Empowerment for Continuous Agent-Environment Systems

Tobias Jung

Joint work with Daniel Polani

Contents:

1. Motivation and definition (discrete case)
2. Approximate computation (continuous case)
3. Application: empowerment-based control

This research was partially funded by the EC as part of the FEELIX GROWING project.

What is empowerment?

Question

How can we find interesting states/actions in an MDP ...

... without actually solving the whole problem first (with DP/RL methods)?

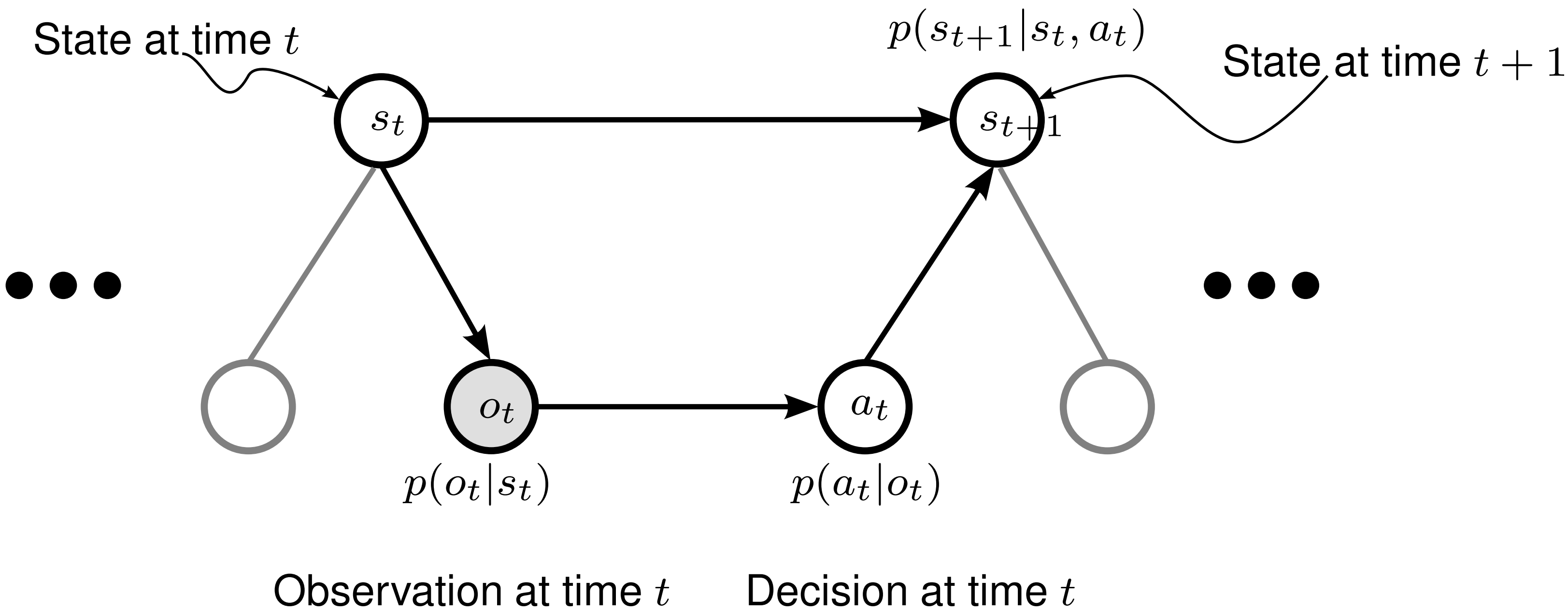
Motivation: we seek to identify general principles that *may* help us answering this question.

We present: Empowerment (Heuristic: 'Being in control of one's own future is good.')

- Information-theoretic formulation.
- Independent of specific goals, only considers 'general' properties of the dynamics of the decision process.
- Computable from local quantities.

A graphical model

Represent: agent-environment interaction over time as a graphical model (memory-less):



With

- Random variables s_t, o_t, a_t, $t = 1, 2, \ldots$ (having finite domain)
- Conditional distribution tables for
  - Sensor: $p(o_t|s_t)$ (without losing generality we will ignore sensors)
  - Agent's decision: $p(a_t|o_t)$
  - State transition: $p(s_{t+1}|s_t, a_t)$

Definition of empowerment (discrete case)

Now: let's consider the single transition from (a fixed) s to s'.

Definition: For every state s we define **empowerment** $C(s)$ as the *channel capacity* between selection of an action a and resulting successor state s':

$$\boxed{C(s) := \max_{\vec{p}(a)} I(\mathcal{S}_s : \mathcal{A}_s) = \max_{\vec{p}(a)} \sum_{s'} \sum_{a} p(a) p(s'|s,a) \log \left\{ \frac{p(s'|s,a)}{\sum_{a'} p(s'|s,a')p(a')} \right\}}$$

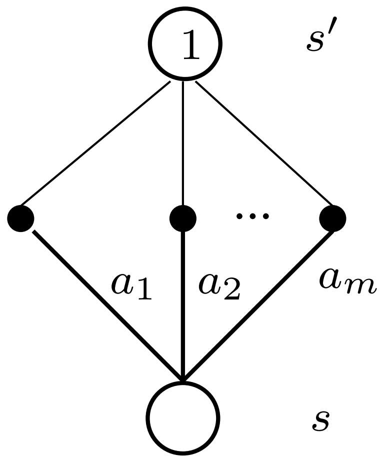
where

$\mathcal{A}_s$	discrete random variable modeling selection of action
$\vec{p}(a)$	distribution over $\mathcal{A}_s$ (number-of-actions vector)
$\mathcal{S}_s$	discrete random variable modeling occurance of successor state given s
$p(s'|s,a)$	transition probabilities (dynamics of the world)

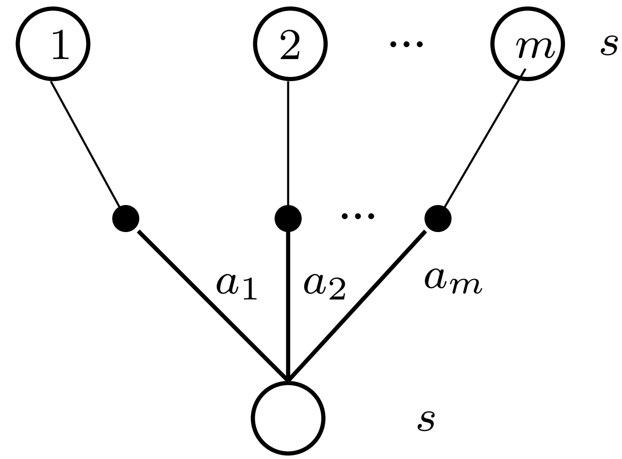
Algorithm: $C(s)$ can be computed via **Blahut-Arimoto**, if transition probs are known.

