Batch mode reinforcement learning based on the synthesis of artificial trajectories

Damien Ernst

University of Liège, Belgium

Princeton - December 2012

Batch mode Reinforcement Learning

Learning a high-performance policy for a sequential decision problem where:

- a numerical criterion is used to define the performance of a policy. (An optimal policy is the policy that maximizes this numerical criterion.)
- “the only” (or “most of the”) information available on the sequential decision problem is contained in a set of trajectories.

Batch mode RL stands at the intersection of three worlds: **optimization** (maximization of the numerical criterion), **system/control theory** (sequential decision problem) and **machine learning** (inference from a set of trajectories).

A typical batch mode RL problem

Discrete-time dynamics:
$x_{t+1} = f(x_t, u_t, w_t) \quad t = 0, 1, \ldots, T-1$ where $x_t \in X$, $u_t \in U$ and $w_t \in W$. w_t is drawn at every time step according to $P_w(\cdot)$.

Reward observed after each system transition:
$r_t = \rho(x_t, u_t, w_t)$ where $\rho : X \times U \times W \to \mathbb{R}$ is the reward function.

Type of policies considered: $h : \{0, 1, \ldots, T-1\} \times X \to U$.

Performance criterion: Expected sum of the rewards observed over the T-length horizon
$PC^h(x) = J^h(x) = \underset{w_0, \ldots, w_{T-1}}{E}[\sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t)]$ with $x_0 = x$ and $x_{t+1} = f(x_t, h(t, x_t), w_t)$.

Available information: A set of elementary pieces of trajectories $\mathcal{F}_n = \{(x^l, u^l, r^l, y^l)\}_{l=1}^n$ where y^l is the state reached after taking action u^l in state x^l and r^l the instantaneous reward associated with the transition. The functions f, ρ and P_w are unknown.

Batch mode RL and function approximators

Training function approximators (radial basis functions, neural nets, trees, etc) using the information contained in the set of trajectories is a key element to most of the resolution schemes for batch mode RL problems with state-action spaces having a large (infinite) number of elements.

Two typical uses of FAs for batch mode RL:

- the FAs **model of the sequential decision problem** (in our typical problem f, r and P_w). The model is afterwards exploited as if it was the real problem to compute a high-performance policy.
- the FAs represent (state-action) **value functions** which are used in iterative schemes so as to converge to a (state-action) value function from which a high-performance policy can be computed. Iterative schemes based on the dynamic programming principle (e.g., LSPI, FQI, Q-learning).

Why look beyond function approximators?

FAs based techniques: mature, can successfully solve many real life problems but:

1. not well adapted to risk sensitive performance criteria
2. may lead to unsafe policies - poor performance guarantees
3. may make suboptimal use of near-optimal trajectories
4. offer little clues about how to generate new experiments in an optimal way

1. not well adapted to risk sensitive performance criteria

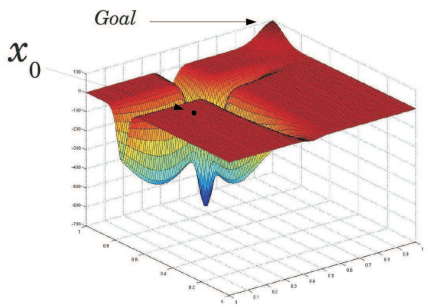
An example of risk sensitive performance criterion:

$$PC^h(x) = \begin{cases} -\infty & \text{if} \quad P(\sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t) < b) > c \\ J^h(x) & \text{otherwise.} \end{cases}$$

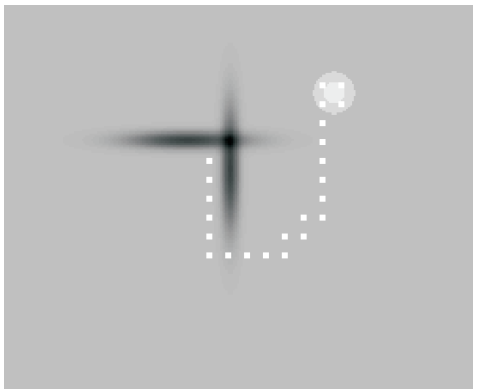
FAs with dynamic programming: very problematic because (state-action) value functions need to become functions that take as values "probability distributions of future rewards" and not "expected rewards".
FAs with model learning: more likely to succeed; but what about the challenges of fitting the FAs to model the distribution of future states reached (rewards collected) by policies and not only an average behavior?

