Full Bayesian RL via LF-IBIS

Cecilia Viscardi

DISES, University of Salerno – email: cviscardi@unisa.it

joint with

Stefano Masini – PhD candidate, National Ph.D. in Artificial Intelligence for Society

Michela Baccini – DiSIA, University of Florence

SISBAYES 2025 - September 4, 2025

Why Reinforcement Learning?

Reinforcement Learning (RL) is a Machine Learning paradigm for decision-making. It has been studied since the 1980s (Watkins, 1989), but in the last decade it has become useful and widely popular in several fields:

- **Games**: from chess to modern video games;
- **Healthcare**: supports personalized medicine (e.g. dynamic treatment regimes);
- **AI**: trains and refines large AI models (e.g., ChatGPT) via RL from human feedback.

We aim to provide a method to approach Statistical RL from a *full Bayesian* perspective. This requires the formulation a new algorithm for likelihood-free inference: the **LF-IBIS**.

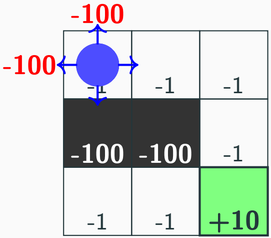
Outline of the talk

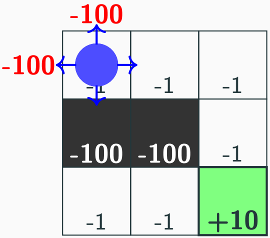
1. Reinforcement Learning
2. LF-IBIS
3. Results
4. Discussion

Reinforcement Learning

Main characters:

- **Agent** – blue dot
- **Environment** – the grid

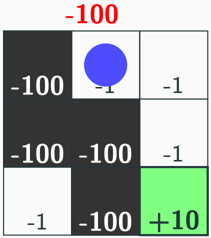




Step 1: Agent starts at top-left.

At each step n...

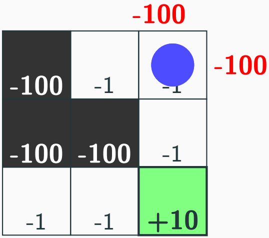
- The **agent** can take an **action** $a_n \in \{l, r, t, d\}$ (left, right, top, down).
- Its **policy** π is a probability distribution over $\{l, r, t, d\}$.



Step 2: Moves right. Reward = **-1**

At each step n...

- The environment changes its state s_n following probabilistic rules.
- Its behaviour depends on previous **actions** and a probability distribution governed by **parameters** μ.

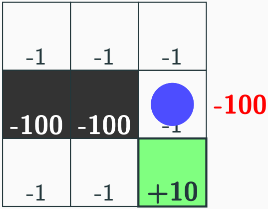


At each step n...

- To a pair (a_n, s_n) corresponds a **reward** r_n.

Step 3: Moves right again. Reward = **-1**

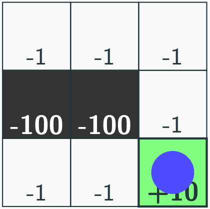
Reinforcement Learning (Sutton and Barto, 2018)



Step 4: Moves down. Reward = **-1**

AIM:

- The **agent** wants to reach the **goal** state while maximizing the reward.
- BUT its does not know the **environment** behaviour and μ.



Step 5: Reaches goal! Reward = **+10**

Statistical Framework 1/2

- The interactions **agent**/**environment** generate a **history**

$$\mathbf{x} = \{(a_n, s_n, r_n)\}_{n=1}^{N}$$

- $A_n \in \mathcal{A}$ and $S_n \in \mathcal{S}$ are (discrete) random variables
- The reward is a function $R_n : (\mathcal{A} \times \mathcal{S}) \rightarrow \mathbb{R}$

- SOME ASSUMPTIONS:

A1 $\mathbf{x}$ is the realization of Markov Decision Process:

$$S_{n+1} \perp\!\!\!\perp \{s_j, a_j\}_{j=1}^{n-1} \mid (a_n, s_n)$$

and the environment dynamic is described by transition matrices P_n;

A2 The process is homogeneous

$$\mathcal{S}_n \equiv \mathcal{S}, \mathcal{A}_n \equiv \mathcal{A}, P_n \equiv P \;\; \forall n$$

Statistical Framework 2/2

- The elements of the **environment**'s transition matrix

$$\Pr(s' \mid s, a, \mu)$$

- In classical RL, the agent uses the Bellman equation (Bellaman, 1958) to evaluate the *cumulative reward* through the **value function**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} \Pr(s' \mid s, a, \mu)[r(a, s') + \gamma V_\pi(s')]$$

- An **optimal policy** $\pi^*(a \mid s)$ is any policy that allows to reach the optimal value $V^*(s) = \max_\pi V_\pi(s)$ for each state.