Illustration: understanding empowerment

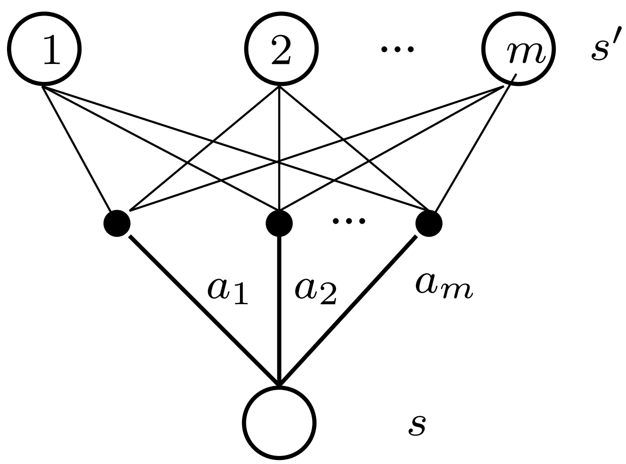
Special cases: suppose $p(s'|s,a)$ is one of the following:



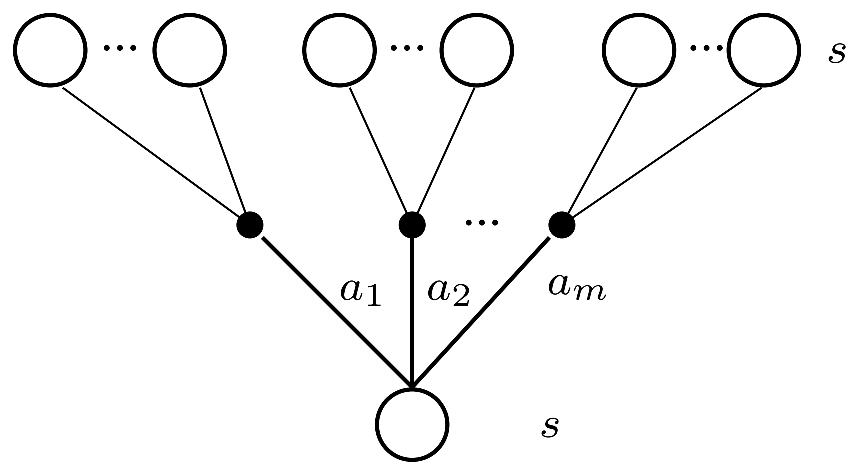
Empowerment zero
(all actions leading to the same successor state)



Empowerment max ($= \log(m)$)
(all actions leading to different successor states)



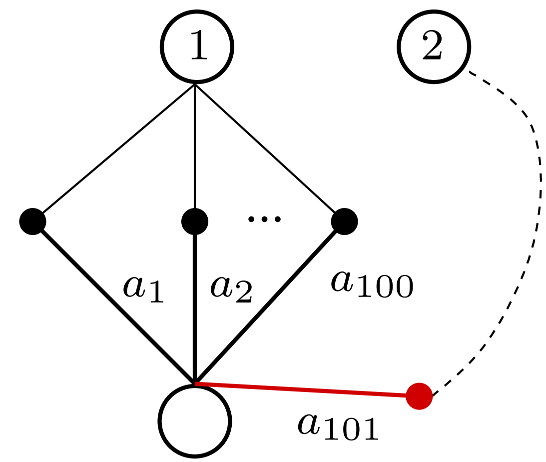
Empowerment zero
(if $p(s'|s,a) = p(s'|s)$)



Empowerment max ($= \log(m)$)
(all actions leading to disjunct sets of possible successor states)

Illustration: empowerment vs. mutual information

For all cases on the preceding slide, empowerment (channel capacity) was equal to the mutual information. Of course, this is not always the case ...

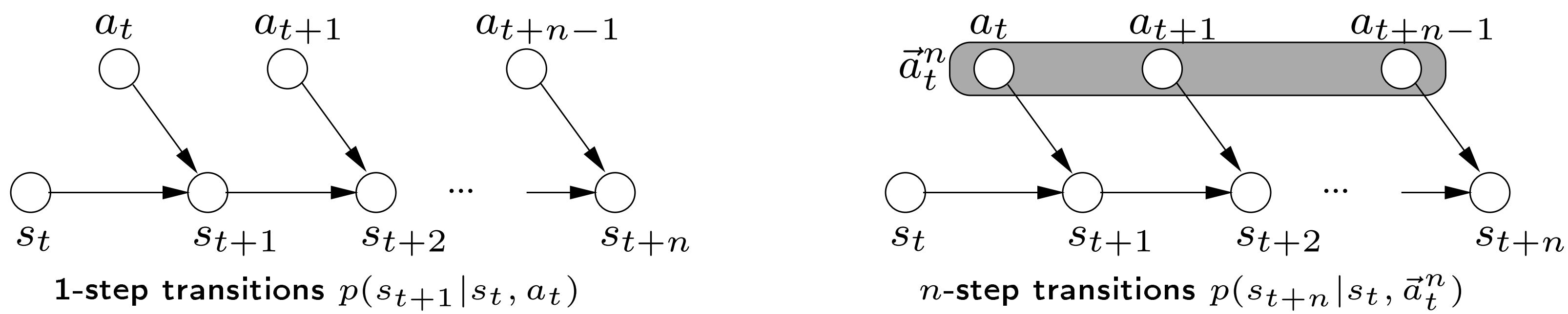


Mutual info: 0.055
Empowerment: 0.6931 ($\exp(\cdot) = 2$)

$$p^*(a) = \begin{bmatrix} p^*(a_1) \\ \vdots \\ p^*(a_{100}) \\ {\color{red} p^*(a_{101})} \end{bmatrix} = \begin{bmatrix} 0.5/100 \\ \vdots \\ 0.5/100 \\ {\color{red} 0.5} \end{bmatrix}$$

Things are getting even more interesting once we start to consider temporally-extended actions (n-step) ....