2. may lead to unsafe policies - poor performance guarantees

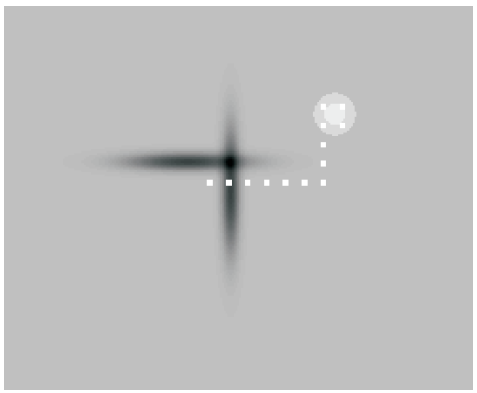
- Benchmark: puddle world
- RL algorithms: FQI with trees



- Trajectory set covering the puddle $\Rightarrow$ Optimal policy



- Trajectory set not covering the puddle $\Rightarrow$ Suboptimal (unsafe) policy



Typical performance guarantee in the deterministic case for FQI = (estimated return by FQI of the policy it outputs minus constant$\times$('size' of the largest area of the state space not covered by the sample)).

3. may make suboptimal use of near-optimal trajectories

Suppose a deterministic batch mode RL problem and that in the set of trajectory, there is a trajectory:
$(x_0^{opt.\ traj.}, u_0, r_0, x_1, u_1, r_1, x_2, \ldots, x_{T-2}, u_{T-2}, r_{T-2}, x_{T-1}, u_{T-1}, r_{T-1}, x_T)$
where the u_ts have been selected by an optimal policy.
Question: Which batch mode RL algorithms will output a policy which is optimal for the initial state $x_0^{opt.\ traj.}$ whatever the other trajectories in the set? **Answer:** Not that many and certainly not those using parametric FAs.

In my opinion: batch mode RL algorithms can only be successful on large-scale problems if (i) in the set of trajectories, many trajectories have been generated by (near-)optimal policies (ii) the algorithms exploit very well the information contained in those (near-)optimal trajectories.

4. offer little clues about how to generate new experiments in an optimal way

---

Many real-life problems are variants of batch mode RL problems for which (a limited number of) additional trajectories can be generated (under various constraints) to enrich the initial set of trajectories.

Question: How should these new trajectories be generated? Many approaches based on the analysis of the FAs produced by batch mode RL methods have been proposed; results are mixed.

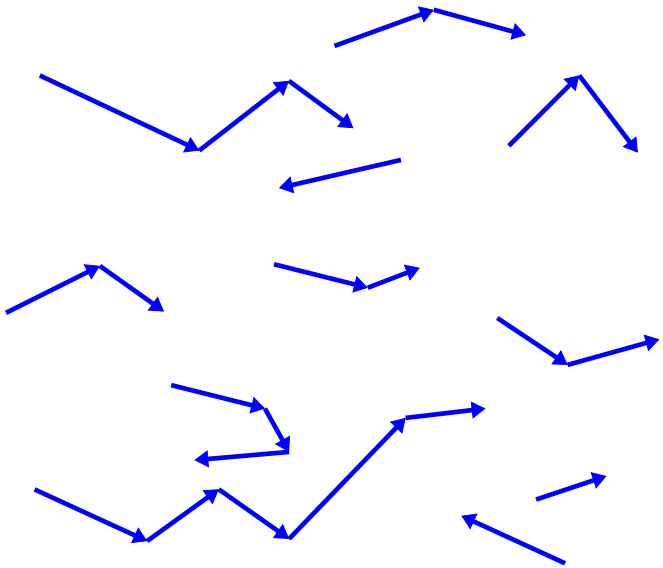
Rebuilding trajectories

We conjecture that mapping the set of trajectories into FAs generally lead to the loss of essential information for addressing these four issues $\Rightarrow$ We have developed a new line of research for solving batch mode RL that does not use at all FAs.

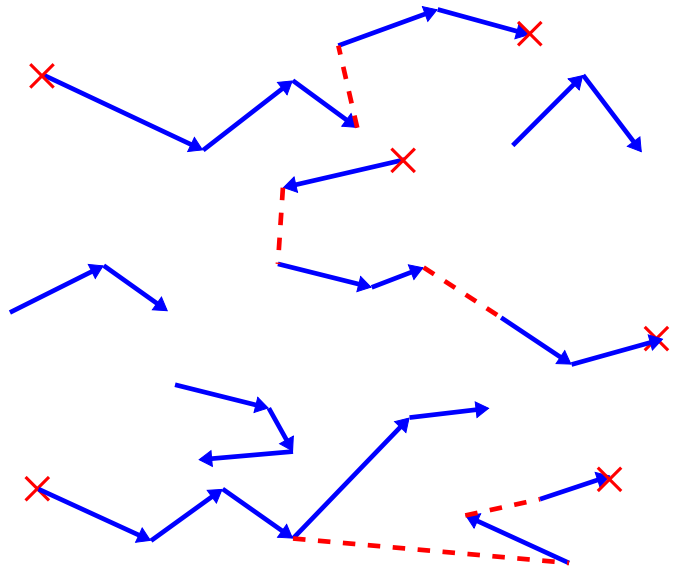
Line of research articulated around the the rebuilding of artificial (likely "broken") trajectories by using the set of trajectories input of the batch mode RL problem; a rebuilt trajectory is defined by the elementary pieces of trajectory it is made of.

