Computing near-optimal policies from trajectories by solving a sequence of standard supervised learning problems

Damien Ernst

Hybrid Systems Control Group - Supélec

TU Delft, November 16, 2006

From trajectories to optimal policies: control community versus computer science community

Problem of inference of (near) optimal policies from trajectories has been studied by the control community and the computer science community.

Control community favors a two-stage approach: identification of an analytical model and derivation from it of an optimal policy.

Computer science community favors approaches which compute optimal policies directly from the trajectories, without relying on the identification of an analytical model.

Learning optimal policies from trajectories is known in the computer science community as reinforcement learning.

Reinforcement learning

Classical approach: to infer from trajectories state-action value functions.

Discrete (and not to large) state and action spaces: state-action value functions can be represented in tabular form.

Continuous or large state and action spaces: function approximators need to be used. Also, learning must be done from finite and generally very sparse sets of trajectories.

Parametric function approximators with gradient descent like methods: very popular techniques but do not generalize well enough to move from the academic to the real-world !!!

Using supervised learning to solve the generalization problem in reinforcement learning

Problem of generalization over an information space. Occurs also in supervised learning (SL).

What is SL ? To infer from a sample of input-output pairs (input = information state; output = class label or real number) a model which explains "at best" these input-output pairs.

Supervised learning highly successful: state-of-the art SL algorithms have been successfully applied to problems where information state = thousands of components.

What we want: we want to use the generalization capabilities of supervised learning in reinforcement learning.

Our answer: an algorithm named fitted Q iteration which infers (near) optimal policies from trajectories by solving a sequence of standard supervised learning problems.

Problem formulation

Deterministic version

Discrete-time dynamics: $x_{t+1} = f(x_t, u_t) \quad t = 0, 1, \ldots$ where $x_t \in X$ and $u_t \in U$.

Cost function: $c(x, u) : X \times U \rightarrow \mathbf{R}$. $c(x, u)$ bounded by B_c.
Instantaneous cost: $c_t = c(x_t, u_t)$

Discounted infinite horizon cost associated to stationary policy $\mu : X \rightarrow U$: $J^{\mu}(x) = \lim_{N \rightarrow \infty} \sum_{t=0}^{N-1} \gamma^t c(x_t, \mu(x_t))$ where $\gamma \in [0, 1[$.

Optimal stationary policy μ^* : Policy that minimizes J^{μ} for all x.

Objective: Find an optimal policy μ^*.

We do not know: The discrete-time dynamics and the cost function.

We know instead: A set of trajectories $\{(x_0, u_0, c_0, x_1, \cdots, u_{T-1}, c_{T-1}, x_T)^i\}_{i=1}^{nbTraj}$.

Some dynamic programming results

Sequence of state-action value functions $Q_N: X \times U \rightarrow \mathbb{R}$

$$Q_N(x,u) = c(x,u) + \gamma \min_{u' \in U} Q_{N-1}(f(x,u),u'), \quad \forall N > 1$$

with $Q_1(x,u) \equiv c(x,u)$, converges to the Q-function, unique solution of the Bellman equation:

$$Q(x,u) = c(x,u) + \gamma \min_{u' \in U} Q(f(x,u),u').$$

Necessary and sufficient optimality condition:

$$\mu^*(x) \in \arg\min_{u \in U} Q(x,u)$$

Suboptimal stationary policy μ_N^*:

$$\mu_N^*(x) \in \arg\min_{u \in U} Q_N(x,u).$$

Bound on μ_N^*:

$$J^{\mu_N^*} - J^{\mu^*} \leq \frac{2\gamma^N B_c}{(1-\gamma)^2}.$$

Fitted Q iteration: the algorithm

Set of trajectories $\{(x_0, u_0, c_0, x_1, \cdots, c_{T-1}, u_{T-1}, x_T)^i\}_{i=1}^{nbTraj}$ transformed into a set of system transitions $\mathcal{F} = \{(x_t^l, u_t^l, c_t^l, x_{t+1}^l)\}_{l=1}^{\#\mathcal{F}}$.

Fitted Q iteration computes from $\mathcal{F}$ the functions $\hat{Q}_1$, $\hat{Q}_2$, ..., $\hat{Q}_N$, approximations of Q_1, Q_2, ..., Q_N.