N-step empowerment



1-step transitions: dynamics of the system naturally modeled at the level of 1-step transitions

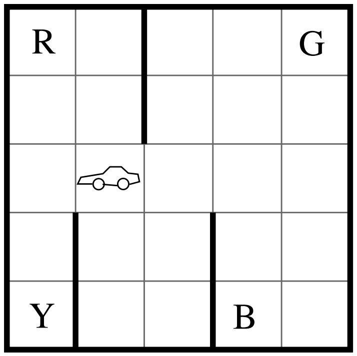
n-step transitions: consider open-loop **action sequences** $\vec{a}_t^n$ of n 1-step actions $\vec{a}_t^n := \big(a_t, \ldots, a_{t+n-1}\big)$ and induced transitions from s_t to s_{t+n} under $\vec{a}_t^n$:

$$p(s_{t+n}|s_t, \vec{a}_t^n) = \sum_{s_{t+n}} \cdots \sum_{s_{t+1}} p(s_{t+n}|s_{t+n-1}, a_{t+n-1}) \cdots p(s_{t+1}|s_t, a_t)$$

n-step empowerment: In general, will consider empowerment for n-step transitions [which technically doesn't change anything, just replace $p(s_{t+1}|s_t, a_t)$ by $p(s_{t+n}|s_t, \vec{a}_t^n)$ and loop over all possible n-step actions instead of over all 1-step actions].

Remark: set of possible n-step actions is formed through exhaustive enumeration [thus number-of-n-step-actions $=$ number-of-1-step-actions^n].

Example: taxi-domain



State: factored representation (5*5*5*4=500 states)

- x-location $\{1, \ldots, 5\}$
- y-location $\{1, \ldots, 5\}$
- passenger {'Y','R','B','G','Car'}
- destination {'Y','R','B','G'}

State: flat representation $\{1, \ldots, 500\}$ **!**

Actions: 'North', 'South', 'East', 'West', 'Pick-up', 'Drop-off'

Transitions:

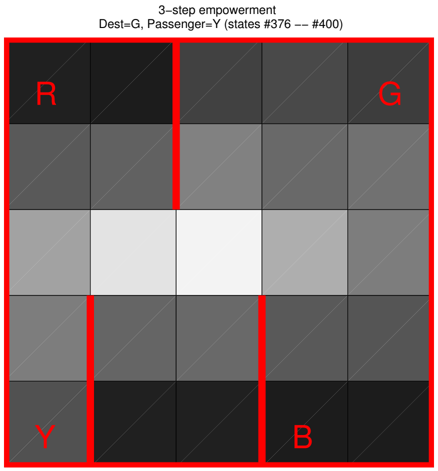
- Movement 80% successful, 20% deviation to left/right
- 'Pick-up' only succeeds if passenger waits at current location (else no effect)
- 'Drop-off' only succeeds if destination is at current location (else no effect)

Note:

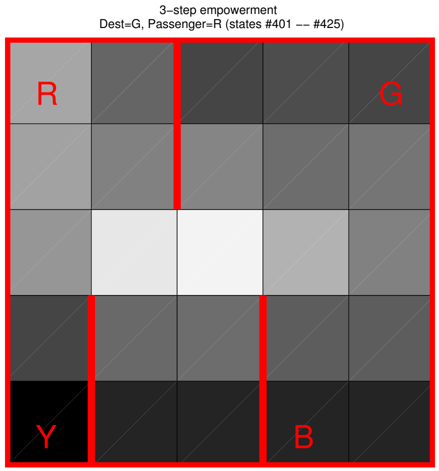
- **No reward.** We just look at the dynamics, see if we can do something ...
- Episode ends: once a passenger is dropped off, we immediately move to the center (and randomly generate new start/destination)

Taxi domain: 3-step empowerment (pick-up)

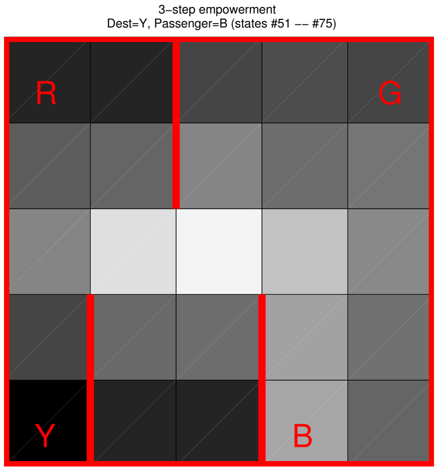
Now let's look at the 3-step empowerment of states if a passenger is ...



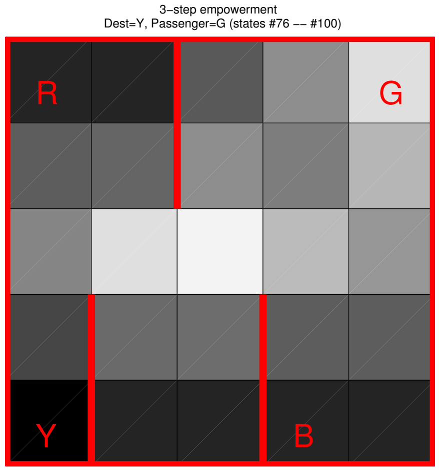
...waiting at 'Y'



...waiting at 'R'



...waiting at 'B'

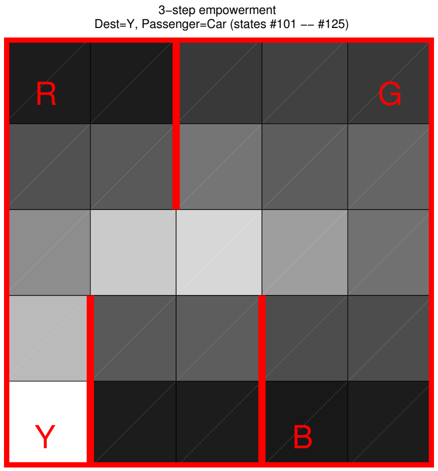


...waiting at 'G'

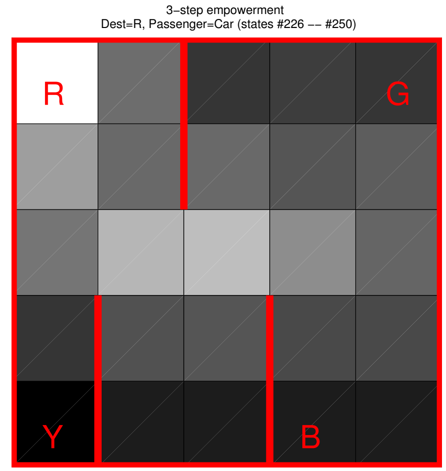
... and what happens if a passenger is picked up?