The rebuilt trajectories are analysed to compute various things: a high-performance policy, performance guarantees, where to sample, etc.

BLUE ARROW = elementary piece of trajectory



Set of trajectories given as input of the batch RL problem



Examples of 5-length rebuilt trajectories made from elements of this set

Model-Free Monte Carlo Estimator

Building an oracle that estimates the performance of a policy: important problem in batch mode RL.

Indeed, if an oracle is available, problem of estimating a high-performance policy can be reduced to an optimization problem over a set of candidate policies.

If a model of sequential decision problem is available, a Monte Carlo estimator (i.e., rollouts) can be used to estimate the performance of a policy.

We detail an approach that estimates the performance of a policy by rebuilding trajectories so as to mimic the behavior of the Monte Carlo estimator.

Context in which the approach is presented

Discrete-time dynamics:
$x_{t+1} = f(x_t, u_t, w_t) \quad t = 0, 1, \ldots, T-1$ where $x_t \in X$, $u_t \in U$ and $w_t \in W$. w_t is drawn at every time step according to $P_w(\cdot)$

Reward observed after each system transition:
$r_t = \rho(x_t, u_t, w_t)$ where $\rho : X \times U \times W \to \mathbb{R}$ is the reward function.

Type of policies considered: $h : \{0, 1, \ldots, T-1\} \times X \to U$.

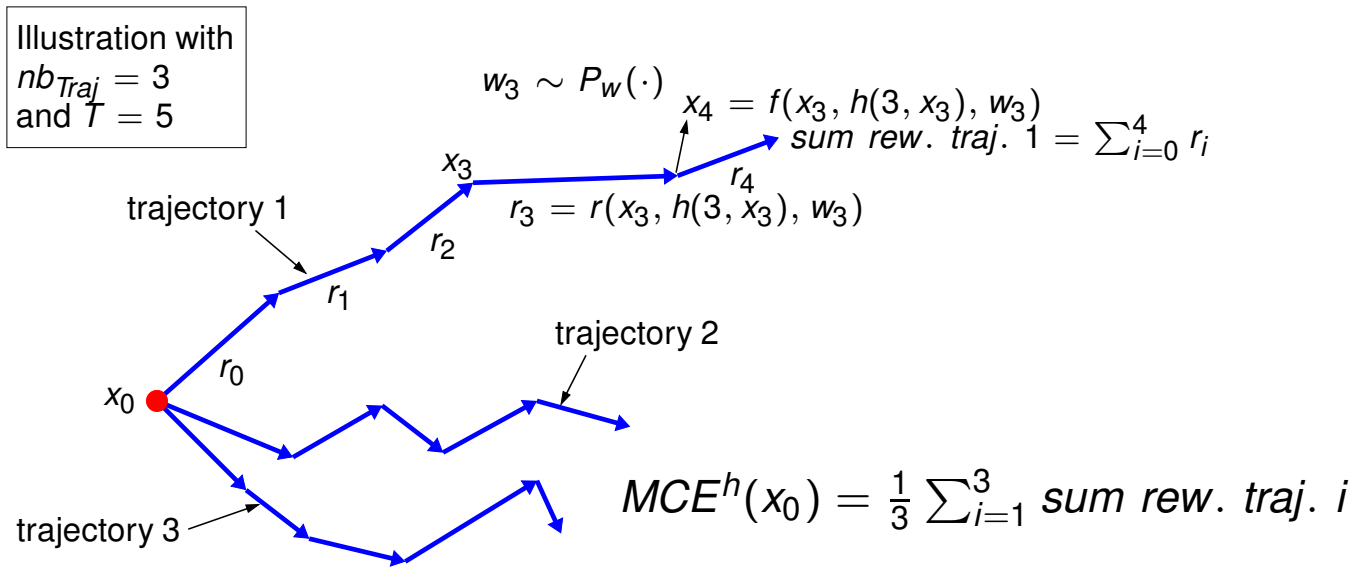
Performance criterion: Expected sum of the rewards observed over the T-length horizon
$PC^h(x) = J^h(x) = \underset{w_0, \ldots, w_{T-1}}{E}[\sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t)]$ with $x_0 = x$ and $x_{t+1} = f(x_t, h(t, x_t), w_t)$.

Available information: A set of elementary pieces of trajectories $\mathcal{F}_n = \{(x^l, u^l, r^l, y^l)\}_{l=1}^n$. f, ρ and P_w are unknown.

Approach aimed at estimating $J^h(x)$ from $\mathcal{F}_n$.

Monte Carlo Estimator

Generate nb_{Traj} T-length trajectories by simulating the system starting from the initial state x_0; for every trajectory compute the sum of rewards collected; average these sum of rewards over the nb_{Traj} trajectories to get an estimate $MCE^h(x_0)$ of $J^h(x_0)$.



