A few reinforcement learning stories

R. Fonteneau
University of Liège, Belgium

May 5th, 2017
ULB - IRIDIA
Outline

Context: machine learning & (deep) reinforcement learning in brief

Batch Mode Reinforcement Learning

Synthesizing Artificial Trajectories

Estimating the Performances of Policies
Context: machine learning and (deep) reinforcement learning in brief
Machine learning is about extracting {patterns, knowledge, information} from data.
Machine learning algorithms have recently shown impressive results, in particular when input data are images: this has led to the identification of a subfield of Machine Learning called Deep Learning.
Reinforcement learning, an area of machine learning originally inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Deep reinforcement learning combines deep learning with reinforcement learning (and, consequently, in DP / MPC schemes).
Recent (Deep) Reinforcement Learning Successes

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis
Batch Mode Reinforcement Learning
Reinforcement Learning

Reinforcement Learning (RL) aims at finding a policy maximizing received rewards by interacting with the environment.

Examples of rewards:

- Reinforcement Learning (RL) aims at finding a policy maximizing received rewards by interacting with the environment.
Batch Mode Reinforcement Learning

- All the available information is contained in a **batch collection of data**
- Batch mode RL aims at computing a (near-)optimal policy from this collection of data

Finite collection of trajectories of the agent
Near-optimal decision strategy
Batch Mode Reinforcement Learning
Batch Mode Reinforcement Learning

Batch collection of trajectories of patients
Objectives

• Main goal: **Finding a "good" policy**

• Many associated subgoals:
  - Evaluating the performance of a given policy
  - Computing performance guarantees
  - Computing safe policies
  - Choosing how to generate additional transitions
  - ...
Main Difficulties

Main difficulties of the batch mode setting:

- Dynamics and reward functions are unknown (and not accessible to simulation)
- The state-space and/or the action space are large or continuous
- The environment may be highly stochastic
Usual Approach

To combine dynamic programming with function approximators (neural networks, regression trees, SVM, linear regression over basis functions, etc)

Function approximators have two main roles:

- To offer a concise representation of state-action value function for deriving value / policy iteration algorithms
- To generalize information contained in the finite sample
Remaining Challenges

The **black box nature of function approximators** may have some unwanted effects:

- hazardous generalization
- difficulties to compute performance guarantees
- unefficient use of optimal trajectories

A proposition: **synthesizing artificial trajectories**
Synthesizing Artificial Trajectories
Formalization

Reinforcement learning

System dynamics: \( x_{t+1} = f(x_t, u_t, w_t) \)
\[ t \in \{0, \ldots, T - 1\} \quad x_t \in \mathcal{X} \subset \mathbb{R}^d \quad u_t \in \mathcal{U} \quad w_t \in \mathcal{W} \]
\[ w_t \sim p_{\mathcal{W}}(\cdot) \]

Reward function: \( r_t = \rho(x_t, u_t, w_t) \)

Performance of a policy \( h : \{0, \ldots, T - 1\} \times \mathcal{X} \to \mathcal{U} \)

\[
J^h(x_0) = \mathbb{E}\left[ R^h(x_0, w_0, \ldots, w_{T-1}) \right]
\]

\[
R^h(x_0, w_0, \ldots, w_{T-1}) = \sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t)
\]

where

\[
x_{t+1} = f(x_t, h(t, x_t), w_t)
\]
Formalization

Batch mode reinforcement learning

The system dynamics, reward function and disturbance probability distribution are **unknown**

Instead, we have access to a **sample of one-step system transitions**:

\[
\mathcal{F}_n = \{(x^l, u^l, r^l, y^l)\}_{l=1}^n \quad \forall l \in \{1, \ldots, n\},
\]

\[
r^l = \rho(x^l, u^l, w^l)
\]

\[
y^l = f(x^l, u^l, w^l)
\]

\[
w^l \sim p_W(\cdot)
\]
Artificial trajectories are (ordered) sequences of elementary pieces of trajectories:

\[
\left[ \left( x_l^0, u_l^0, r_l^0, y_l^0 \right), \ldots, \left( x_l^{T-1}, u_l^{T-1}, r_l^{T-1}, y_l^{T-1} \right) \right] \in \mathcal{F}_n^T
\]

\[
l_t \in \{1, \ldots, n\}, \quad \forall t \in \{0, \ldots, T - 1\}
\]
Artificial Trajectories: What For?