Taxi domain: 3-step empowerment (drop-off)

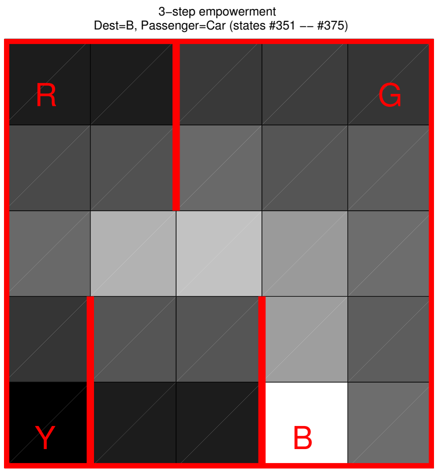
Once the passenger is picked up, empowerment changes like this if the passenger wants to



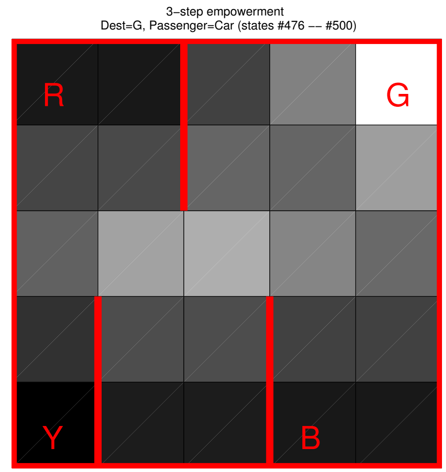
... go to ’Y’



... go to ’R’



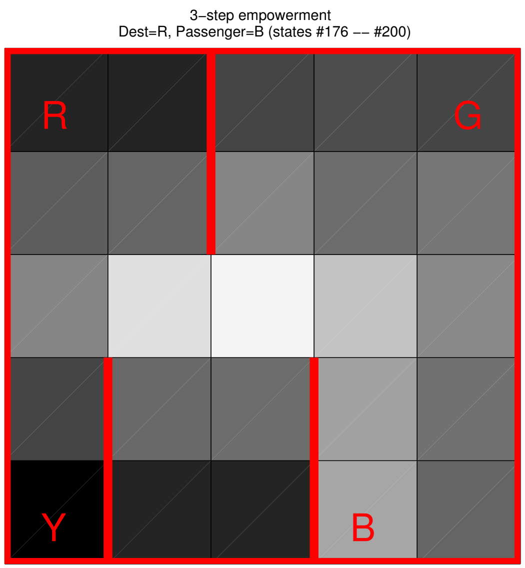
... go to ’B’



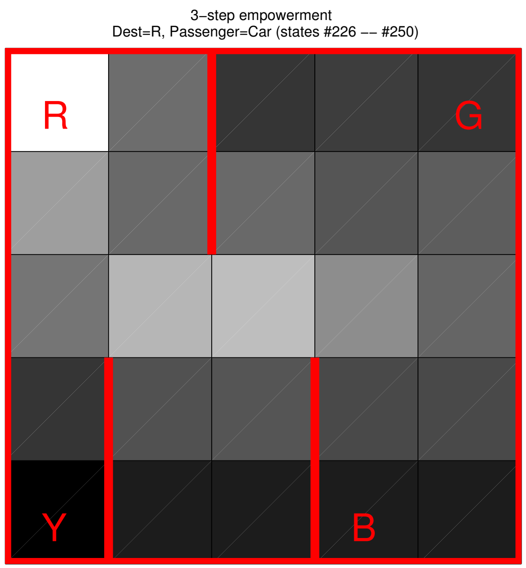
... go to ’G’

Taxi domain: 3-step empowerment (complete)

In other words: Assume our 'goal' is to bring the passenger from 'B' to 'R'.



Before 'pick-up'



Immediately after 'pick-up', before 'drop-off'

Observe:

- Following the trail of highly-empowered states brings us to each of the two sub-goal states.
- In fact, we could [nearly] solve the taxi-domain just looking at the empowerment values and greedily choosing actions accordingly.
- Which is remarkable, because we didn't have to introduce an artificial 'reward' to make the system behave as we want it to behave.

Summary (so far)

Empowerment: 'Hub states'

- **Information-theoretic** formulation (cf. bottleneck states = graph-theoretic formulation)
- **Unsupervised & goal-free:** only considers general properties of the dynamics of the decision process [much like PCA, which only considers general properties of the data, i.e. the variance, but not if a particular direction is actually helpful in solving the problem at hand].
- **Local:** computed from transition function alone (cf. cost-to-go function in RL=global).
- Considers effects of actions on different time-scales.

Applications:

- Identify 'interesting' states to create *possible* subgoals to facilitate planning/learning at different levels of abstraction.
- Identify 'irrelevant' actions (actions that eventually have the same outcome).
- Drive exploration (instead of blindly trying out all possible actions, empowerment gives us a heuristic which to try first).

However, up to now we were only able to examine empowerment for toy problems with discrete state spaces.

Let's see how we can scale-up to continuous state spaces ...

How can we calculate empowerment in $\mathbb{R}^d$?

Empowerment (continuous case)

Objective: In the general case we would have to compute

$$\boxed{C(\mathbf{x}) = \sup_{p(\vec{\mathbf{u}}_t^n)} \int_{\mathcal{X}} \int_{\mathcal{U}^n} p(\vec{\mathbf{u}}_t^n) p(\mathbf{x}_{t+n}|\mathbf{x}_t, \vec{\mathbf{u}}_t^n) \log \left\{ \frac{p(\mathbf{x}_{t+n}|\mathbf{x}_t, \vec{\mathbf{u}}_t^n)}{p(\mathbf{x}_{t+n}|\mathbf{x}_t)} \right\} d\mathbf{x}_{t+n}\, d\vec{\mathbf{u}}_t^n}$$

where

$\mathbf{x}_t$	state, a D-dimensional vector ($\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^D$)
$\mathbf{u}_t$	control, a N_A-dimensional vector ($\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^{N_A}$)
$\vec{\mathbf{u}}_t^n$	n-step control, a $N_n := (N_A)^n$-dimensional vector ($\vec{\mathbf{u}}_t^n \in \mathcal{U}^n \subset \mathbb{R}^{N_n}$)
$p(\mathbf{x}_{t+n}|\mathbf{x}_t, \vec{\mathbf{u}}_t^n)$	n-step transition probabilities

Big problem: in practice **intractable** (and no closed form solution possible):

- How to integrate over the D-dimensional state space?
- How to intergrate and maximize over the N_n-dimensional n-step action space?
- And what's with $p(\mathbf{x}_{t+n}|\mathbf{x}_t, \vec{\mathbf{u}}_t^n)$? How do we get the n-step transitions in first place?