Bias $MCE^h(x_0) = \underset{nb_{Traj} * T\ rand.\ var.\ w \sim P_w(\cdot)}{E}[MCE^h(x_0) - J^h(x_0)] = \mathbf{0}$

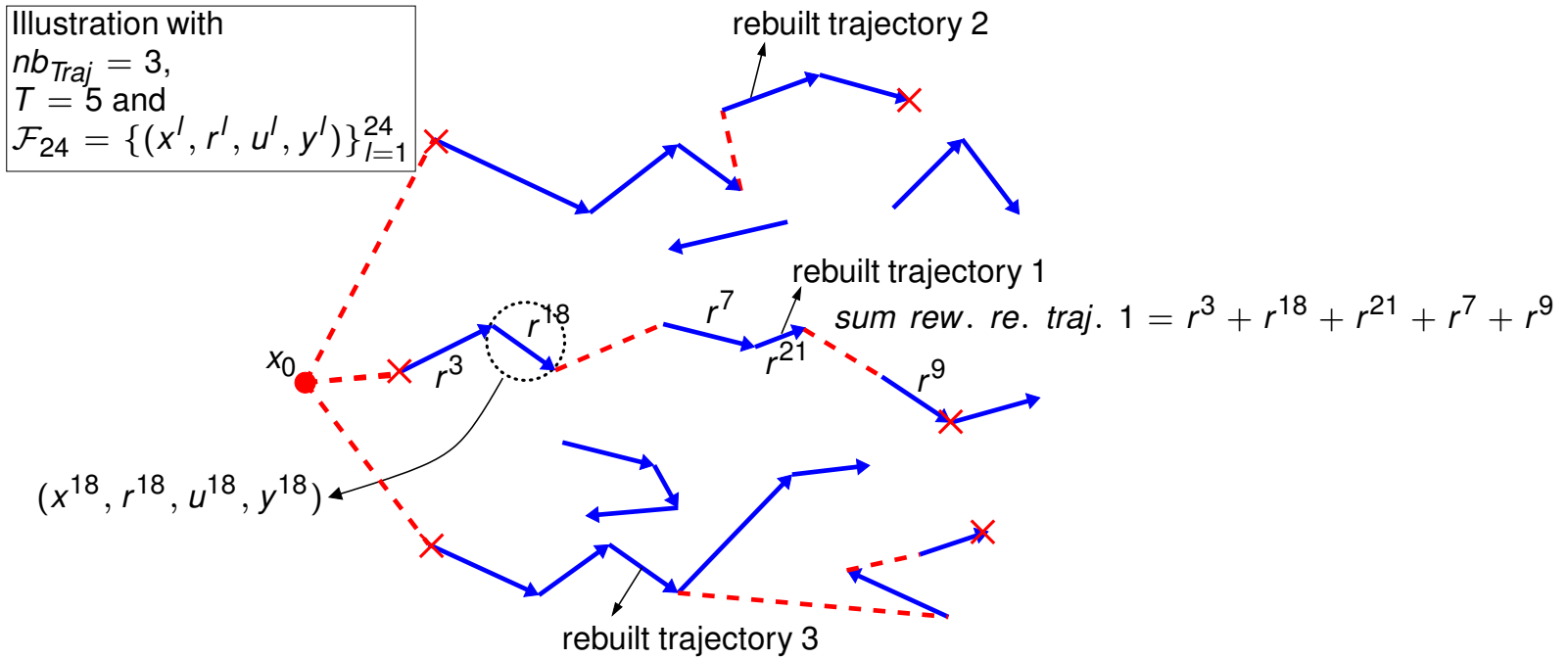
Var. $MCE^h(x_0) = \frac{1}{nb_{Traj}}$(Var. of the sum of rewards along a traj.)

Description of Model-free Monte Carlo Estimator (MFMC)

Principle: To rebuild nb_{Traj} T-length trajectories using the elements of the set $\mathcal{F}_n$and to average the sum of rewards collected along the rebuilt trajectories to get an estimate $MFMCE^h(\mathcal{F}_n, x_0)$ of $J^h(x_0)$.

Trajectories rebuilding algorithm: Trajectories are sequentially rebuilt; an elementary piece of trajectory can only be used once; trajectories are grown in length by selecting at every instant $t = 0, 1, \ldots, T-1$ the elementary piece of trajectory (x, u, r, y) that minimizes the distance $\Delta((x, u), (x^{end}, h(t, x^{end})))$ where x^{end} is the ending state of the already rebuilt part of the trajectory ($x^{end} = x_0$ if $t = 0$).

Remark: When sequentially selecting the pieces of trajectories, no information on the value of the disturbance w "behind" the new piece of elementary trajectory $(x, u, r = \rho(x, u, w), y = f(x, u, w))$ that is going to be selected is given if only (x, u) and the previous elementary pieces of trajectories selected are known. Important for having a meaningful estimator !!!



Analysis of the MFMC

Random set $\tilde{\mathcal{F}}_n$ defined as follows:
Made of n elementary pieces of trajectory where the first two components of an element (x^l, u^l) are given by the first two element of the lth element of $\mathcal{F}_n$ and the last two are generated by drawing for each l a disturbance signal w^l at random from $P_W(\cdot)$ and taking $r^l = \rho(x^l, u^l, w^l)$ and $y^l = f(x^l, u^l, w^l)$.
$\mathcal{F}_n$ is a **realization of the random set** $\tilde{\mathcal{F}}_n$.