Artificial trajectories can help for:

- Estimating the performances of policies
- Computing performance guarantees
- Computing safe policies
- Choosing how to generate additional transitions
Artificial Trajectories: What For?

Artificial trajectories can help for:

- Estimating the performances of policies
- Computing performance guarantees
- Computing safe policies
- Choosing how to generate additional transitions
Estimating the Performances of Policies
Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then Monte Carlo estimation would allow estimating the performance of $h$. 
Model-free Monte Carlo Estimation

\[ \mathbb{M}_3^h(x_0) = \left( r_{l_0}^1 + r_{l_1}^1 + r_{l_2}^1 + r_{l_3}^1 \right) + \left( r_{l_0}^2 + r_{l_1}^2 + r_{l_2}^2 + r_{l_3}^2 \right) + \left( r_{l_0}^3 + r_{l_1}^3 + r_{l_2}^3 + r_{l_3}^3 \right) \]
Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then Monte Carlo (MC) estimation would allow estimating the performance of $h$

We propose an approach that mimics MC estimation by rebuilding $p$ artificial trajectories from one-step system transitions
If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo (MC) estimation** would allow estimating the performance of $h$

We propose an approach that mimics MC estimation by rebuilding $p$ artificial trajectories from one-step system transitions. These artificial trajectories are built so as to **minimize the discrepancy (using a distance metric $\Delta$) with a classical MC sample** that could be obtained by simulating the system with the policy $h$; each one step transition is used at most once.
If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo (MC) estimation** would allow estimating the performance of $h$

We propose an approach that mimics MC estimation by rebuilding $p$ **artificial trajectories** from one-step system transitions.

These artificial trajectories are built so as to **minimize the discrepancy (using a distance metric $\Delta$)** with a classical MC **sample** that could be obtained by simulating the system with the policy $h$; each one step transition is used **at most once**.

We average the cumulated returns over the $p$ artificial trajectories to obtain the **Model-free Monte Carlo estimator** (MFMC) of the expected return of $h$:

$$M_p^h (F_n, x_0) = \frac{1}{p} \sum_{i=1}^{p} \sum_{t=0}^{T-1} r_{l_t}^i$$
Model-free Monte Carlo Estimation

\[ M_3^h (F_n, x_0) = \frac{(r_0^1 + r_1^1 + r_2^1 + r_0^1) + (r_0^2 + r_1^2 + r_2^2 + r_3^2) + (r_3^3 + r_1^3 + r_2^3 + r_3^3)}{3} \]
The MFMC algorithm

Example with $T = 3$, $p = 2$, $n = 8$

$$\mathcal{F}_n = \left\{ (x^l, u^l, r^l, y^l) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X} \right\}_{l=1}^n$$
The MFMC algorithm

\((x_0, h(0, x_0))\)
The MFMC algorithm

\[ G = \mathcal{F}_n \]
The MFMC algorithm

$G = \mathcal{F}_n$
The MFMC algorithm

\[
\left( y_{t_0}^1, h(1, y_{t_0}^1) \right)
\]

\[
\left( x_{t_0}^1, u_{t_0}^1, r_{t_0}^1, y_{t_0}^1 \right)
\]

\[
(x_0, h(0, x_0))
\]

\[
\mathcal{G} = \mathcal{G} \setminus \left\{ (x_{t_0}^1, u_{t_0}^1, r_{t_0}^1, y_{t_0}^1) \right\}
\]
The MFMC algorithm

\[ (y^l_0, h(1, y^l_0)) \]

\[ (x^l_0, u^l_0, r^l_0, y^l_0) \]

\[ (x_0, h(0, x_0)) \]

\[ \mathcal{G} = \mathcal{G} \setminus \{ (x^l_0, u^l_0, r^l_0, y^l_0) \} \]
The MFMC algorithm