Computation done iteratively by solving a sequence of standard supervised learning (SL) problems. Training sample for the k^{th} $(k \geq 1)$ problem is $\left\{\left((x_t^l, u_t^l),\ c_t^l + \gamma \min_{u \in U} \hat{Q}_{k-1}(x_{t+1}^l, u)\right)\right\}_{l=1}^{\#\mathcal{F}}$ with $\hat{Q}_0(x, u) \equiv 0$. From the k^{th} training sample, the supervised learning algorithm outputs $\hat{Q}_k$.

$\hat{\mu}_N^*(x) \in \arg\min_{u \in U} \hat{Q}_N(x, u)$ is taken as approximation of $\mu^*(x)$.

Fitted Q iteration: some remarks

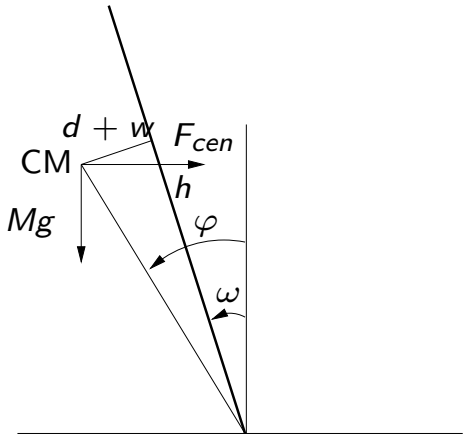
Performances of the algorithm depends on the supervised learning (SL) method chosen.

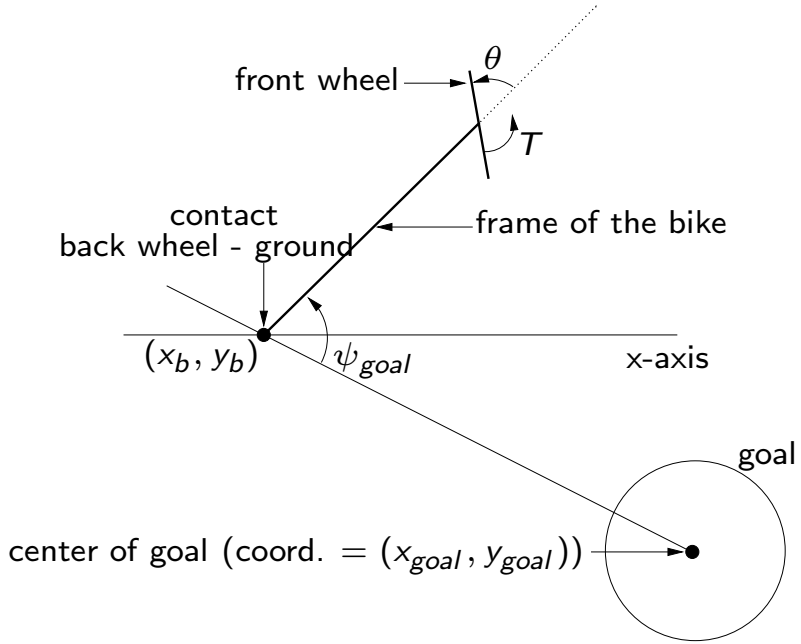
Excellent performances have been observed when combined with supervised learning methods based on ensemble of regression trees.

Works also for stochastic systems

Consistency can be ensured under appropriate assumptions on the SL method, the sampling process, the system dynamics and the cost function.