Bias and variance of MFMCE defined as:

$$Bias\ MFMCE^h(\tilde{\mathcal{F}}_n, x_0) = \underset{w^1,\ldots,w^n \sim P_w}{E}[MFMCE^h(\tilde{\mathcal{F}}_n, x_0) - J^h(x_0)]$$

$$Var.\ MFMCE^h(\tilde{\mathcal{F}}_n, x_0) = \underset{w^1,\ldots,w^n \sim P_w}{E}[(MFMCE^h(\tilde{\mathcal{F}}_n, x_0) - \underset{w^1,\ldots,w^n \sim P_w}{E}[MFMCE^h(\tilde{\mathcal{F}}_n, x_0)])^2]$$

We provide **bounds of the bias and variance** of this estimator.

Assumptions

1] The functions f, ρ and h are Lipschitz continuous:
$\exists L_f, L_\rho, L_h \in R^+ : \forall (x, x', u, u', w) \in X^2 \times U^2 \times W$;
$\|f(x, u, w) - f(x', u', w)\|_X \leq L_f(\|x - x'\|_X + \|u - u'\|_U)$
$|\rho(x, u, w) - \rho(x', u', w)| \leq L_\rho(\|x - x'\|_X + \|u - u'\|_U)$
$\|h(t, x) - h(t, x')\|_U \leq L_h\|x - x'\| \ \forall t \in \{0, 1, \ldots, T - 1\}$.

2] The distance Δ is chosen such that:
$\Delta((x, u), (x', u')) = (\|x - x'\|_X + \|u - u'\|_U)$.

Characterization of the bias and the variance

Theorem.

$Bias\ MFMCE^h(\tilde{\mathcal{F}}_n, x_0) \leq C * sparsity\_of\_\mathcal{F}_n(nb_{Traj} * T)$

$Var.\ MFMCE^h(\tilde{\mathcal{F}}_n, x_0) \leq$
$(\sqrt{Var.\ MCE^h(x_0)} + 2C * sparsity\_of\_\mathcal{F}_n(nb_{Traj} * T))^2$

with $C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} [L_f(1 + L_h)]^i$ and with the $sparsity\_of\_\mathcal{F}_n(k)$ defined as the minimal radius r such that all balls in $X \times U$ of radius r contain at least k state-action pairs (x^l, u^l) of the set $\mathcal{F}_n = \{(x^l, u^l, r^l, y^l)\}_{l=1}^n$.

Test system

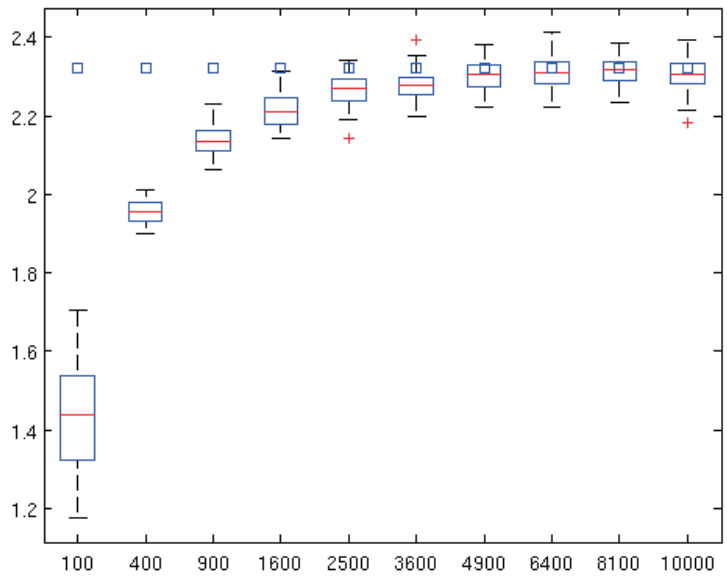
Discrete-time dynamics: $x_{t+1} = \sin(\frac{\pi}{2}(x_t + u_t + w_t))$ with $X = [-1, 1]$, $U = [-\frac{1}{2}, \frac{1}{2}]$, $W = [-\frac{0.1}{2}, \frac{-0.1}{2}]$ and $P_w(\cdot)$ a uniform pdf.

Reward observed after each system transition: $r_t = \frac{1}{2\pi} e^{-\frac{1}{2}(x_t^2 + u_t^2)} + w_t$

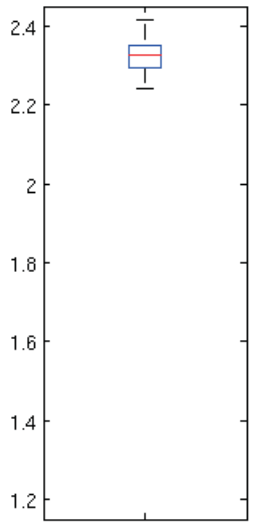
Performance criterion: Expected sum of the rewards observed over a 15-length horizon ($T = 15$).