\[ \mathcal{G} = \mathcal{G} \setminus \{(x^{l_0}_{1}, u^{l_0}_{1}, r^{l_0}_{1}, y^{l_0}_{1})\} \]
The MFMC algorithm

\[ [(y^{l_0}_0, h(1, y^{l_0}_0)), (y^{l_1}_1, h(2, y^{l_1}_1))] \]

\[ (x^{l_1}_0, u^{l_1}_0, r^{l_1}_0, y^{l_1}_0) \]

\[ (x^{l_1}_1, u^{l_1}_1, r^{l_1}_1, y^{l_1}_1) \]

\[ (x_0, h(0, x_0)) \]

\[ G = G \setminus \{(x^{l_1}_1, u^{l_1}_1, r^{l_1}_1, y^{l_1}_1)\} \]
The MFMC algorithm

\[ (x_0^l, h(0, x_0)) \]  
\[ (x_1^l, u_1^l, r_1^l, y_1^l) \]  
\[ (y_0^l, h(1, y_0^l)) \]  
\[ (y_1^l, h(2, y_1^l)) \]  
\[ \mathcal{G} = \mathcal{G} \setminus \{(x_1^l, u_1^l, r_1^l, y_1^l)\} \]
The MFMC algorithm

\[ \mathcal{G} = \mathcal{G} \setminus \{(x_{l_1}^{l_1}, u_{l_1}^{l_1}, r_{l_1}^{l_1}, y_{l_1}^{l_1})\} \]
The MFMC algorithm

\[ \mathcal{G} = \mathcal{G} \setminus \{(x_{l2}^1, u_{l2}^1, r_{l2}^1, y_{l2}^1)\} \]

\[ T-1 \sum_{t=0} r_{l_t}^1 \]
The MFMC algorithm

\[ \mathcal{G} = \mathcal{G} \setminus \{(x_{t_2}^{l_1}, u_{t_2}^{l_1}, r_{t_2}^{l_1}, y_{t_2}^{l_1})\} \]

\[ \sum_{t=0}^{T-1} \tau_{l_t}^{l_1} \]
The MFMC algorithm

\[ G = G \setminus \{(x^{l_1}_{t_2}, u^{l_1}_{t_2}, r^{l_1}_{t_2}, y^{l_1}_{t_2})\} \]

\[ \sum_{t=0}^{T-1} r^{l_1}_t \]

\[ (x^{l_1}_0, h(0, x_0)) \]

\[ (x^{l_1}_1, u^{l_1}_1, r^{l_1}_1, y^{l_1}_1) \]

\[ (x^{l_1}_1, h(1, y^{l_1}_0)) \]

\[ (y^{l_1}_0, h(2, y^{l_1}_1)) \]

\[ (x^{l_1}_2, u^{l_1}_2, r^{l_1}_2, y^{l_1}_2) \]
The MFMC algorithm

\[ T-1 \sum_{t=0}^{\tau} r_t^{l_1} \]

\[ (x_{t_0}^{l_1}, u_{t_0}^{l_1}, r_{t_0}^{l_1}, y_{t_0}^{l_1}) \]

\[ (x_{t_1}^{l_1}, u_{t_1}^{l_1}, r_{t_1}^{l_1}, y_{t_1}^{l_1}) \]

\[ (y_{t_0}^{l_1}, h(1, y_{t_0}^{l_1})) \]

\[ (y_{t_1}^{l_1}, h(2, y_{t_1}^{l_1})) \]

\[ (x_{t_2}^{l_1}, u_{t_2}^{l_1}, r_{t_2}^{l_1}, y_{t_2}^{l_1}) \]

\[ (x_{t_0}^{l_2}, u_{t_0}^{l_2}, r_{t_0}^{l_2}, y_{t_0}^{l_2}) \]

\[ (x_{t_1}^{l_2}, u_{t_1}^{l_2}, r_{t_1}^{l_2}, y_{t_1}^{l_2}) \]

\[ (y_{t_0}^{l_2}, h(1, y_{t_0}^{l_2})) \]

\[ (y_{t_1}^{l_2}, h(2, y_{t_1}^{l_2})) \]

\[ G = G \setminus \{ (x_{t_0}^{l_2}, u_{t_0}^{l_2}, r_{t_0}^{l_2}, y_{t_0}^{l_2}) \} \]
The MFMC algorithm

\[
\begin{align*}
(x^l_0, u^l_0, r^l_0, y^l_0) & \\
(x^l_0, h(0, x^l_0)) & \\
(y^l_0, h(1, y^l_0)) & \\
(y^l_1, h(2, y^l_1)) & \\
(x^l_2, u^l_2, r^l_2, y^l_2) & \\
T-1 \sum_{t=0} r^l_t & \\
\end{align*}
\]

\[G = G \setminus \{(x^l_0, u^l_0, r^l_0, y^l_0)\} \]
The MFMC algorithm

\[ \mathcal{G} = \mathcal{G} \setminus \left\{ (x_0^{l_0}, u_0^{l_0}, r_0^{l_0}, y_0^{l_0}) \right\} \]

\[ T - 1 \sum_{t=0}^{T-1} r_t^{l_1} \]
The MFMC algorithm

\[
\begin{align*}
(x_0, h(0, x_0)) & \quad (x_0^l, u_0^l, r_0^l, y_0^l) & \quad (x_0^l, u_0^l, r_0^l, y_0^l) \\
(y_0^l, h(1, y_0^l)) & \quad (y_1^l, h(2, y_1^l)) & \quad (x_1^l, u_1^l, r_1^l, y_1^l) & \quad (x_1^l, u_1^l, r_1^l, y_1^l) \\
(x_1^l, u_1^l, r_1^l, y_1^l) & \quad (x_1^l, u_1^l, r_1^l, y_1^l) & \quad (x_2^l, u_2^l, r_2^l, y_2^l) & \quad (x_2^l, u_2^l, r_2^l, y_2^l) \\
(y_0^l, h(1, y_0^l)) & \quad (y_1^l, h(2, y_1^l)) & \quad (x_2^l, u_2^l, r_2^l, y_2^l) & \quad (x_2^l, u_2^l, r_2^l, y_2^l) \\
\end{align*}
\]

\[
\mathcal{G} = \mathcal{G} \setminus \{(x_1^l, u_1^l, r_1^l, y_1^l)\}
\]
The MFMC algorithm

\[ (y_{l_0}^1, h(1, y_{l_0}^1)) \quad (y_{l_1}^1, h(2, y_{l_1}^1)) \]

\[ (x_{l_0}^1, u_{l_0}^1, r_{l_0}^1, y_{l_0}^1) \quad (x_{l_1}^1, u_{l_1}^1, r_{l_1}^1, y_{l_1}^1) \]

\[ (x_{l_0}^2, u_{l_0}^2, r_{l_0}^2, y_{l_0}^2) (x_{l_1}^2, u_{l_1}^2, r_{l_1}^2, y_{l_1}^2) \]

\[ (y_{l_0}^2, h(1, y_{l_0}^2)) \quad (y_{l_1}^2, h(2, y_{l_1}^2)) \]

\[ T-1 \sum_{t=0}^{-1} r_{l_t}^1 \]

\[ G = G \setminus \{(x_{l_1}^2, u_{l_1}^2, r_{l_1}^2, y_{l_1}^2)\} \]
The MFMC algorithm

\[
\begin{align*}
(y^l_0, h(1, y^l_0)) & \quad (y^l_1, h(2, y^l_1)) \\
(x^l_0, u^l_0, r^l_0, y^l_0) & \quad (x^l_1, u^l_1, r^l_1, y^l_1) \\
(x^l_0, h(0, x_0)) & \\
(x^l_2, u^l_2, r^l_2, y^l_2) & \\
(x^l_1, u^l_1, r^l_1, y^l_1) & \\
(x^l_0, u^l_0, r^l_0, y^l_0) (x^l_1, u^l_1, r^l_1, y^l_1) & \\
(y^l_0, h(1, y^l_0)) & \quad (y^l_1, h(2, y^l_1)) \\
\mathcal{G} = \mathcal{G} \setminus \{(x^l_1, u^l_1, r^l_1, y^l_1)\} & \\
\end{align*}
\]

\[
T-1 \sum_{t=0} r^l_t
\]
The MFMC algorithm
The MFMC algorithm

\[ \mathcal{M}_2^h(\mathcal{F}_8, x_0) = \frac{1}{2} \left( \sum_{t=0}^{2} r_{l_t}^1 + \sum_{t=0}^{2} r_{l_t}^2 \right) \]
Theoretical Analysis

Assumptions

Lipschitz continuity assumptions:

\[ \exists L_f, L_\rho, L_h \in \mathbb{R}^+ : \forall (x, x', u, u', w) \in \mathcal{X}^2 \times \mathcal{U}^2 \times \mathcal{W}, \]

\[ \| f(x, u, w) - f(x', u', w) \|_\mathcal{X} \leq L_f (\| x - x' \|_\mathcal{X} + \| u - u' \|_\mathcal{U}), \]

\[ | \rho(x, u, w) - \rho(x', u', w) | \leq L_\rho (\| x - x' \|_\mathcal{X} + \| u - u' \|_\mathcal{U}), \]

\[ \forall t \in [0, T - 1], \| h(t, x) - h(t, x') \|_\mathcal{U} \leq L_h \| x - x' \|_\mathcal{X} \]
Theoretical Analysis

Assumptions

Distance metric $\Delta$

$$\forall (x, x', u, u') \in \mathcal{X}^2 \times \mathcal{U}^2,$$

$$\Delta((x, u), (x', u')) = (\|x - x'\|_x + \|u - u'\|_u)$$

$k$-dispersion

$$\alpha_k(\mathcal{P}_n) = \sup_{(x, u) \in \mathcal{X} \times \mathcal{U}} \{ \Delta_k^{\mathcal{P}_n}(x, u) \}$$

$$\Delta_k^{\mathcal{P}_n}(x, u) \quad \text{denotes the distance of } (x, u) \text{ to its } k\text{-th nearest neighbor (using the distance } \Delta) \text{ in the sample}$$

$$\mathcal{P}_n = [(x^l, u^l)]_{l=1}^n$$
The k-dispersion can be seen as the smallest radius such that all $\Delta$-balls in $X \times U$ contain at least $k$ elements from

$$\mathcal{P}_n = [(x^l, u^l)]_{l=1}^n$$
Theoretical Analysis

Theoretical results

Expected value of the MFMC estimator

\[ E_{p, P_n}^h (x_0) = \mathbb{E}_{w^1, \ldots, w^n \sim \mathcal{P}_W(.)} \left[ \mathcal{M}_p^h \left( \tilde{F}_n \left( \mathcal{P}_n, w^1, \ldots, w^n \right), x_0 \right) \right] \]
Theoretical Analysis

Theoretical results

Expected value of the MFMC estimator

\[ E_{p, \mathcal{P}_n}^h(x_0) = \mathbb{E}_{w^1, \ldots, w^n \sim p_{\mathcal{W}}(\cdot)} \left[ m_p^h \left( \tilde{F}_n (\mathcal{P}_n, w^1, \ldots, w^n), x_0 \right) \right] \]

Theorem

\[ |J^h(x_0) - E_{p, \mathcal{P}_n}^h(x_0)| \leq C \alpha_{\rho T} (\mathcal{P}_n) \]

with

\[ C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} (L_f (1 + L_h))^i \]
Theoretical Analysis

Theoretical results

Variance of the MFMC estimator

\[ V_{p,P_n}^h(x_0) = \mathbb{E}_{w^1,\ldots,w^n \sim p_W(\cdot)} \left[ \left( m_p^h \left( \tilde{F}_n \left( P_n, w^1, \ldots, w^n \right), x_0 \right) - E_{p,P_n}^h(x_0) \right)^2 \right] \]
Theoretical Analysis

Theoretical results

Variance of the MFMC estimator

\[ V_{p,P_n}^h(x_0) = \mathbb{E}_{w^1, \ldots, w^n \sim p_W(.)} \left[ \left( M_p^h \left( \tilde{F}_{n} \left( P_n, w^1, \ldots, w^n \right), x_0 \right) - E_{p,P_n}^h(x_0) \right)^2 \right] \]

