mathrm{d}P_\mu(s' \mid s, a)\pi(a \mid s)$$

Policy Evaluation

Find parameters $\omega \in \Omega$ minimising:

$$h(\omega) + \int_{\mathcal{S}} \left\| \mathrm{v}_\omega(s) - \mathcal{R}(s) - \gamma \int_{\mathcal{S}} \mathrm{v}_\omega(s')\, \mathrm{d}\hat{P}^\pi_\mu(s' | s) \right\|$$

Penalty term
Approximate value function
Approximate transition kernel

Policy Improvement

Improved policy π' maximises:

$$\mathcal{R}(s) + \gamma \sum_{a \in \mathcal{A}} \int_{\mathcal{S}} V^\pi_\mu(s)\, \mathrm{d}P_\mu(s' \mid s, a)\pi'(a \mid s)$$

Approximate Policy Iteration for a given μ

- Start with some policy π_k, where $k = 0$
1. Get parameters ω_k such that $V^{\pi_k}_\mu \approx \mathrm{v}_{\omega_k}$
2. Get improved policy π_{k+1} from v_{ω_k}
- Repeat until convergence

Experimental Results

Environments:
- Mountain Car
- Pendulum

Approximate Dynamic Programming (ADP) schemes:
- Fitted Value Iteration (FVI)
- Least Square Policy Temporal Difference (LSTD)

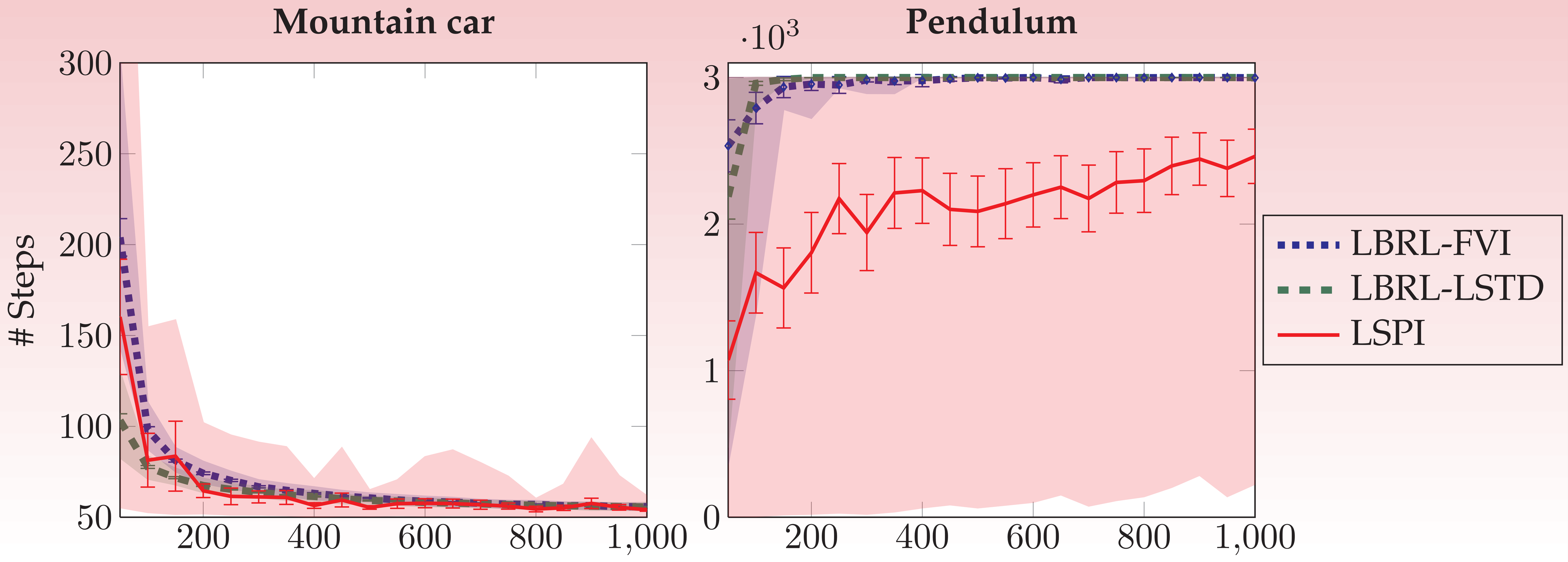
Benchmark: Least Square Policy Iteration (LSPI)
- LSTD directly on the observed data, without explicit model

Comparisons:
- Offline: Collect a set of trajectories (rollouts) from a *uniformly random policy*, separate test
- Online: Test performance while collecting data interactively using our *own generated policies*

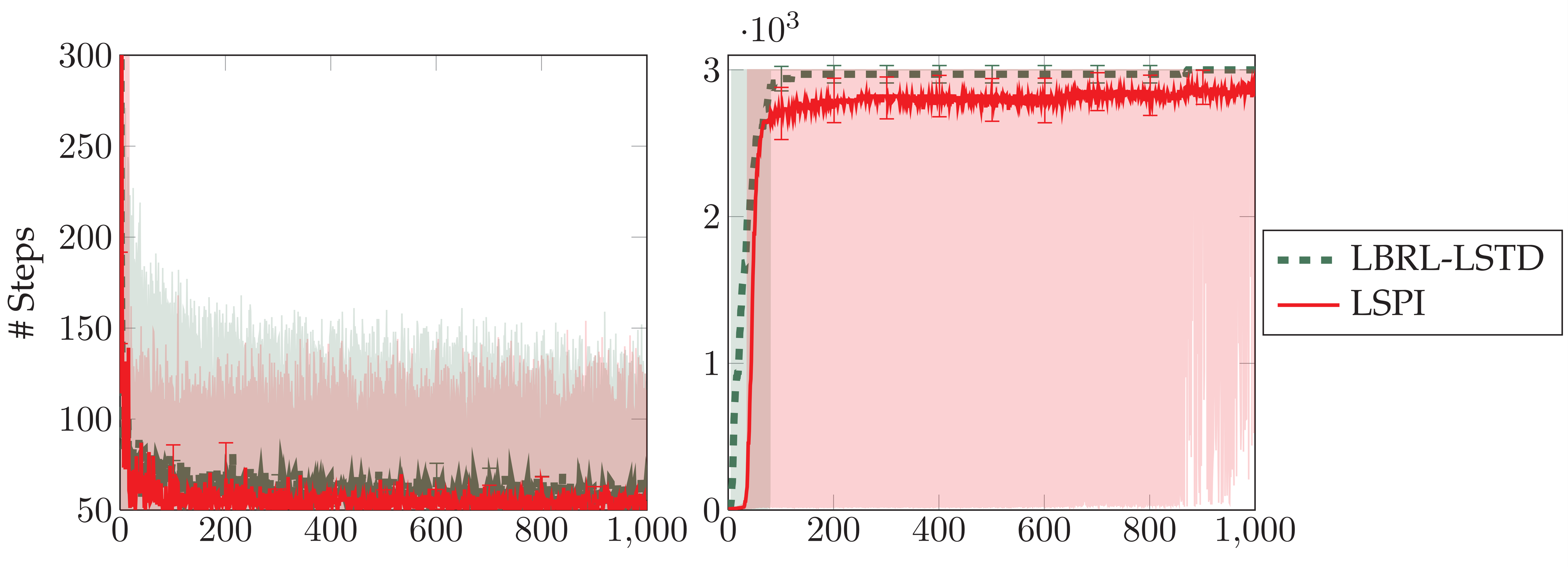
Why would LBRL be better than LSPI?

✓ If the environment is nearly linear, the model will fit well
✓ We can perform arbitrary computations in the sampled model
✓ Thompson sampling ensures continual exploration

Offline Performance



Online Performance



The error bars show 95% confidence intervals, while the shaded regions show 90% percentiles over 100 runs