We want to evaluate the performance of the policy $h(t, x) = -\frac{x}{2}$ when $x_0 = -0.5$.

Simulations for $nb_{Traj} = 10$ and size of $\mathcal{F}_n = 100, ..., 10000$.

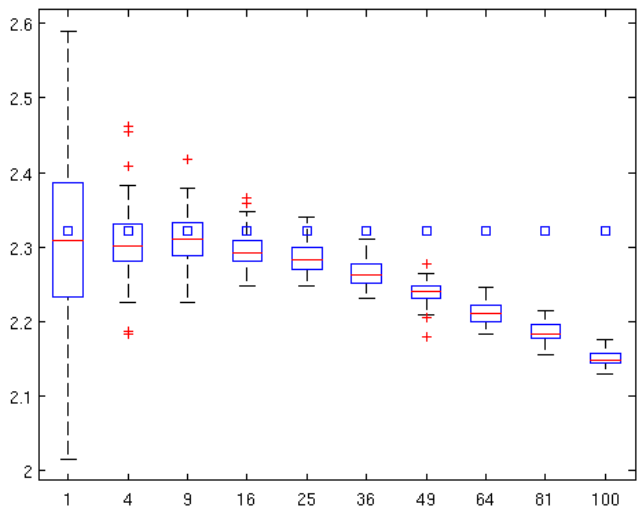


Model-free Monte Carlo Estimator

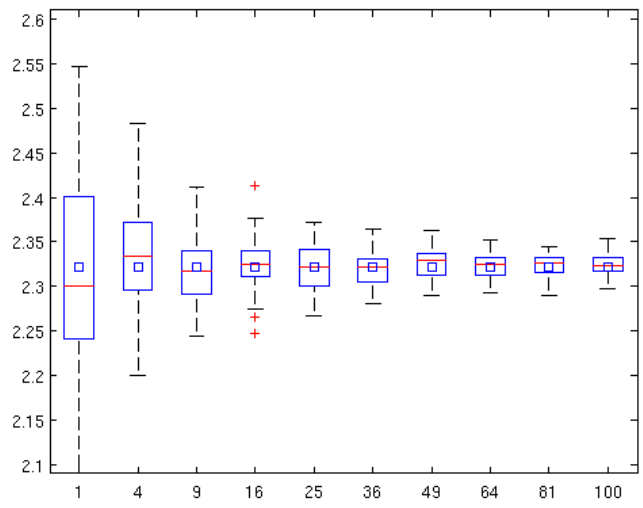


Monte Carlo Estimator

Simulations for $nb_{Traj} = 1, \ldots, 100$ and size of $\mathcal{F}_n = 10,000$.



Model-free Monte Carlo Estimator



Monte Carlo Estimator

Remember what was said about RL + FAs:

1. not well adapted to risk sensitive performance criteria

Suppose the risk sensitive performance criterion:

$$PC^h(x) = \begin{cases} -\infty & \text{if} \quad P(J^h(x) < b) > c \\ J^h(x) & \text{otherwise} \end{cases}$$

where $J^h(x) = E[\sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t)]$.

MFMCE adapted to this performance criterion:

Rebuilt nb_{Traj} starting from x_0 using the set $\mathcal{F}_n$ as done with the MFMCE estimator. Let *sum_rew_traj_i* be the sum of rewards collected along the ith trajectory. Output as estimation of $PC^h(x_0)$:

$$\begin{cases} -\infty & \text{if} \quad \frac{\sum_{i=1}^{nb_{Traj}} I_{\{sum\_rew\_traj\_i < b\}}}{nb_{Traj}} > c \\ \sum_{i=1}^{nb_{Traj}} \frac{sum\_rew\_traj\_i}{nb_{Traj}} & \text{otherwise.} \end{cases}$$

MFMCE in the deterministic case

We consider from now on that: $x_{t+1} = f(x_t, u_t)$ and $r_t = \rho(x_t, u_t)$.

One single trajectory is sufficient to compute exactly $J^h(x_0)$ by Monte Carlo estimation.