**Theorem**

\[ V_{p,P_n}^h(x_0) \leq \left( \frac{\sigma_{R}^h(x_0)}{\sqrt{p}} + 2C \alpha_p T (P_n) \right)^2 \]

with

\[ C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} (L_f(1 + L_h))^i \]
Experimental Illustration

Benchmark

Dynamics:

\[ x_{t+1} = \sin \left( \frac{\pi}{2} (x_t + u_t + w_t) \right) \]

Reward function:

\[ \rho(x_t, u_t, w_t) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_t^2 + u_t^2)} + w_t \]

Policy to evaluate:

\[ h(t, x) = -\frac{x}{2}, \quad \forall x \in \mathcal{X}, \forall t \in \{0, \ldots, T - 1\} \]

\[ \mathcal{X} = [-1, 1], \mathcal{U} = [-\frac{1}{2}, \frac{1}{2}], \mathcal{W} = [-\frac{\epsilon}{2}, \frac{\epsilon}{2}] \] with \( \epsilon = 0.1 \)

Other information:

\( p_w(.) \) is uniform
Experimental Illustration

Influence of $n$

Simulations for $p = 10$, $n = 100 \ldots 10 000$, uniform grid, $T = 15$, $x_0 = -0.5$

Model-free Monte Carlo estimator

Monte Carlo estimator

$n = 100 \ldots 10 000$, $p = 10$
Simulations for $p = 1 \ldots 100$, $n = 10\,000$, uniform grid, $T = 15$, $x_0 = -0.5$

Experimental Illustration

Influence of $p$

Model-free Monte Carlo estimator

Monte Carlo estimator

$p = 1 \ldots 100$, $n=10\,000$
Experimental Illustration

MFMC vs FQI-PE

Comparison with the FQI-PE algorithm using k-NN, \( n=100, \ T=5 \).
Experimental Illustration

MFMC vs FQI-PE

Comparison with the FQI-PE algorithm using k-NN, n=100, T=5.
Research map

**Stochastic setting**
- MFMC: estimator of the expected return
- Bias / variance analysis
- Illustration

**Deterministic setting**
- Continuous action space
  - Bounds on the return
  - Convergence
- Finite action space
  - CGRL
    - Convergence + additional properties
    - Illustration
- Sampling strategy
  - Illustration
Research map

Stochastic setting
- MFMC: estimator of the expected return
- Bias / variance analysis
- Illustration
- Estimator of the VaR

Deterministic setting
- Continuous action space
  - Bounds on the return
  - Convergence
- Finite action space
  - CGRL
    - Convergence + additional properties
    - Illustration
- Sampling strategy
  - Illustration
Estimating the Performances of Policies

Risk-sensitive criterion

Consider again the \( p \) artificial trajectories that were rebuilt by the MFMC estimator. The Value-at-Risk of the policy \( h \)

\[
J_{RS}^{h,(b,c)}(x_0) = \begin{cases} 
-\infty & \text{if } P \left( R^h(x_0, w_0, \ldots, w_{T-1}) < b \right) > c \\
J^h(x_0) & \text{otherwise}
\end{cases}
\]

can be straightforwardly estimated as follows:

\[
\tilde{J}_{RS}^{h,(b,c)}(x_0) = \begin{cases} 
-\infty & \text{if } \frac{1}{p} \sum_{i=1}^{p} \mathbb{I}\{r^i < b\} > c \\
M^h(\mathcal{F}_n, x_0) & \text{otherwise}
\end{cases}
\]

with \( r^i = \sum_{t=0}^{T-1} r^{i,t} \)

\( c \in [0, 1] \quad b \in \mathbb{R} \)
Deterministic Case: Computing Bounds

Bounds from a Single Trajectory

Given an artificial trajectory: $\tau = \left[ (x^l_t, u^l_t, r^l_t, y^l_t) \right]_{t=0}^{T-1}$

\[ x_0 = \rho(x_0, h(0, x_0)), \quad x_1 = f(x_0, h(0, x_0)) \]

\[ x_0 = (x^{l_0}, u^{l_0}, r^{l_0}, y^{l_0}), \quad x_1 = (x^{l_1}, u^{l_1}, r^{l_1}, y^{l_1}) \]

$\delta_0 = \| x^{l_0} - x_0 \|_x + \| u^{l_0} - h(0, x_0) \|_u,$

$\delta_1 = \| y^{l_0} - x^{l_1} \|_x + \| u^{l_1} - h(1, y^{l_0}) \|_u,$

\[ x_{T-1}, x_T \]

$\delta_{T-1} = \| x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, y^{l_{T-1}} \|_x$
Deterministic Case: Computing Bounds

Bounds from a Single Trajectory

Proposition:

Let $\left[ \left( x_{t}^{l_{t}}, u_{t}^{l_{t}}, r_{t}^{l_{t}}, y_{t}^{l_{t}} \right) \right]_{t=0}^{T-1}$ be an artificial trajectory. Then,

$$J^{h}(x_{0}) \geq \sum_{t=0}^{T-1} r_{t}^{l_{t}} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta \left( (y_{t}^{l_{t-1}}, h(t, y_{t}^{l_{t-1}})), (x_{t}^{l_{t}}, u_{t}^{l_{t}}) \right)$$

with

$$L_{Q_{T-t}} = L_{\rho} \sum_{i=0}^{T-t-1} \left( L_{f} (1 + L_{h}) \right)^{i}$$

$$y_{t-1}^{l_{t-1}} = x_{0}$$
Deterministic Case: Computing Bounds

Maximal Bounds

Maximal lower and upper-bounds

\[
L^h(\mathcal{F}_n, x_0) = \max_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta ((y_{t-1}^{l_t}, h(t, y_{t-1}^{l_t})), (x_t^{l_t}, u_t^{l_t}))
\]

\[
U^h(\mathcal{F}_n, x_0) = \min_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} + \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta ((y_{t-1}^{l_t}, h(t, y_{t-1}^{l_t})), (x_t^{l_t}, u_t^{l_t}))
\]
Proposition:

\[ \exists C_b > 0 : \quad J^h(x_0) - L^h(F_n, x_0) \leq C_b \alpha_1(\mathcal{P}_n) \]
\[ U^h(F_n, x_0) - J^h(x_0) \leq C_b \alpha_1(\mathcal{P}_n) \]
Consider the set of open-loop policies:

$$\Pi = \{\pi : \{0, \ldots, T - 1\} \rightarrow \mathcal{U}\}$$

For such policies, bounds can be computed in a similar way.

We can then search for a specific policy for which the associated lower bound is maximized:

$$\hat{\pi}^*_{F_n, x_0} \in \arg\max_{\pi \in \Pi} L^\pi (F_n, x_0)$$

A $O(T n^2)$ algorithm for doing this: the CGRL algorithm (Cautious approach to Generalization in RL)
Inferring Safe Policies

Convergence

**Theorem**

Let $\mathcal{J}^*(x_0)$ be the set of optimal open-loop policies:

$$
\mathcal{J}^*(x_0) = \arg \max_{\pi \in \Pi} J^\pi(x_0),
$$

and let us suppose that $\mathcal{J}^*(x_0) \neq \Pi$ (if $\mathcal{J}^*(x_0) = \Pi$, the search for an optimal policy is indeed trivial). We define

$$
\epsilon(x_0) = \min_{\pi \in \Pi \setminus \mathcal{J}^*(x_0)} \left\{ \left( \max_{\pi' \in \Pi} J^{\pi'}(x_0) \right) - J^\pi(x_0) \right\}.
$$

Then,

$$
\left( C_b \alpha^*(\mathcal{P}_n) < \epsilon(x_0) \right) \implies \hat{\pi}_{\mathcal{F}_n, x_0}^* \in \mathcal{J}^*(x_0).
$$
Inferring Safe Policies

Experimental Results

- The puddle world benchmark
<table>
<thead>
<tr>
<th></th>
<th>CGRL</th>
<th>FQI (Fitted Q Iteration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The state space is uniformly covered by the sample</td>
<td><img src="image1.png" alt="Image of CGRL" /></td>
<td><img src="image2.png" alt="Image of FQI" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Initial state" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information about the Puddle area is removed</td>
<td><img src="image4.png" alt="Image of CGRL" /></td>
<td><img src="image5.png" alt="Image of FQI" /></td>
</tr>
</tbody>
</table>
Inferring Safe Policies

**Theorem**

Let $\pi^*_0 \in \mathcal{J}^*(x_0)$ be an optimal open-loop policy. Let us assume that one can find in $\mathcal{F}_n$ a sequence of $T$ one-step system transitions

$$[ (x^{l_0}, u^{l_0}, r^{l_0}, x^{l_1}), (x^{l_1}, u^{l_1}, r^{l_1}, x^{l_2}), \ldots, (x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, x^{l_T}) ] \in \mathcal{F}_n^T$$

such that

$$x^{l_0} = x_0,$$

$$u^{l_t} = \pi^*_0(t) \quad \forall t \in \{0, \ldots, T-1\}.$$

Let $\hat{\pi}^*_{\mathcal{F}_n, x_0}$ be such that

$$\hat{\pi}^*_{\mathcal{F}_n, x_0} \in \arg \max_{\pi \in \Pi} L^\pi(\mathcal{F}_n, x_0).$$

Then,

$$\hat{\pi}^*_{\mathcal{F}_n, x_0} \in \mathcal{J}^*(x_0).$$
Sampling Strategies

An Artificial Trajectories Viewpoint

Given a sample of system transitions

\[ F_n = \left\{ (x^l, u^l, r^l, y^l) \in X \times U \times \mathbb{R} \times X \right\}_{l=1}^{n} \]

How can we determine where to sample additional transitions?

We define the set of candidate optimal policies:

\[ \Pi(F, x_0) = \left\{ \pi \in \Pi \mid \forall \pi' \in \Pi, U^{\pi}(F, x_0) \geq L^{\pi'}(F, x_0) \right\} \]

A transition \((x, u, r, y) \in X \times U \times \mathbb{R} \times X\) is compatible with \(F\) if

\[ \forall (x^l, u^l, r^l, y^l) \in F, \quad (u^l = u) \quad \Rightarrow \quad \begin{cases} |r - r^l| \leq L^{\rho} \|x - x^l\|_x \\
\|y - y^l\|_x \leq L^{f} \|x - x^l\|_x \end{cases} \]

and we denote by \(C(F)\) the set of all such compatible transitions.
Sampling Strategies

An Artificial Trajectories Viewpoint

Iterative scheme:

\[
(x^{m+1}, u^{m+1}) \in \arg \min_{(x,u) \in \mathcal{X} \times \mathcal{U}} \left\{ \max_{(r,y) \in \mathbb{R} \times \mathcal{X}} \ s.t. (x,u,r,y) \in \mathcal{C}(\mathcal{F}_m), \right. \\
\left. \pi \in \Pi(\mathcal{F}_m \cup \{(x,u,r,y)\}, x_0) \right\}
\]

with

\[
\delta^\pi(\mathcal{F}, x_0) = U^\pi(\mathcal{F}, x_0) - L^\pi(\mathcal{F}, x_0)
\]

Conjecture:

\[
\exists m_0 \in \mathbb{N} \setminus \{0\} : \forall m \in \mathbb{N}, \left( m \geq m_0 \right) \implies \Pi(\mathcal{F}_m, x_0) = \mathcal{Z}^*(x_0)
\]
Sampling Strategies

Illustration

Action space: \[ \mathcal{U} = \{-0.20, -0.10, 0, +0.10, +0.20\} \]

Dynamics and reward function:
\[ f(x, u) = x + u \]
\[ \rho(x, u) = x + u \]

Horizon: \[ T = 3 \]

Initial state:
\[ x_0 = -0.65 \]

Total number of policies:
\[ 5^3 = 125 \]

Number of transitions needed for discriminating:
\[ 5 + 25 + 125 = 155 \]
Thank you