Illustration I: Bicycle balancing and riding



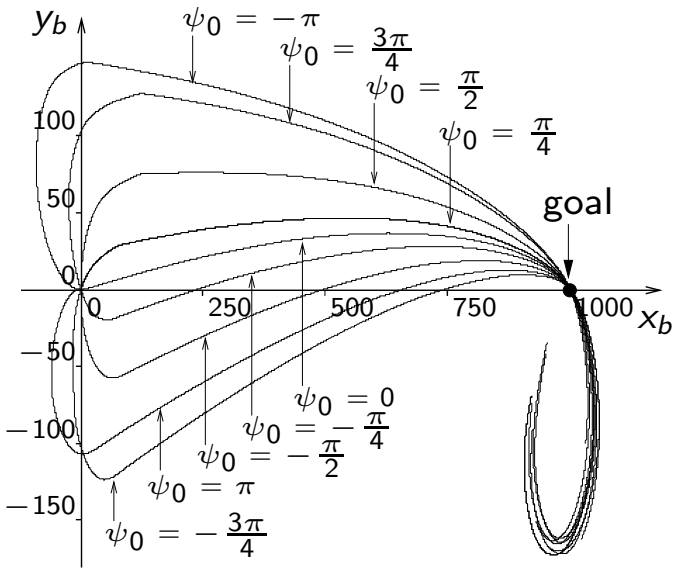


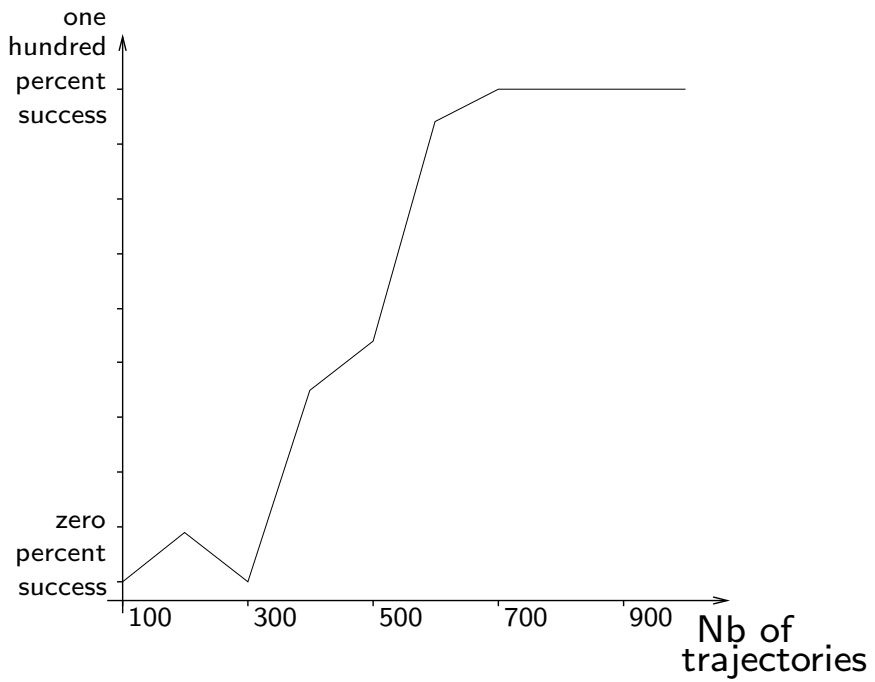
- Stochastic problem
- Seven states variables and two control actions
- Time between t and $t+1 = 0.01\,s$
- Cost function of the type:

$$c(x_t, u_t) = \begin{cases} 1 & \text{if bicycle has fallen} \\ coeff. * (|\psi_{goal_{t+1}}| - |\psi_{goal_t}|) & \text{otherwise} \end{cases}$$

Illustration I: Results

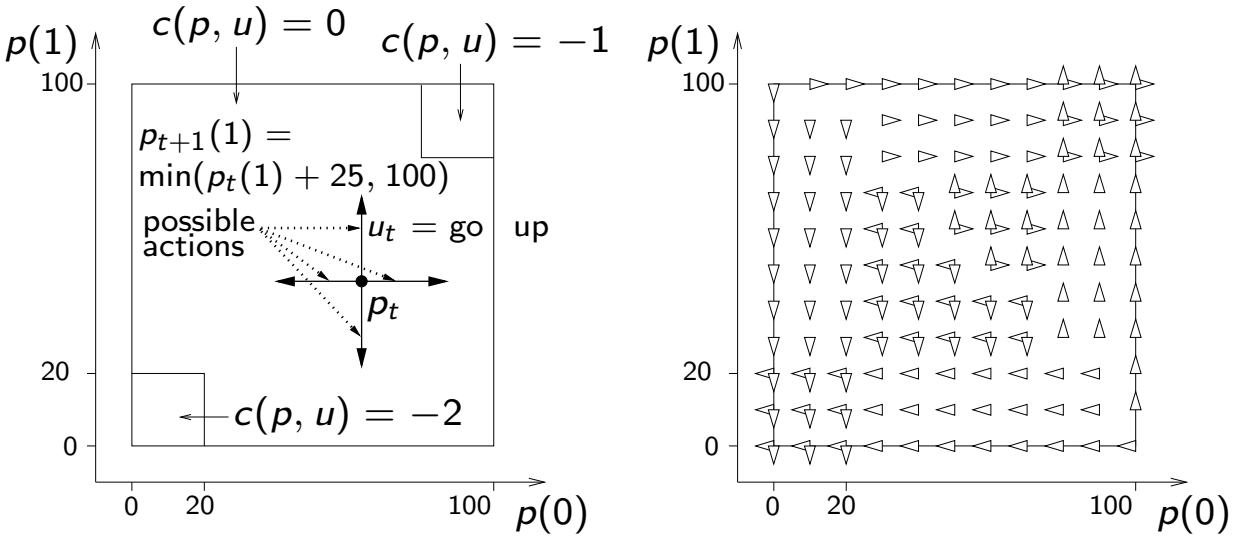
Trajectories generation: action taken at random, bicycle initially far from the goal and trajectory ends when bicycle falls.

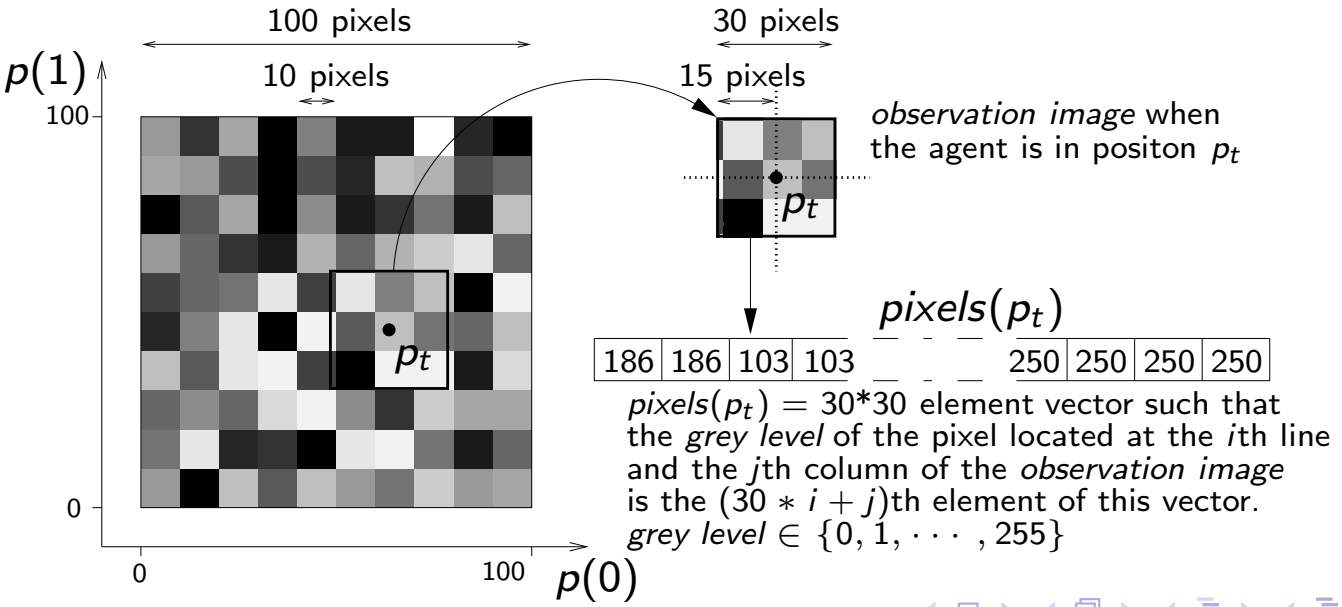


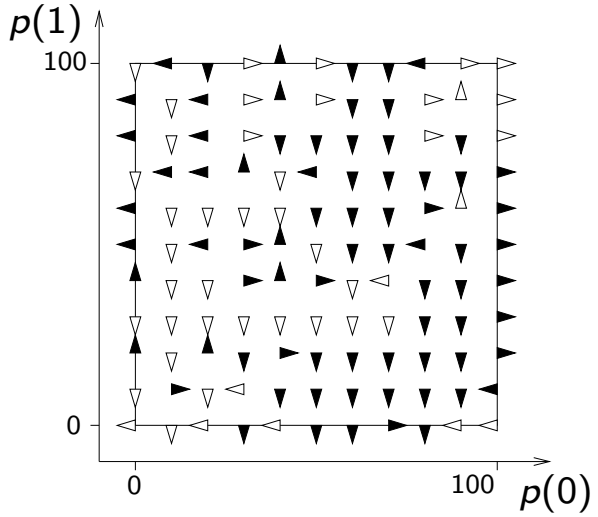


Q-learning with neural networks: $100,000$ times trajectories needed to compute a policy that drives the bicycle to the goal !!!!

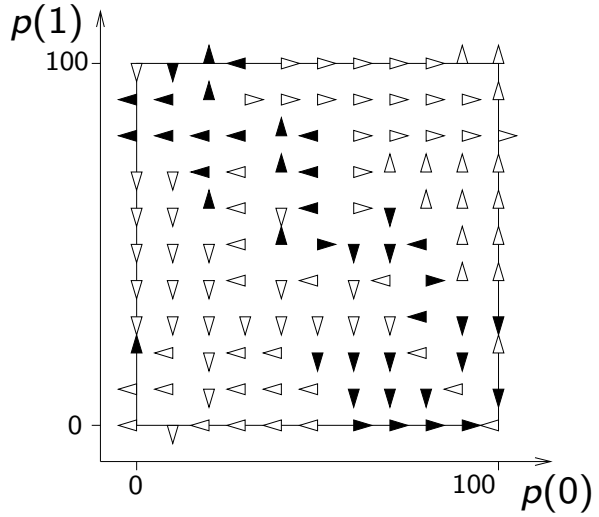
Illustration II: Navigation from visual percepts



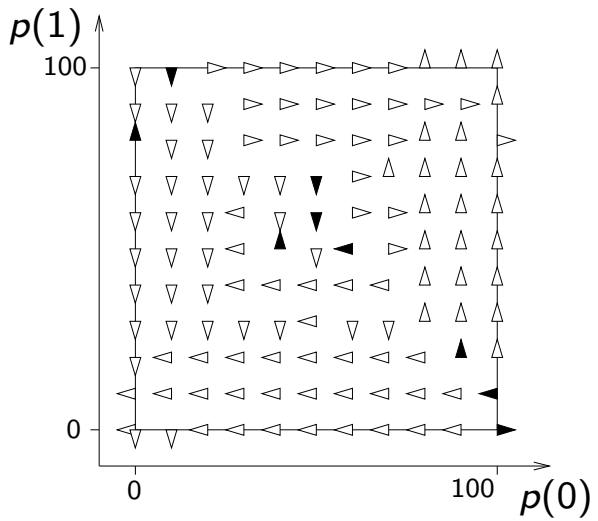




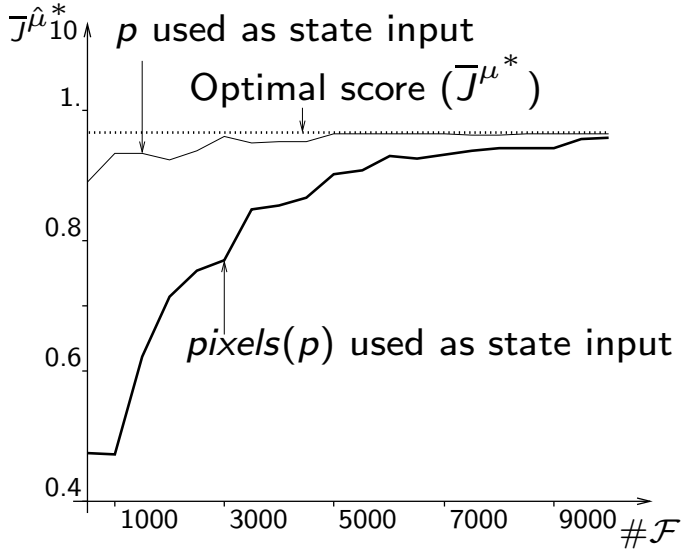
(a) $\hat{\mu}^*_{10}$, 500 system trans.



(b) $\hat{\mu}^*_{10}$, 2000 system trans.



(c) $\hat{\mu}^*_{10}$, 8000 system trans.



(d) score versus nb system trans.

Illustration III: Computation of Structured Treatment Interruption (STI) for HIV+ patients

- STI for HIV: to cycle the patient on and off drug therapy
- In some remarkable cases, STI strategies have enabled the patients to maintain immune control over the virus in the absence of treatment
- STIs offer patients periods of relief from treatment
- We want to compute optimal STI strategies.

Illustration III: What can reinforcement learning techniques offer ?

- Have the potential to infer from clinical data good STI strategies, without modeling the HIV infection dynamics.
- Clinical data: time evolution of patient's state ($CD4^+$ T cell count, systemic costs of the drugs, etc) recorded at discrete-time instant and sequence of drugs administered.
- Clinical data can be seen as trajectories of the immune system responding to treatment.

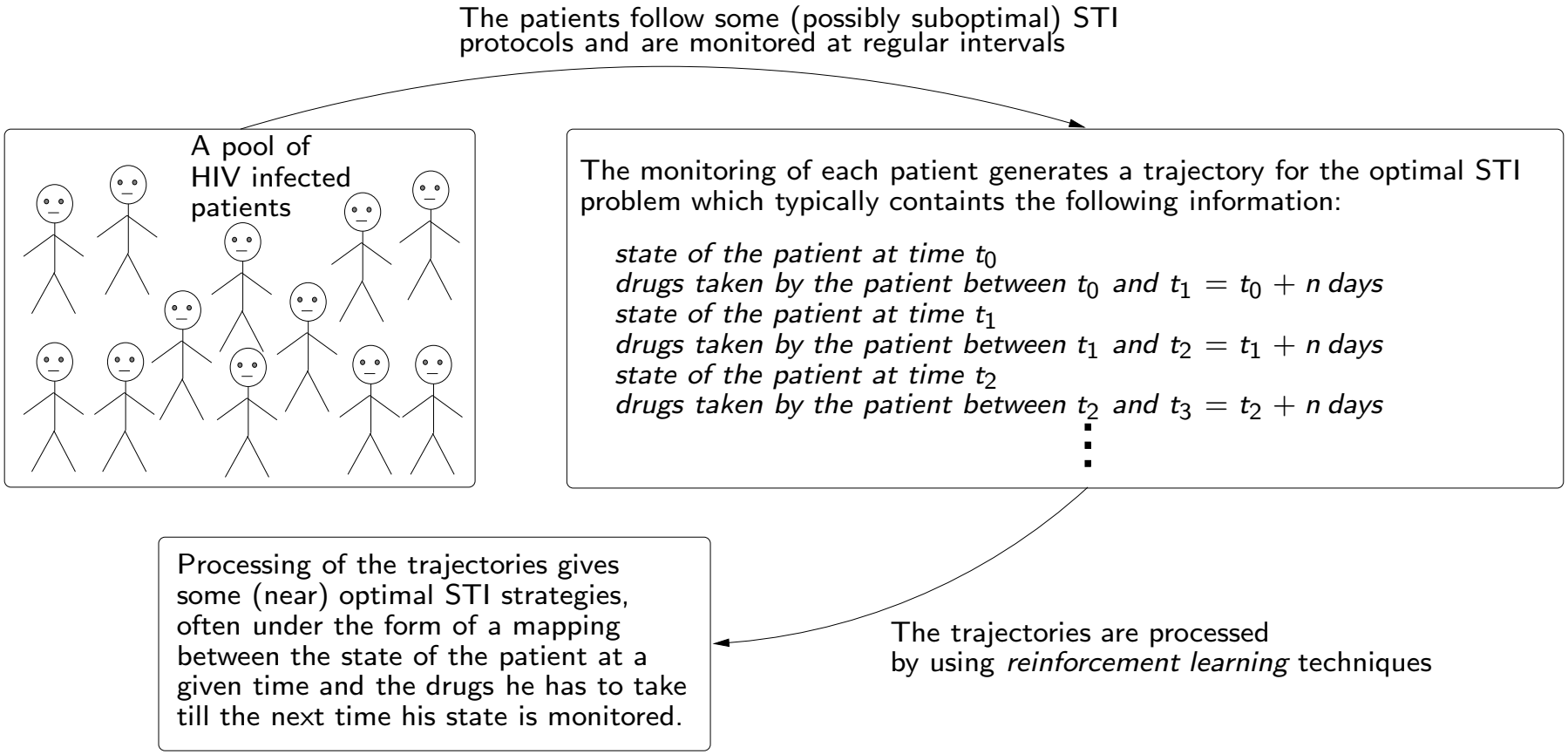


Figure: Determination of optimal STI strategies from clinical data by using reinforcement learning algorithms: the overall principle.

Illustration III: state of the research

- Promising results have been obtained by using "fitted Q iteration" to analyze some artificially generated clinical data.
- Next step: to analyze real-life clinical data generated by the SMART study (Smart Management of Anti-Retroviral Therapies).

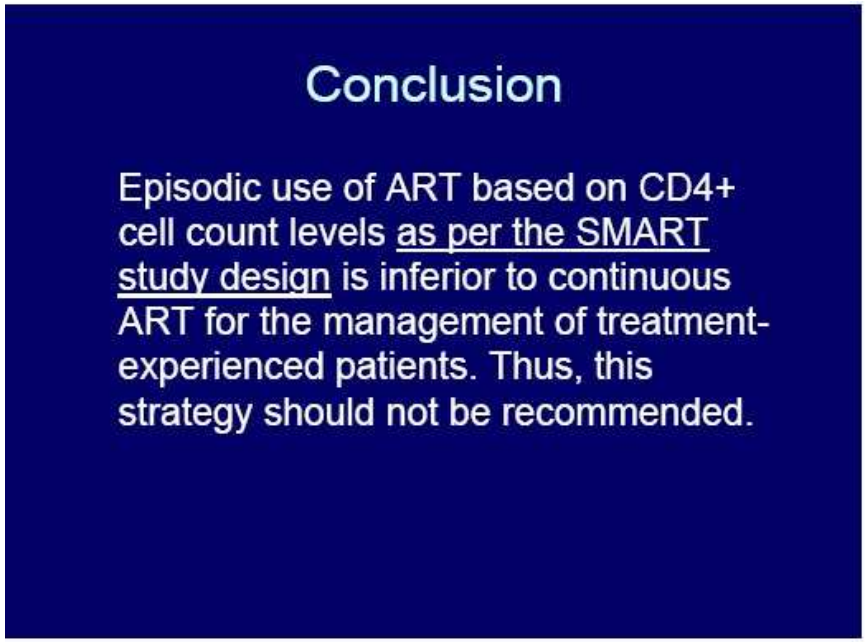


Figure: Taken from http://www.cpcra.org/docs/pubs/2006/croi2006-smart.pdf

Conclusions

- Fitted Q iteration algorithm (combined with ensemble of regression trees) has been evaluated on several problems and was consistently performing much better than other reinforcement learning algorithms.
- Are its performances sufficient to lead to many successful real-life applications ? I think YES !!!
- Why has this algorithm not been proposed before ?

Fitted Q iteration: future research

- Computational burdens grow with the number of trajectories $\Rightarrow$ problems with on-line applications. Possible solution: to keep only the most informative trajectories.
- What are really the best supervised learning algorithm to use in the inner loop of the fitted Q iteration process ? Several criteria need to be considered: distribution of the data, computational burdens, numerical stability of the algorithm, etc.
- Customization of SL methods (e.g. split criteria in trees other than variance reduction, etc)

Supervised learning in dynamic programming: general view

- Dynamic programming: resolution of optimal control problems by extending iteratively the optimization horizon.
- Two main algorithms: value iteration and policy iteration.
- Fitted Q iteration: based on the value iteration algorithm. Fitted Q iteration can be extended to the case where dynamics and cost function are known to become an Approximate Value Iteration algorithm (AVI).
- Approximate Policy Iteration algorithms (API) based on SL have recently been proposed. Can also be adapted to the case where only trajectories are available.
- For both SL based AVI and API, problem of generation of the right trajectories is extremely important.
- How to put all these works into a unified framework ?

Beyond dynamic programming...

- Standard Model Predictive Control (MPC) formulation: solve in a receding time manner a sequence of open-loop deterministic optimal control problems by relying on some classical optimization algorithms (sequential quadratic programming, interior point methods, etc)
- Could SL based dynamic programming be good optimizers for MPC ? Have at least the advantage of not being intrinsically suboptimal when the system is stochastic !!! But problem with computation of $\max_{u \in U} model(x, u)$ when dealing with large U.
- How to extend MPC-like formulations to stochastic systems ? Solutions have been proposed for problems with discrete disturbance spaces but dimension of the search space for the optimization algorithm is $O(number\_of\_disturbances^{optimization\_horizon})$. Research direction: selection of a subset of relevant disturbances.