Approximating empowerment I

1. Discretize n-step controls to a [small] number of symbolic actions $\vec{a}_\nu$, $\nu = 1, \ldots, N_n$:

$$C(\mathbf{x}) := \max_{p(\vec{a})} \sum_{\nu=1}^{N_n} p(\vec{a}_\nu) \int_{\mathcal{X}} p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu) \log \left\{ \frac{p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu)}{\sum_{i=1}^{N_n} p(\mathbf{x}'|\mathbf{x}, \vec{a}_i) p(\vec{a}_i)} \right\} d\mathbf{x}'$$

where

$p(\mathbf{x}'\|\mathbf{x}, \vec{a}_\nu)$	density modeling transitions from $\mathbf{x}$ to $\mathbf{x}'$ under $\vec{a}_\nu$ (here $\vec{a}_\nu$ translates into a n-step control vector).

2. Use simple Monte-Carlo to evaluate remaining integral over state space:

$$\int_{\mathcal{X}} p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu) \log \left\{ \frac{p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu)}{\sum_{i=1}^{N_n} p(\mathbf{x}'|\mathbf{x}, \vec{a}_i) p(\vec{a}_i)} \right\} d\mathbf{x}' \approx \frac{1}{N_{\mathrm{MC}}} \sum_{j=1}^{N_{\mathrm{MC}}} \log \left[\frac{p(\tilde{\mathbf{x}}'_{\nu,j}|\mathbf{x}, \vec{a}_\nu)}{\sum_{i=1}^{N_n} p(\tilde{\mathbf{x}}'_{\nu,j}|\mathbf{x}, \vec{a}_i) p(\vec{a}_i)} \right]$$

where

N_{MC}	number of samples
$\tilde{\mathbf{x}}'_{\nu,j}$	random sample drawn from $p(\mathbf{x}'\|\mathbf{x}, \vec{a}_\nu)$

Approximating empowerment II

3. Gaussian model: Make sure that $p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu)$ is of a form that allows us to easily draw samples from, e.g. assume it's a Gaussian:

$$p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu) = \mathcal{N}(\mathbf{x}'|\boldsymbol{\mu}_\nu(\mathbf{x}), \boldsymbol{\Sigma}_\nu(\mathbf{x}))$$

where

$\boldsymbol{\mu}_\nu(\mathbf{x})$	D-dimensional mean
$\boldsymbol{\Sigma}_\nu(\mathbf{x})$	$D \times D$ covariance matrix

Remarks:

- **Model-learning:** Gaussian assumption of transitions probs is automatically fulfilled, if model learned by, e.g., Gaussian process regression.

Now we plug all these things into our Blahut-Arimoto algorithm ...

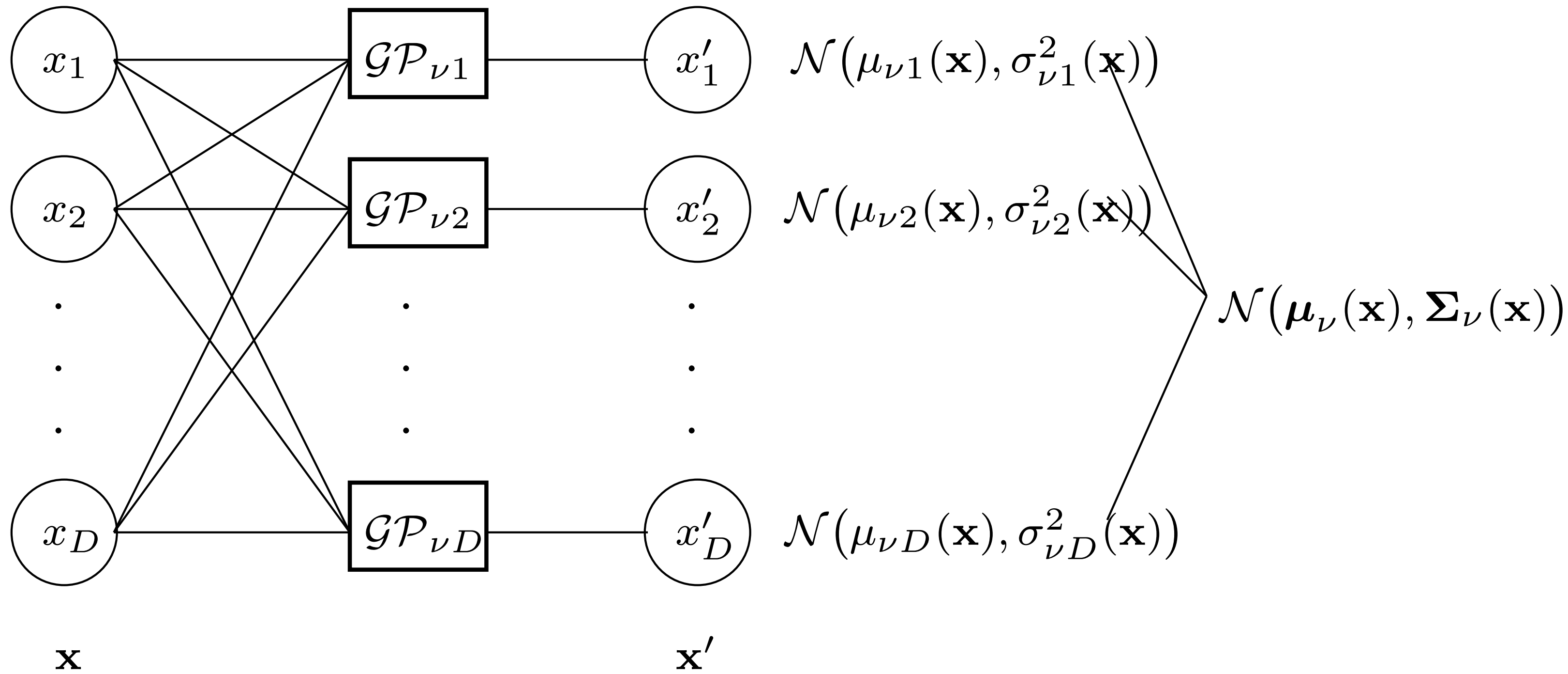
Algorithm: Blahut-Arimoto

(Computational complexity: $k \cdot N_n^2 \cdot N$)

1. **Input:**
   (a) State $\mathbf{x}$ whose empowerment we wish to calculate.
   (b) For every action $\nu = 1, \ldots, N_n$ a state transition model $p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu)$, each fully defined by its mean $\boldsymbol{\mu}_\nu$ and covariance $\boldsymbol{\Sigma}_\nu$.
2. **Initialize:**
   (a) $p_0(\vec{a}_\nu) := 1/N_n$ for $\nu = 1, \ldots, N_n$.
   (b) Draw N samples $\{\tilde{\mathbf{x}}'_{\nu,i}\}_{i=1}^N$ each, from $p(\mathbf{x}'|\mathbf{x}, \vec{a}_\nu) = \mathcal{N}(\boldsymbol{\mu}_\nu(\mathbf{x}), \boldsymbol{\Sigma}_\nu(\mathbf{x}))$ for $\nu = 1, \ldots, N_n$.
   (c) Evaluate $p(\tilde{\mathbf{x}}'_{\nu,i}|\mathbf{x}, \vec{a}_\mu)$ for all $\nu = 1, \ldots, N_n$; $\mu = 1, \ldots, N_n$; $i = 1, \ldots, N$.
3. **Iterate** $k = 1, 2, \ldots$
   (a) $z_k := 0$, $c_{k-1} := 0$
   (b) For $\nu = 1, \ldots, N_n$
      i.
      $$d_{\nu,k-1} := \frac{1}{N} \sum_{j=1}^{N} \log \left[\frac{p(\tilde{\mathbf{x}}'_{\nu,j}|\mathbf{x}, \vec{a}_\nu)}{\sum_{i=1}^{N_n} p(\tilde{\mathbf{x}}'_{\nu,j}|\mathbf{x}, \vec{a}_i) p_{k-1}(\vec{a}_i)} \right]$$
      ii. $c_{k-1} := c_{k-1} + p_{k-1}(\vec{a}_\nu) \cdot d_{\nu,k-1}$
      iii. $p_k(\vec{a}_\nu) := p_{k-1}(\vec{a}_\nu) \cdot \exp\{d_{\nu,k-1}\}$
      iv. $z_k := z_k + p_k(\vec{a}_\nu)$
   (c) For $\nu = 1, \ldots, N_n$
      i. $p_k(\vec{a}_\nu) := p_k(\vec{a}_\nu) \cdot z_k^{-1}$
4. **Output:**
   (a) Empowerment $C(\mathbf{x}) \approx c_{k-1}$ (estimated).
   (b) Distribution $p(\vec{a}) \approx p_{k-1}(\vec{a})$ achieving the maximum mutual information.

Algorithm (sketch): learning a model using GPs

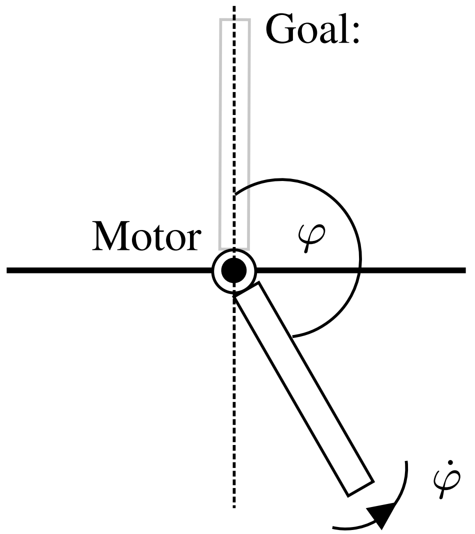
1. Collect sufficiently large number of 1-step transitions $\{\mathbf{x}_\ell, a_\ell, \mathbf{x}'_\ell\}$.
2. Learn 1-step system dynamics from multiple univariate GPs



3. Using that, recursively predict n steps ahead to obtain the desired n-step transition probabilities (which again remains Gaussian when using the Laplace approximation, see [Girard et al., NIPS 2003]).

Experiments

Experiment #1: inverted pendulum



Dynamics: ($l = 1$, $m = 1$, $g = 9.81$, $\mu = 0.05$)

$$\ddot{\varphi}(t) = \frac{-\mu\dot{\varphi}(t) + mgl\sin\varphi(t) + u(t)}{ml^2}$$

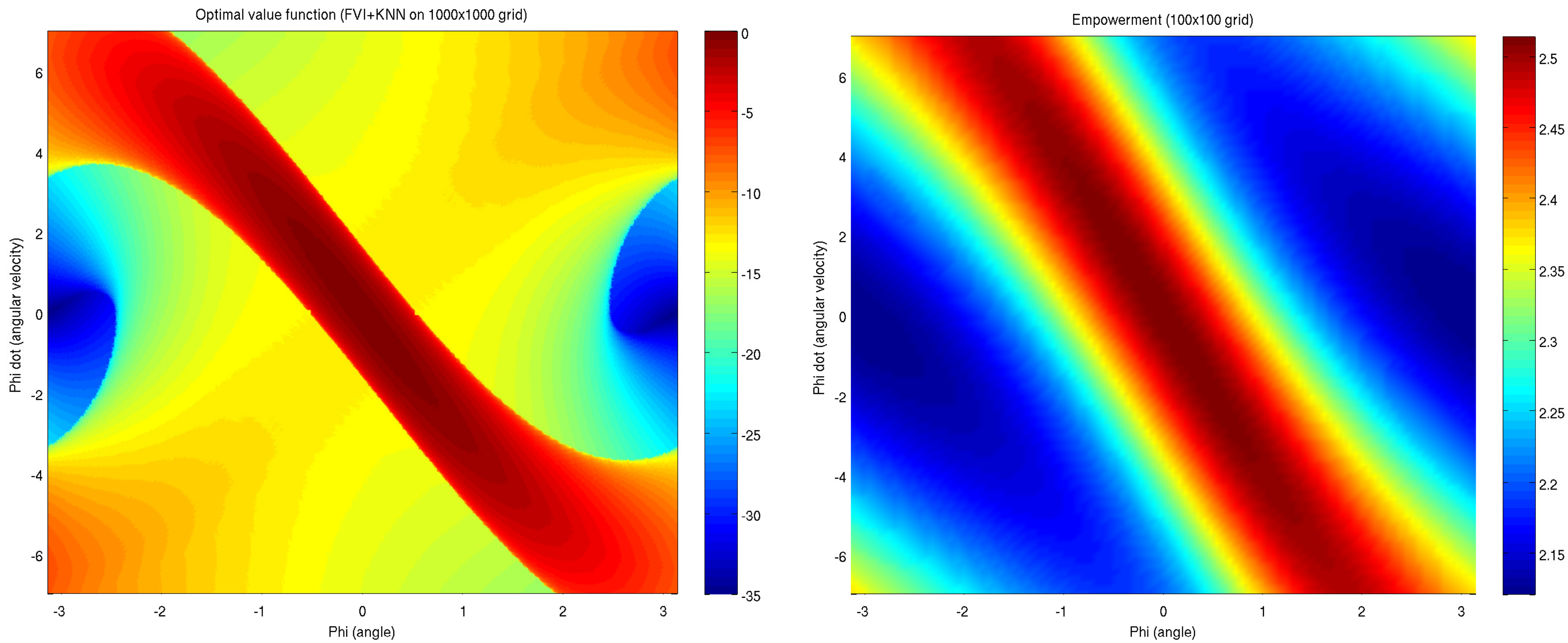
with $u \in \{-5, -2.5, 0, +2.5, +5\}$.

Goal: to give this system *some* purpose, we consider the **pendulum swing-up task**.

Experiment: compare

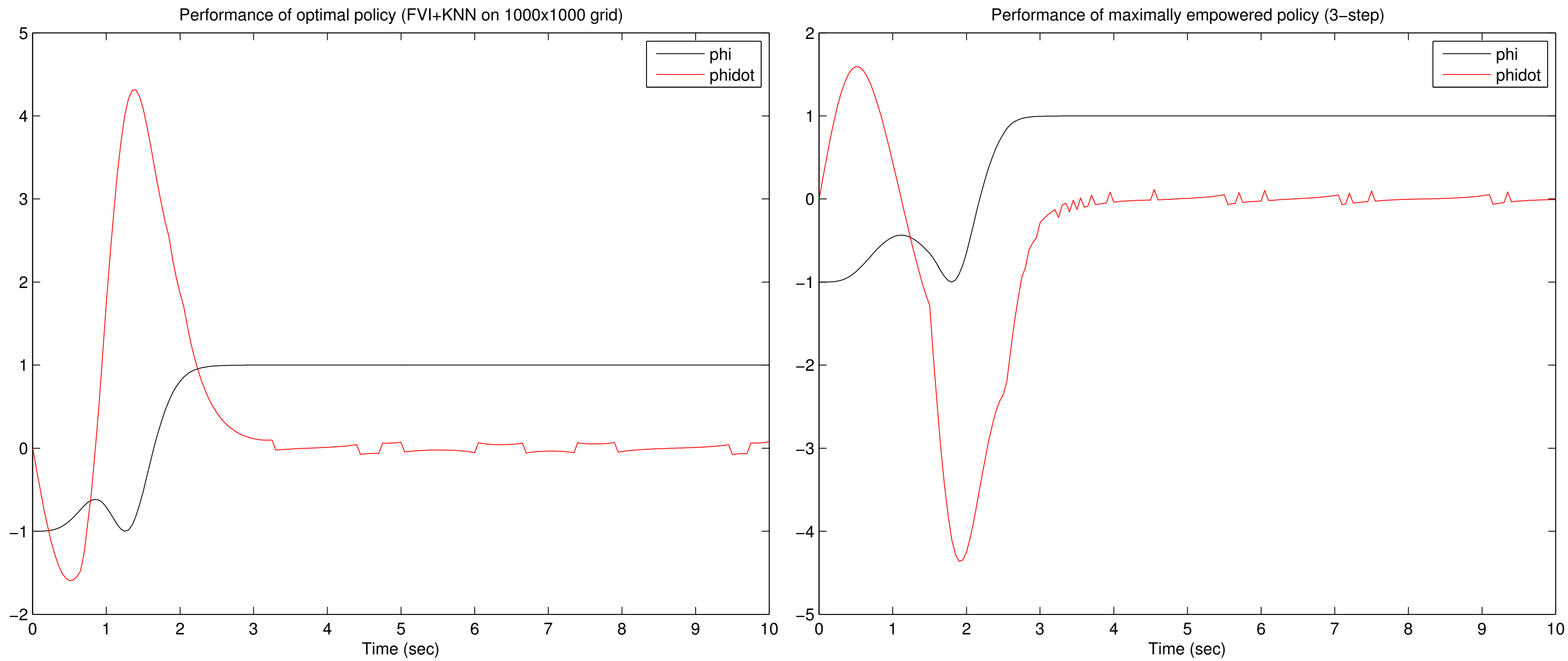
- **Empowerment-based control** (i.e. choosing in every state the action that leads to successor state with maximum empowerment) with
- **Optimal control** (optimal wrt time, i.e. choosing in every state the action that has the lowest cost-to-go for a quadratic cost function penalizing state transitions outside the goal). Optimal control problem is solved by approximate dynamic programming on a high-resolution grid.

Results: empowerment vs. optimal value function



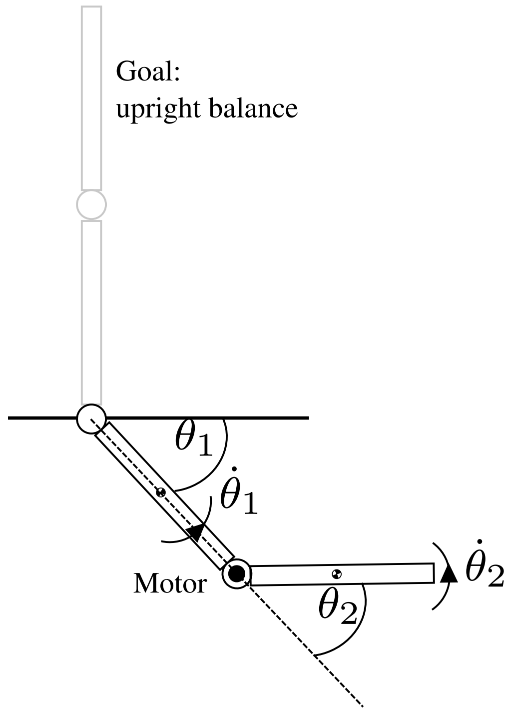
Left: Optimal value function V^* for the pedulum domain, computed with fitted value iteration (using the true state transition function) over a 1000×1000 grid. Right: 3-step Empowerment for the pendulum domain, evaluated for every state on a 100×100 grid (using the learned model). Note how the functions in the respective plots measure two completely different things, yet the overall shape of the result is the same.