Theorem. Let $[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}$ be the trajectory rebuilt by the MFMCE when using the distance measure $\Delta((x,u),(x',u')) = \|x - x'\| + \|u - u'\|$. If f, ρ and h are Lipschitz continuous, we have

$$|MFMCE^h(x_0) - J^h(x_0)| \leq \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

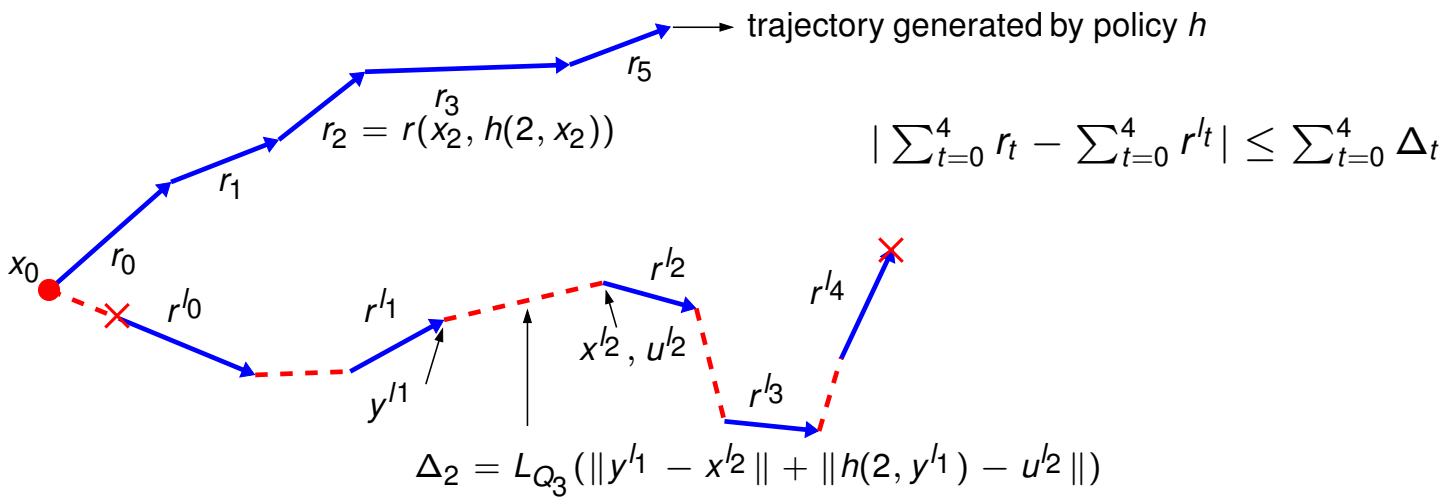
where $y^{l_{-1}} = x_0$ and $L_{Q_N} = L_\rho(\sum_{t=0}^{N-1}[L_f(1 + L_h)]^t)$.

Previous theorem extends to whatever rebuilt trajectory:

Theorem. Let $[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}$ be any rebuilt trajectory. If f, ρ and h are Lipschitz continuous, we have

$$|\sum_{t=0}^{T-1} r_{l_t} - J^h(x_0)| \leq \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

where $\Delta((x,u),(x',u')) = \|x - x'\| + \|u - u'\|$, $y^{l_{-1}} = x_0$ and $L_{Q_N} = L_\rho(\sum_{t=0}^{N-1}[L_f(1+L_h)]^t)$.



Computing a lower bound on a policy

From previous theorem, we have for any rebuilt trajectory $[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}$:

$$J^h(x_0) \geq \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

This suggests to find the rebuilt trajectory that maximizes the right-hand side of the inequality to compute a tight lower bound on h. Let:

$$\mathit{lower\_bound}(h, x_0, \mathcal{F}_n) \triangleq \max_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}} \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

A tight upper bound on $J^h(x)$ can be defined and computed in a similar way:

$$\mathit{upper\_bound}(h, x_0, \mathcal{F}_n) \triangleq$$
$$\min_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}} \sum_{t=0}^{T-1} r^{l_t} + \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

Why are these bounds tight? Because:

$$\exists C \in \mathbb{R}^+ : J^h(x) - \mathit{lower\_bound}(h, x, \mathcal{F}_n) \leq C * \mathit{sparsity\_of\_}\mathcal{F}_n(1)$$
$$\mathit{upper\_bound}(h, x, \mathcal{F}_n) - J^h(x) \leq C * \mathit{sparsity\_of\_}\mathcal{F}_n(1)$$

Functions $\mathit{lower\_bound}(h, x_0, \mathcal{F}_n)$ and $\mathit{higher\_bound}(h, x_0, \mathcal{F}_n)$ can be implemented in a "smart way" by seeing the problem as a problem of finding the shortest path in a graph. Complexity **linear with** T and **quadratic with** $|\mathcal{F}_n|$.

Remember what was said about RL + FAs:

2. may lead to unsafe policies - poor performance guarantees

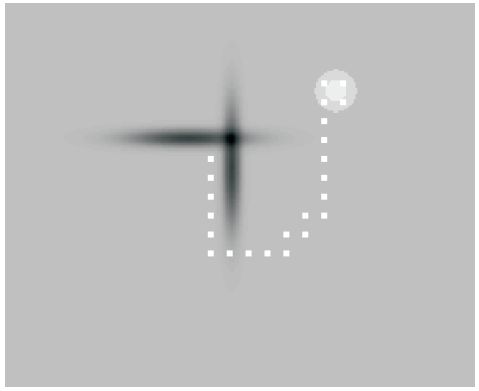
Let $\mathcal{H}$ be a set of candidate high-performance policies. To obtain a policy with good performance guarantees, we suggest to solve the following problem:

$$h \in \arg\max_{h \in \mathcal{H}} \; \mathit{lower\_bound}(h, x_0, \mathcal{F}_n)$$