Planning under uncertainties: an example

Development of MPC-like algorithms for scheduling production under uncertainties through selection of "interesting disturbance scenarios".

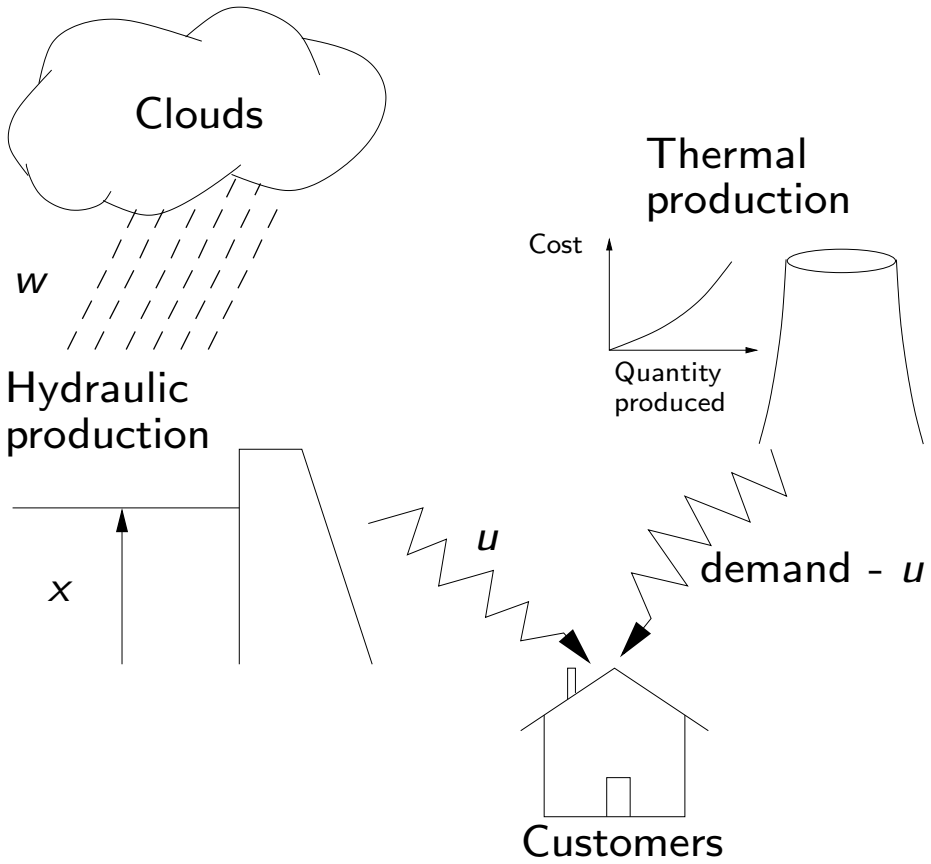


Figure: A typical hydro power plant production scheduling problem.

Additional readings

"Tree-based batch mode reinforcement learning". D. Ernst, P. Geurts and L. Wehenkel. In Journal of Machine Learning Research. April 2005, Volume 6, pages 503-556.

"Selecting concise sets of samples for a reinforcement learning agent". D. Ernst. In Proceedings of CIRAS 2005, 14-16 December 2005, Singapore. (6 pages)

"Clinical data based optimal STI strategies for HIV: a reinforcement learning approach". D. Ernst, G.B. Stan, J. Goncalves and L. Wehenkel. In Proceedings of Benelearn 2006, 11-12 May 2006, Ghent, Belgium. (8 pages)

"Reinforcement learning with raw image pixels as state input". D. Ernst, R. Marée and L. Wehenkel. International Workshop on Intelligent Computing in Pattern Analysis/Synthesis (IWICPAS). Proceedings series: LNCS, Volume 4153, page 446-454, August 2006.

"Reinforcement learning versus model predictive control: a comparison". D. Ernst, M. Glavic, F. Capitanescu and L. Wehenkel. Submitted.

"Planning under uncertainties by solving non-linear optimization problems: a complexity analysis". D. Ernst, B. Defourny and L. Wehenkel. In preparation.