Statistical Framework 2/2

- The elements of the **environment**'s transition matrix

$$\Pr(s' \mid s, a, \mu)$$

- In classical RL, the agent uses the Bellman equation (Bellaman, 1958) to evaluate the *cumulative reward* through the **value function**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} \Pr(s' \mid s, a, \mu)[r(a, s') + \gamma V_\pi(s')]$$

- An **optimal policy** $\pi^*(a \mid s)$ is any policy that allows to reach the optimal value $V^*(s) = \max_\pi V_\pi(s)$ for each state.

- AIM Computing $\pi^*(a \mid s)$
- PROBLEM $\Pr(s' \mid s, a, \mu)$is unknown

Bayesian Reinforcement Learning

- **Data**:

$$\mathbf{x} = x_{1:N} = \underbrace{\{(a_1, s_1, r_1), ..., (a_N, s_N, r_N)\}}_{\text{observed history}}$$

- **Likelihood**:

$$\Pr(X = x \mid \mu, \pi) = \rho(s_1) \prod_{n=1}^{N} \pi(a_n \mid s_n) \Pr(S_{n+1} = s_{n+1} \mid a_n, s_n, \mu)$$

$$L(\mu|\mathbf{x}) \propto \underbrace{\Pr\left(S_{1:N} = s_{1:N}|a_{1:N}, \mu\right)}_{f(\mathbf{x}|\mu)} = \prod_{n=1}^{N} \Pr\left(S_{n+1} = s_{n+1}|s_n, a_n, \mu\right)$$

- **Goal in Full Bayesian RL**:

$$p(\mu \mid \mathbf{x}) \propto p(\mu) f(x_{1:N} \mid \mu)$$

Complex posterior: why?

$$p(\mu \mid \mathbf{x}) \propto p(\mu) f(x_{1:N} \mid \mu)$$

The posterior uncertainty around μ can be propagated to

- The value function V_π
- The optimal policy π^*

Complex posterior: why?

$$p(\mu \mid \mathbf{x}) \propto p(\mu) f(x_{1:N} \mid \mu)$$

The posterior uncertainty around μ can be propagated to

- The value function V_π
- The optimal policy π^*

Problems:

1. **Online update**: data are not available in advance, but revealed step by step via interaction with the environment;

Complex posterior: why?

$$p(\mu \mid \mathbf{x}) \propto p(\mu) f(x_{1:N} \mid \mu)$$

The posterior uncertainty around μ can be propagated to

- The value function V_π
- The optimal policy π^*

Problems:

1. **Online update**: data are not available in advance, but revealed step by step via interaction with the environment;
2. **Intractable likelihood**: the environment's behaviour does not always have an analytical formulation, but it can be simulated through a computer program.

LF-IBIS

Likelihood-free Iterated Importance Sampling

To address these challenges, we propose a new likelihood-free algorithm: **LF-IBIS**.

- A hybrid algorithm that combines
  1. Approximate Bayesian Computation (ABC)
  2. Iterated Batch Importance Sampling (IBIS)

- Enables RL in both
  - online and offline settings
  - model-based and model-free settings

Likelihood-free inference: ABC (Sisson et al., 2018)

At each iteration $s \in \{1, ..., S\}$

1. Draw $\mu^{(s)}$ from $\underbrace{p(\cdot)}_{\text{prior}}$
2. Simulate y from $\underbrace{f(\cdot|\mu^{(s)})}_{\text{simulator}}$
3. Accept $\mu^{(s)}$ if $d(x, y) \leq \epsilon$