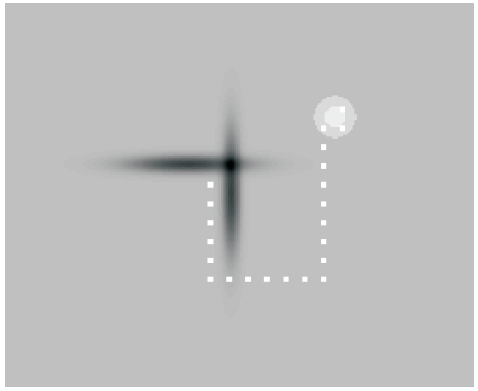
If $\mathcal{H}$ is the set of open-loop policies, solving the above optimization problem can be seen as identifying an "optimal" rebuilt trajectory and outputting as open-loop policy the sequence of actions taken along this rebuilt trajectory.

- Trajectory set covering the puddle:

FQI with trees

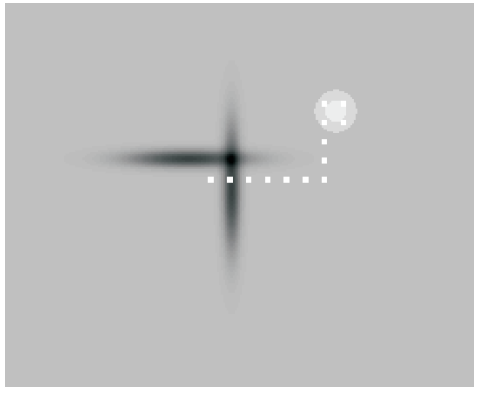


$h \in \arg\max_{h \in \mathcal{H}} \mathit{lower\_bound}(h, x_0, \mathcal{F}_n)$

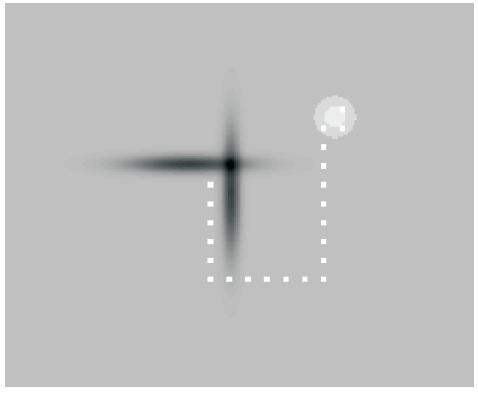


- Trajectory set not covering the puddle:

FQI with trees



$h \in \arg\max_{h \in \mathcal{H}} \mathit{lower\_bound}(h, x_0, \mathcal{F}_n)$



Remember what was said about RL + FAs:

3. may make suboptimal use of near-optimal trajectories

Suppose a deterministic batch mode RL problem and that in $\mathcal{F}_n$ you have the elements of the trajectory:
$(x_0^{opt.\ traj.}, u_0, r_0, x_1, u_1, r_1, x_2, \ldots, x_{T-2}, u_{T-2}, r_{T-2}, x_{T-1}, u_{T-1}, r_{T-1}, x_T)$
where the u_ts have been selected by an optimal policy.
Let $\mathcal{H}$ be the set of open-loop policies. Then, the sequence of actions $h \in \arg\max_{h \in \mathcal{H}} \mathit{lower\_bound}(h, x_0^{opt.\ traj.}, \mathcal{F}_n)$ is an optimal one whatever the other trajectories in the set.

Actually, the sequence of action h outputted by this algorithm tends to be an **append of subsequences of actions belonging to optimal trajectories**.

Remember what was said about RL + FAs:

4. offer little clues about how to generate new experiments in an optimal way

---

The functions $lower\_bound(h, x_0, \mathcal{F}_n)$ and $upper\_bound(h, x_0, \mathcal{F}_n)$ can be exploited for generating new trajectories.

For example, suppose that you can sample the state-action space several times so as to generate m new elementary pieces of trajectories to enrich your initial set $\mathcal{F}_n$. We have proposed a technique to determine m **"interesting"** sampling locations based on these bounds.

This technique - which is still preliminary - targets sampling locations that lead to the largest bound width decrease for candidate optimal policies.

Closure

Rebuilding trajectories: interesting concept for solving many problems related to batch mode RL.

Actually, the solution outputted by many RL algorithms (e.g., model-learning with kNN, fitted Q iteration with trees) can be characterized by a set of "rebuilt trajectories".

$\Rightarrow$ I suspect that this concept of rebuilt trajectories could lead to a **general paradigm** for analyzing and designing RL algorithms.

Presentation based on the paper:

Batch mode reinforcement learning based on the synthesis of artificial trajectories. R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. Annals of Operations Research, Volume 208, Issue 1, September 2013, Pages 383-416.

A picture of the authors: