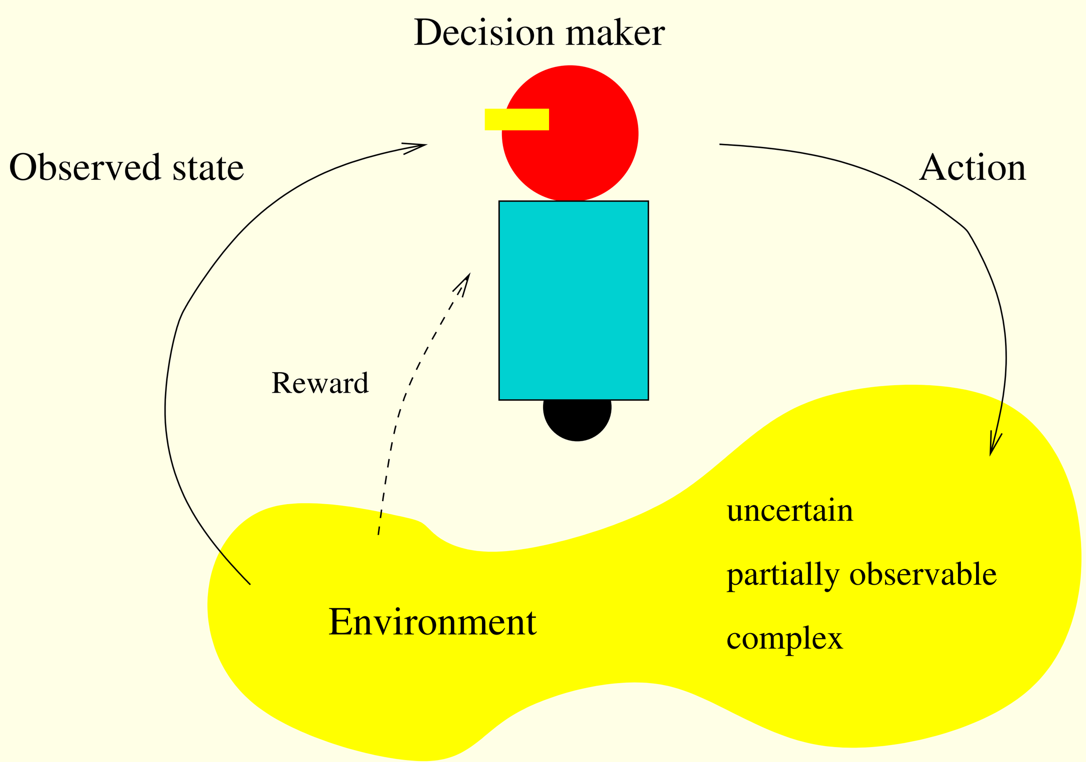
# Sparsity in Adaptive Control

**Philippe Preux, Sertan Girgin**, INRIA, LIFL, CNRS, University of Lille, France, philippe.preux@inria.fr

**Abstract:** We investigate methods and algorithms to obtain sparse representations in the context of adaptive control. We are particularly interested in situations in which we look for a control in an unknown, stochastic, possibly non stationary environments, using no prior knowledge. Here, we present our work based on the use of cascade-correlation networks which yields very sparse representations, yet keeping the ability to obtain highly performing controls.

## 1. Approximate Dynamic Programming and Reinforcement Learning

### 1.1 The situation




### 1.2 Formal framework

Given a Markov decision problem:

- a set of instants of decision, $t \in T$
- a set of states $x \in \mathcal{X}$ (discrete or continuous)
- a set of actions $a \in \mathcal{A}$
- a transition function: $\mathcal{P}(x, a, x') \equiv Pr[x_{t+1} = x' | x_t = x, a_t = a]$
- a return function: $\mathcal{R}(x, a, x') \equiv \mathbb{E}[r_{t+1} = x' | x_t = x, a_t = a] \in \mathbb{R}$
- an objective such as: optimize $J(x) \equiv \sum_{k \geq 0} \gamma^k r_{t+k} | x_t = x, \gamma \in [0, 1)$
- everything is stationary.

find: the optimal policy $\pi^*$ that optimizes the objective function.
Theorem (Blackwell): in this setting, $\pi^*$ is stationary and deterministic: $\pi^* : \mathcal{X} \rightarrow \mathcal{A}$: in each state, there is an optimal action(or possibly, several strictly equivalent optimal actions).

### 1.3 ADP vs.RL

**Approximate dynamic programming** (ADP) when the PDM is known,
**Reinforcement Learning** (RL) when only $\mathcal{X}$ and $\mathcal{A}$ are known.

### 1.4 How?

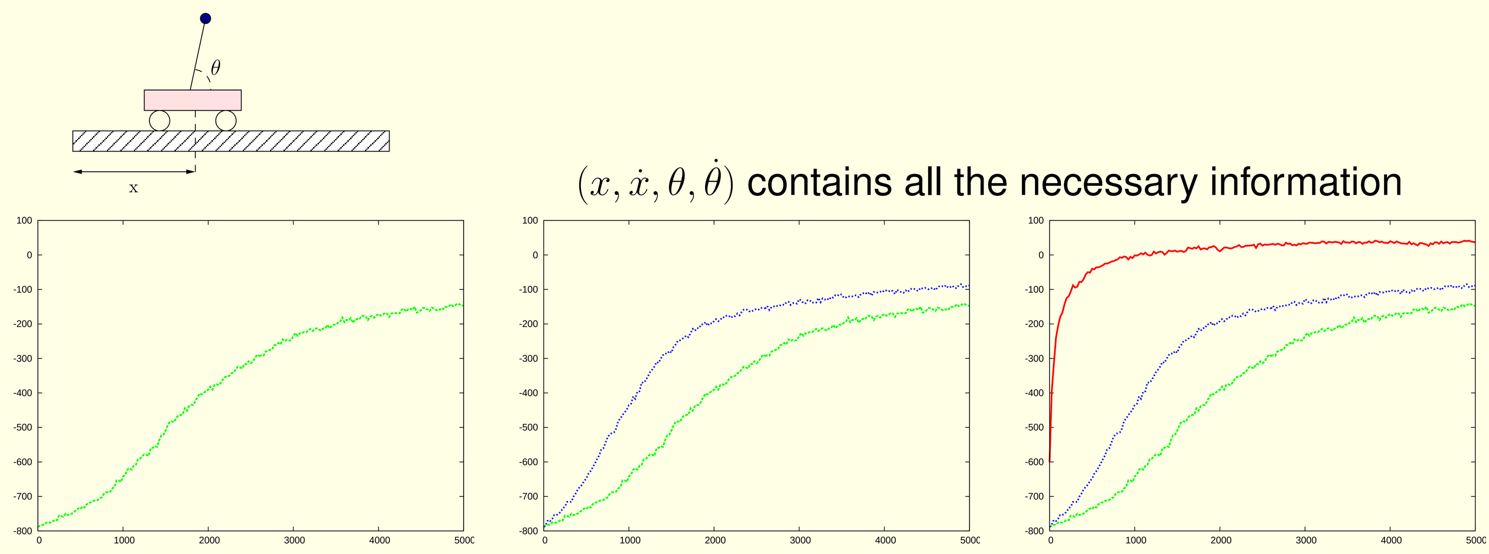
**Compute the value function** of the optimal policy (ADP and RL): $V(x) = \max(\mathbb{E}J(x))$.
This $V$ is the solution of a non linear equation (Bellman optimality equation) that can be computed in various ways.
$\pi^*$ is easy to deduce from this $V$.

**Compute directly the optimal policy** (RL only) by sampling trajectories.

$V$ is obtained asymptotically by solving a series of regression problems: $V_0 \rightsquigarrow V_1 \rightsquigarrow ... V_i \rightsquigarrow ... V$.

### 1.5 Representation of states

- in real settings, we do not know how to represent states in an "optimal" way, that is, such that the problem is Markovian, and such that the computational cost is lowered.
- even on toy examples, the Markovian representation may be enriched to get improved performance:




$(x, \dot{x}, \theta, \dot{\theta})$ contains all the necessary information

$\mathcal{X} = (x, \dot{x}, \theta, \dot{\theta})$ | $\mathcal{X} = (x, \dot{x}, \theta, \dot{\theta}, \sin(\theta), \cos(\theta))$ | $\mathcal{X} = (x, \dot{x}, \theta, \dot{\theta})$ and automatically discovered features

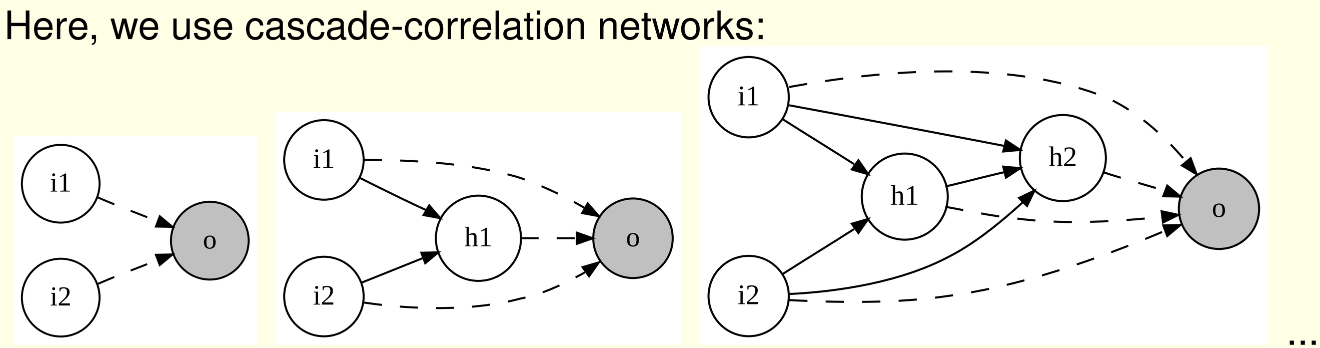
## 2. Non Parametric Function Approximation for ADP & RL

### 2.1 Basics

We use non parametric function approximators to obtain at the same time:

- a sparse approximation of $V$,
- a "good" representation of states.

Here, we use cascade-correlation networks:




(These 3 figures show how a CCN grows)

The +'s:

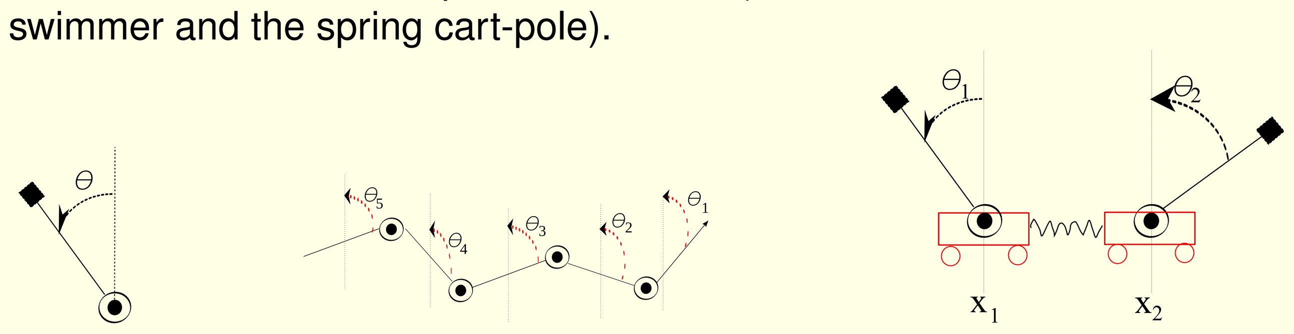
- very efficient,
- works quite well in practice.

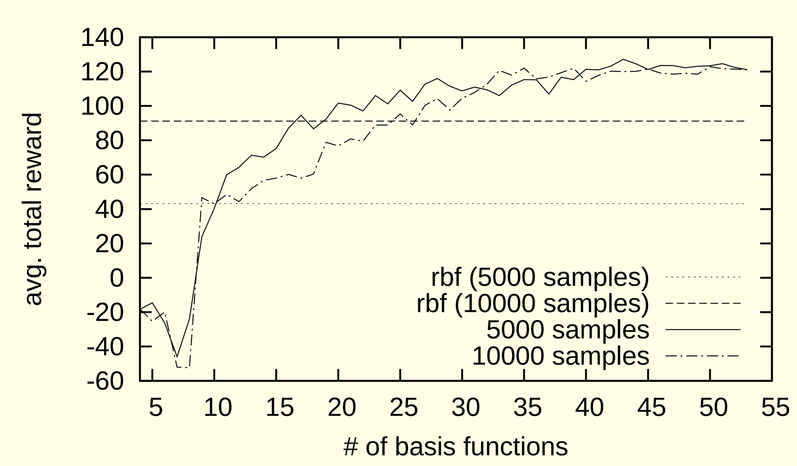
The -'s:

- not theoretically grounded,
- very difficult to interpret,
- grows but never shrinks.

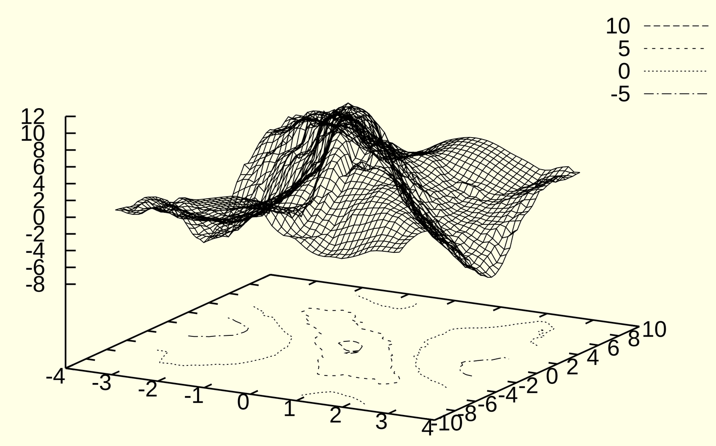
### 2.2 Experiments in RL

Results on the inverted pendulum task: (same kind of results for other tasks, such as the swimmer and the spring cart-pole).




CCN produces a very sparse representation of the value function, which yields excellent performance: with only 10 hidden units, we do better than a 512 RBF-network for instance.



The function being learned.

We obtain the same kind of results in the case of Approximate Dynamic Programming.

## 3. Conclusion

- very sparse solutions in adaptive control applications,
- very sparse representations along with highly performing policies,
- efficient in terms of computation time,

## 4. Future work

- scalability towards larger problems,
- various issues in control, and the level of approximations required to obtain good policies.

## References


[1] S. Girgin and Ph. Preux. Feature discovery in reinforcement learning using genetic programming. In *Proc.*$11^{th}$ *Euro-GP*, volume 4971 of *LNCS*, pages 218–229. Springer, 2008.

[2] S.Girgin and Ph.Preux. Incremental basis function expansion in reinforcement learning using cascade-correlation networks. In *Proc. ICML-A*, pages 75–82. IEEE Press, 2008.

[3] S. Girgin and Ph. Preux. Basis expansion in natural actor critic methods. In *Recent Advances in Reinforcement Learning*, volume 5323 of *LNAI*, pages 111–124. Springer, 2009.

[4] Ph. Preux, S. Girgin, and M. Loth. Feature discovery in approximate dynamic programming. In *Proc. IEEE ADPRL*, 2009.