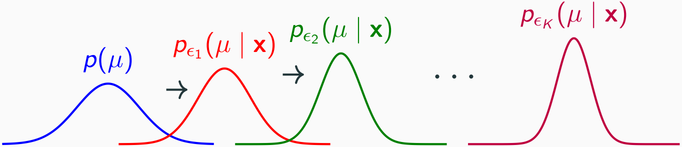
$$\underbrace{p_\epsilon(\mu \mid x)}_{\text{approximate posterior}} \quad \propto \quad p(\mu) \underbrace{\frac{1}{M_\mu} \sum_{m=1}^{M_\mu} \mathbb{1}[d(y_m^\mu, x) \leq \epsilon]}_{\text{approximate likelihood}}$$

ABC approximate likelihood

$$f(x \mid \mu) \approx \frac{1}{M_\mu} \sum_{m=1}^{M_\mu} K\big(d(y_m^\mu, x); \epsilon\big)$$

$$f(x \mid \mu) \approx \frac{1}{M_\mu} \sum_{m=1}^{M_\mu} K\big(d(\eta(y_m^\mu), \eta(x)); \epsilon\big)$$

Sequential Monte Carlo ABC (SMC-ABC) (Del Moral et al., 2012)



- **SMC-ABC** iterates Importance Sampling (IS) steps
- The algorithm uses a sequence $\epsilon_1 \geq \epsilon_2 ... \geq \epsilon$
- At *i*-th iteration, the proposed particles are samples from $p_{\epsilon_{i-1}}$ and the output is a sample from p_{ϵ_i}

Sequential Monte Carlo ABC (SMC-ABC) (Del Moral et al., 2012)

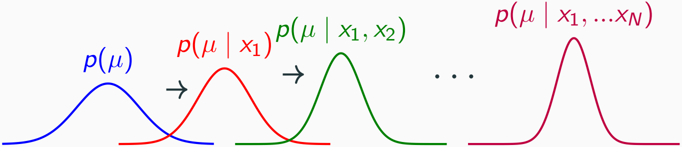
1. **Reweighting**:
   Update weights

$$\omega_i^{(s)} = \frac{\text{target}}{\text{proposal}} = \omega_{i-1}^{(s)} \times \frac{\sum_{m=1}^{M} K\big(d(\mathbf{x}, \mathbf{y}_m^{(s)}); \epsilon_i\big)}{\sum_{m=1}^{M} K\big(d(\mathbf{x}, \mathbf{y}_m^{(s)}); \epsilon_{i-1}\big)}$$

2. **Resampling**:
   Resample S particles using weights $(\omega_i^{(1)}, ..., \omega_i^{(S)})$ as probabilities

3. **Moving**:
   Move each $\mu_i^{(s)}$ using a Markov kernel of invariant density $p_{\epsilon_i}(\mu \mid \mathbf{x})$

Iterated Batch Importance Sampling (Chopin, 2002)



- **IBIS** iterates IS steps
- The output of each step i is a weighted sample from $p(\mu \mid x_{1:i})$
- At i-th iteration, the proposal distribution is $p(\mu \mid x_{1:i-1})$ and the target distribution is $p(\mu \mid x_{1:i})$

Iterated Batch Importance Sampling (Chopin, 2002)

1. **Reweighting**:
   Update weights

$$\omega_i^{(s)} = \frac{\text{target}}{\text{proposal}} = \omega_{i-1}^{(s)} \times \frac{f(x_{1:i} \mid \mu)}{f(x_{1:i-1} \mid \mu)} = \omega_{i-1} \times f(x_i \mid x_{1:i-1}, \mu)$$

2. **Resampling**:
   Resample S particles using weights $(\omega_i^{(1)}, \ldots, \omega_i^{(S)})$ as probabilities
3. **Moving**:
   Move each $\mu_i^{(s)}$ using a Markov kernel of invariant density $p(\mu \mid x_{1:i})$

Idea:

$$f(x_{n+1} \mid x_{1:n}, \mu) \approx \underbrace{\frac{\sum_{m=1}^{M} K\left(d(y_{1:n,m}), x_{1:n}); \epsilon\right)}{\sum_{m=1}^{M} K\left(d(y_{1:n-1,m}), x_{1:n-1}); \epsilon\right)}}_{\text{ABC-like estimate}}$$

LF-IBIS Algorithm

Algorithm 1 LF-IBIS

IS-ABC: $\mu_1^{(s)} \sim p(\cdot)$; $y_{1,m}^{(s)} \sim f(\cdot \mid \mu_1^{(s)})$ for $m \in \{1, \dots M\}$; $\omega_1^{(s)}(\epsilon_1) = \frac{1}{M} \sum_{m=1}^{M} K(d(x_1, y_{1,m}^{(s)}); \epsilon_1)$; $n \leftarrow 1$

while $n \leq N$ or $\epsilon_n > \epsilon$ **do**

  if no new agent-env interactions **then**

    Find a new ϵ^* adaptively

    Update weights:

$$\omega_n^{(s)}(\epsilon^*) = \omega_n^{(s)}(\epsilon_n) \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon^*)}{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}$$

    $\epsilon_n \leftarrow \epsilon^*$

  else

    Simulate and append $y_{n+1,m}^{(s)} \sim f(\cdot \mid \mu_n^{(s)}, y_{1:n,m}^{(s)})$ for $m \in \{1, \dots, M\}$

    $n \leftarrow n + 1$, $\epsilon_n \leftarrow \epsilon_{n-1}$

    Compute weights:

$$\omega_n^{(s)} = \omega_{n-1}^{(s)} \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}{\sum_{m=1}^{M} K(d(x_{1:n-1}, y_{1:n-1,m}^{(s)}); \epsilon_n)}$$

  Resample by using $\omega_n^{(1)} \dots \omega_n^{(S)}$

  Move $\mu^{(s)}$ through an MCMC kernele with acceptance ratio:

$$\text{ar} = \frac{p(\mu^*) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{*}); \epsilon_n) q(\mu^*, \mu_n^{(s)})}{p(\mu_n^{(s)}) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n) q(\mu_n^{(s)}, \mu^*)}$$

Output: For each $\mu_n^{(s)}$, compute policy $\pi^*(a \mid s) = \arg\max_\pi V_\pi(s; \mu_n^{(s)})$

LF-IBIS Algorithm

Algorithm 2 LF-IBIS

IS-ABC: $\mu_1^{(s)} \sim p(\cdot)$; $y_{1,m}^{(s)} \sim f(\cdot \mid \mu_1^{(s)})$ for $m \in \{1, \dots M\}$; $\omega_1^{(s)}(\epsilon_1) = \frac{1}{M} \sum_{m=1}^{M} K(d(x_1, y_{1,m}^{(s)}); \epsilon_1)$; $n \leftarrow 1$

while $n \leq N$ or $\epsilon_n > \epsilon$ **do**

  if no new agent-env interactions **then**

    Find a new ϵ^* adaptively

    Update weights:

$$\omega_n^{(s)}(\epsilon^*) = \omega_n^{(s)}(\epsilon_n) \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon^*)}{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}$$

    $\epsilon_n \leftarrow \epsilon^*$

  else

    Simulate and append $y_{n+1,m}^{(s)} \sim f(\cdot \mid \mu_n^{(s)}, y_{1:n,m}^{(s)})$ for $m \in \{1, \dots, M\}$

    $n \leftarrow n+1$, $\epsilon_n \leftarrow \epsilon_{n-1}$

    Compute weights:

$$\omega_n^{(s)} = \omega_{n-1}^{(s)} \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}{\sum_{m=1}^{M} K(d(x_{1:n-1}, y_{1:n-1,m}^{(s)}); \epsilon_n)}$$

  Resample by using $\omega_n^{(1)} \dots \omega_n^{(S)}$

  Move $\mu^{(s)}$ through an MCMC kernele with acceptance ratio:

$$\text{ar} = \frac{p(\mu^*) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{*}); \epsilon_n) q(\mu^*, \mu_n^{(s)})}{p(\mu_n^{(s)}) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n) q(\mu_n^{(s)}, \mu^*)}$$

Output: For each $\mu_n^{(s)}$, compute policy $\pi^*(a \mid s) = \arg\max_\pi V_\pi(s; \mu_n^{(s)})$

LF-IBIS Algorithm

Algorithm 3 LF-IBIS

IS-ABC: $\mu_1^{(s)} \sim p(\cdot)$; $y_{1,m}^{(s)} \sim f(\cdot \mid \mu_1^{(s)})$ for $m \in \{1, \dots M\}$; $\omega_1^{(s)}(\epsilon_1) = \frac{1}{M} \sum_{m=1}^{M} K(d(x_1, y_{1,m}^{(s)}); \epsilon_1)$; $n \leftarrow 1$

while $n \leq N$ or $\epsilon_n > \epsilon$ **do**

  if no new agent-env interactions **then**

    Find a new ϵ^* adaptively

    Update weights:

$$\omega_n^{(s)}(\epsilon^*) = \omega_n^{(s)}(\epsilon_n) \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon^*)}{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}$$

    $\epsilon_n \leftarrow \epsilon^*$

  else

    Simulate and append $y_{n+1,m}^{(s)} \sim f(\cdot \mid \mu_n^{(s)}, y_{1:n,m}^{(s)})$ for $m \in \{1, \dots, M\}$

    $n \leftarrow n + 1$, $\epsilon_n \leftarrow \epsilon_{n-1}$

    Compute weights:

$$\omega_n^{(s)} = \omega_{n-1}^{(s)} \cdot \frac{\sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n)}{\sum_{m=1}^{M} K(d(x_{1:n-1}, y_{1:n-1,m}^{(s)}); \epsilon_n)}$$

  Resample by using $\omega_n^{(1)} \dots \omega_n^{(S)}$

  Move $\mu^{(s)}$ through an MCMC kernele with acceptance ratio:

$$\text{ar} = \frac{p(\mu^*) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{*}); \epsilon_n) q(\mu^*, \mu_n^{(s)})}{p(\mu_n^{(s)}) \sum_{m=1}^{M} K(d(x_{1:n}, y_{1:n,m}^{(s)}); \epsilon_n) q(\mu_n^{(s)}, \mu^*)}$$

Output: For each $\mu_n^{(s)}$, compute policy $\pi^*(a \mid s) = \arg\max_\pi V_\pi(s; \mu_n^{(s)})$

LF-IBIS: some details

As an ABC algorithm, LF-IBIS has some specific characteristics that require the use of

- Specific **kernel function**

$$K\left(d(y^{(s)}_{1:i,m}, x_{1:i}); \epsilon_i\right) = \begin{cases} 1 & \text{if } d(y^{(s)}_{1:i,m}, x_{1:i}) \leq \epsilon_i \\ e^{-\dfrac{d(y^{(s)}_{1:i,m}, x_{1:i})}{\epsilon_i^2}} & \text{if } d(y^{(s)}_{1:i,m}, x_{1:i}) > \epsilon_i \end{cases}$$

- Suitable **criteria for adapting ϵ values**
    1. Effective Sample Size (ESS)
    2. Number of Unique Particles (UP)
- **Similarity criteria**
    1. Observation-based
    $$d\left(\eta(y^{(s)}_{1:i,m}), \eta(x_{1:i})\right) = \frac{1}{\sqrt{2}} \sqrt{\sum_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} \left(\sqrt{q_{s'|(s,a)}} - \sqrt{p_{s'|(s,a)}}\right)^2}$$
    2. Utility-based
    $$d\left(\eta(y^{(s)}_{1:i,m}), \eta(x_{1:i})\right) = \left| U_{x_{1:i}} - U_{y^{(s)}_{1:i,m}} \right|$$

Results

Toy example: Response-Adaptive Randomization (Deliu, 2021)

- The agent aims to allocate patients to treatment/placebo to maximize the number of disease remissions: $A_n \in \mathcal{A} = \{0, 1\}$

$$S_n = R_n = \begin{cases} 1 & \text{if the } n\text{-th patient goes into remission} \\ 0 & \text{otherwise} \end{cases}$$

- For each n under the same policy $\pi(A_n = 1 \mid S_{n-1} = 0) = \pi(A_n = 1 \mid S_{n-1} = 1) = \pi$
- μ_0 and μ_1 are the remission probabilities in the control and in the treatment group, respectively
- Environment's responses are: $Z_0 \sim \text{Ber}(\mu_0)$ and $Z_1 \sim \text{Ber}(\mu_1)$, where Z_0 denotes $S_n \mid A_n = 0$ and Z_1 denotes $S_n \mid A_n = 1$

Some results

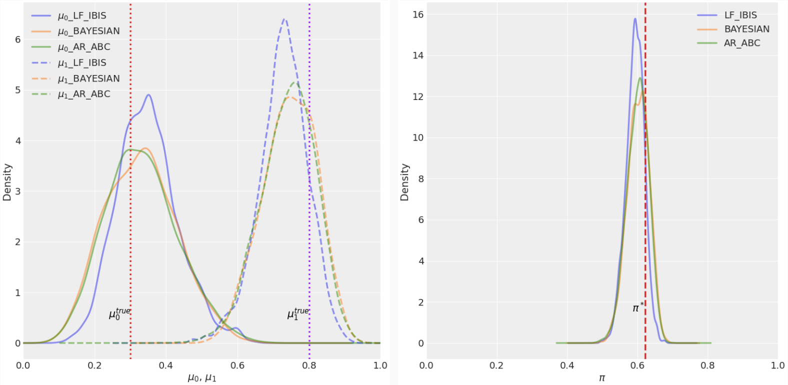


Figure 1: Posterior distributions for μ_0 and μ_1 (left) and π^* (right) along with true values.

TRUE POSTERIOR ; AR-ABC with full data; obs-based LF-IBIS with ESS.

Some results

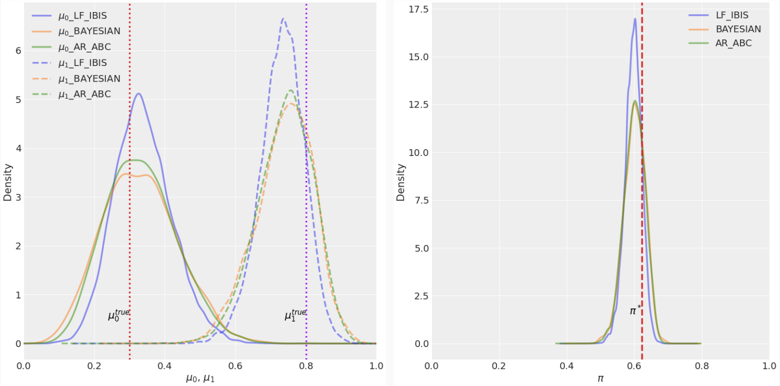


Figure 2: Posterior distributions for μ_0 and μ_1 (left) and π^* (right) along with true values.

TRUE POSTERIOR ; AR-ABC with full data; obs-based LF-IBIS with UP.

Some results

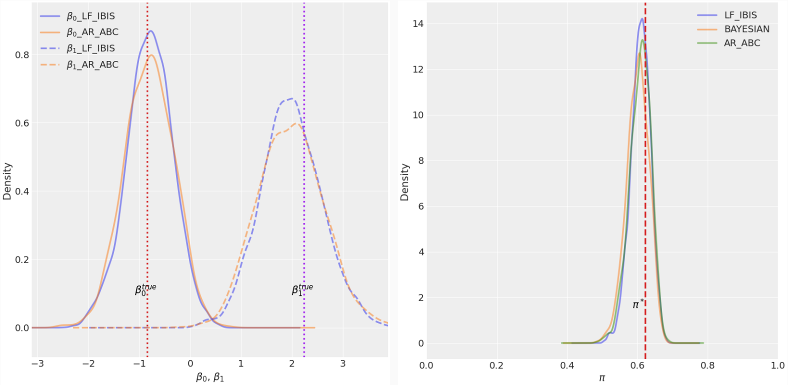


Figure 3: Posterior distributions for $\beta_0 = logit(\mu_0)$ and $\beta_1 = logit(\mu_1) - \beta_0$ (left) and π^* (right) along with true values.
TRUE POSTERIOR ; AR-ABC with full data; obs-based LF-IBIS with ESS.

Theoretical results (WIP)

Heuristically, under some mild assumptions, one can argue that:

1. As $\epsilon \to 0, M \to \infty$

$$\underbrace{\tilde{f}_{M,\epsilon}(x_{n+1} \mid x_{1:n}, \mu)}_{\text{approximate reweighting factor}} \quad \to f(x_{n+1} \mid x_{1:n}, \mu)$$

2. As $\epsilon \to 0$, the kernel $K\big(d(y,x);\epsilon\big)$ converges to an indicator function, and the proof of the convergence of the approximate posterior to the true posterior applies (Prangle et al., 2018).

Discussion

Discussion and conclusion

- We propose a novel strategy for full Bayesian Reinforcement Learning (fBRL).
- The approach relies on a new likelihood-free algorithm.
- The algorithm allows for continuous updating of estimates and supports environments with black-box models.
- Its effectiveness was demonstrated empirically using a toy example.
- The method is flexible and can be extended beyond the RL setting.
- Future work will focus on several extensions:
  - Handling batches of size greater than one.
  - Developing adaptive strategies to address the exploration–exploitation trade-off (WIP).
  - Providing theoretical results and applying the method to real-world scenarios (WIP).

References

Watkins, C. J. C. H.: Learning from delayed rewards. (1989). Phd thesis King's College.

Bellman, R.: A Markovian Decision Process. In: J. Appl. Math. Mech, pp. 679-684 (1957).

Sutton, R., Barto, A.: Reinforcement learning: An introduction. MIT press (2018).

Chopin, N.: A Sequential Particle Filter Method for Static Models. Biometrika, 89, no. 3, pp. 539–51 (2002).

Del Moral, P., Doucet, A., Jasra, A.: An adaptive sequential Monte Carlo method for approximate Bayesian computation. Stat Comput 22, 1009–1020 (2012).

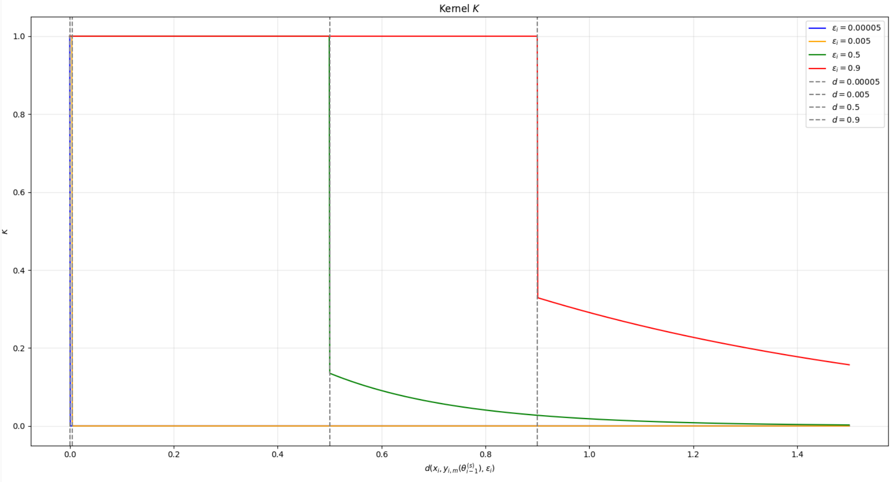
Dimitrakakis, C., Tziortziotis, N.: ABC Reinforcement Learning. 30th International Conference on Machine Learning (2013).

Deliu, N.: Reinforcement learning in modern biostatistics: benefits, challenges and new proposals, (2021).

Prangle, D., Everitt, R.G. and Kypraios, T, : A rare event approach to high-dimensional approximate Bayesian computation.

Sisson, Scott A., Fan, Y. and Beaumont,M. (eds): Handbook of Approximate Bayesian computation. Stat Comput 28, 819–834 (2018).

Kernel function



Details about experiments

- Final $\epsilon \approx 0.02$ (≈ 0.03 with different parametrization)
- $S = 10000$ (ESS); $S = 50000$ (UP)
- $M = 50$
- $\alpha = 0.98$
- Initial length of the history 10, $N = 48$

Further results

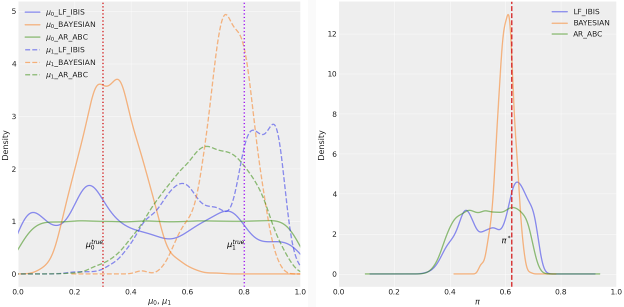


Figure 4: Posterior distributions for μ_0 and μ_1 (left) and π^* (right) along with true values.

TRUE POSTERIOR ; AR-ABC with full data; utility-based LF-IBIS with ESS.

Further results