Results: performance



Phase plot of φ and $\dot{\varphi}$ when following the respective greedy policy from the last slide. Note that for φ, the y-axis shows the height of the pendulum (+1 means upright, the goal state).

Experiment #2: acrobot inverted balance



Dynamics: see [Spong 95]

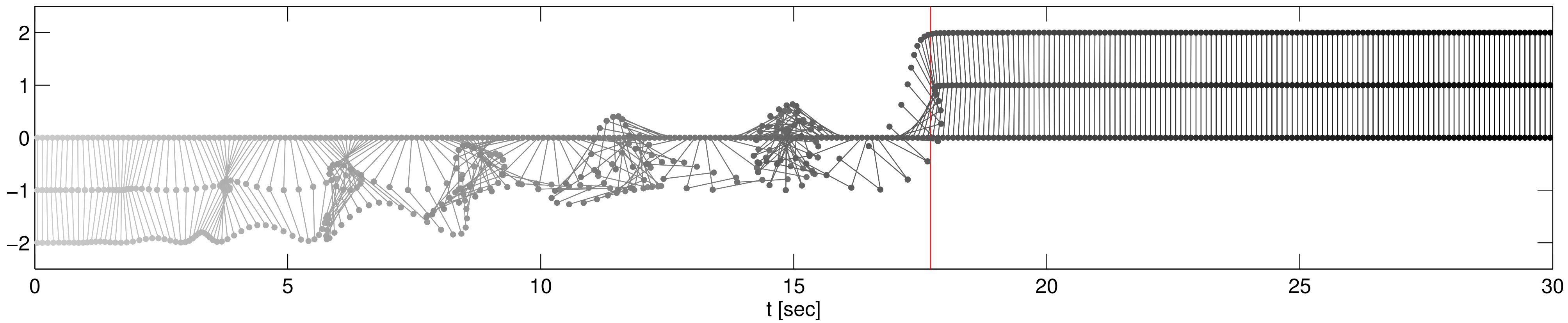
Goal: to give this system *some* purpose, we consider the **inverted balance task**.

Experiment: (same as before)

- **Empowerment-based control** (i.e. choosing in every state the action that leads to successor state with maximum empowerment)

Notice: this is a fairly difficult problem and RL typically only attempts the easier “swing-up” task

Experiment #2: results



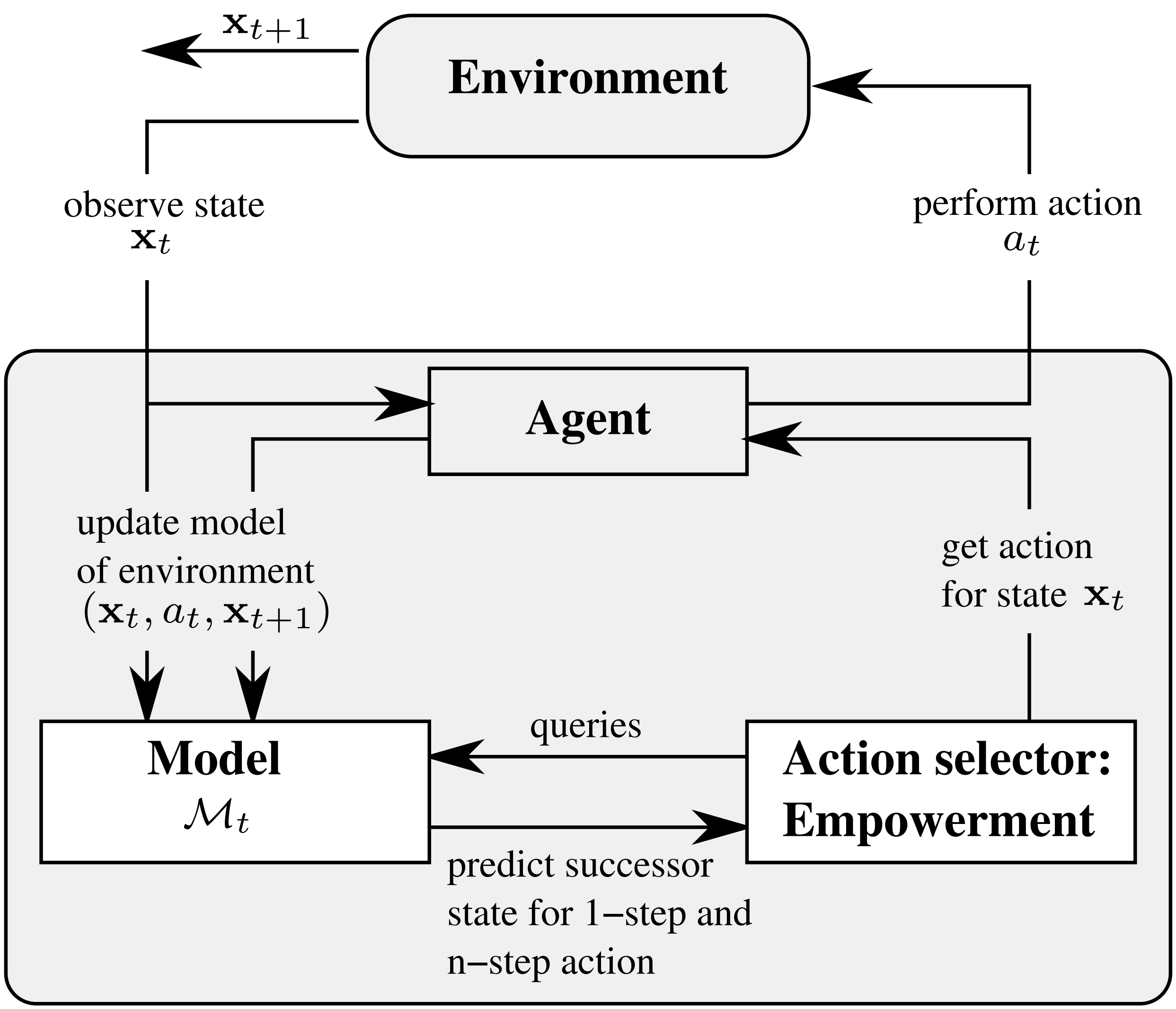
Note that we added a non-primitive "balance" action to allow stabilizing. Bang-bang alone is not sufficient.

Experiment #3: exploration + model learning

- **Up to this point** we have used the true transitions dynamics in the empowerment calculations.
- Question: what happens if the dynamics is not known in advance?
- **Now:** combine and interleave empowerment with online model learning, thus

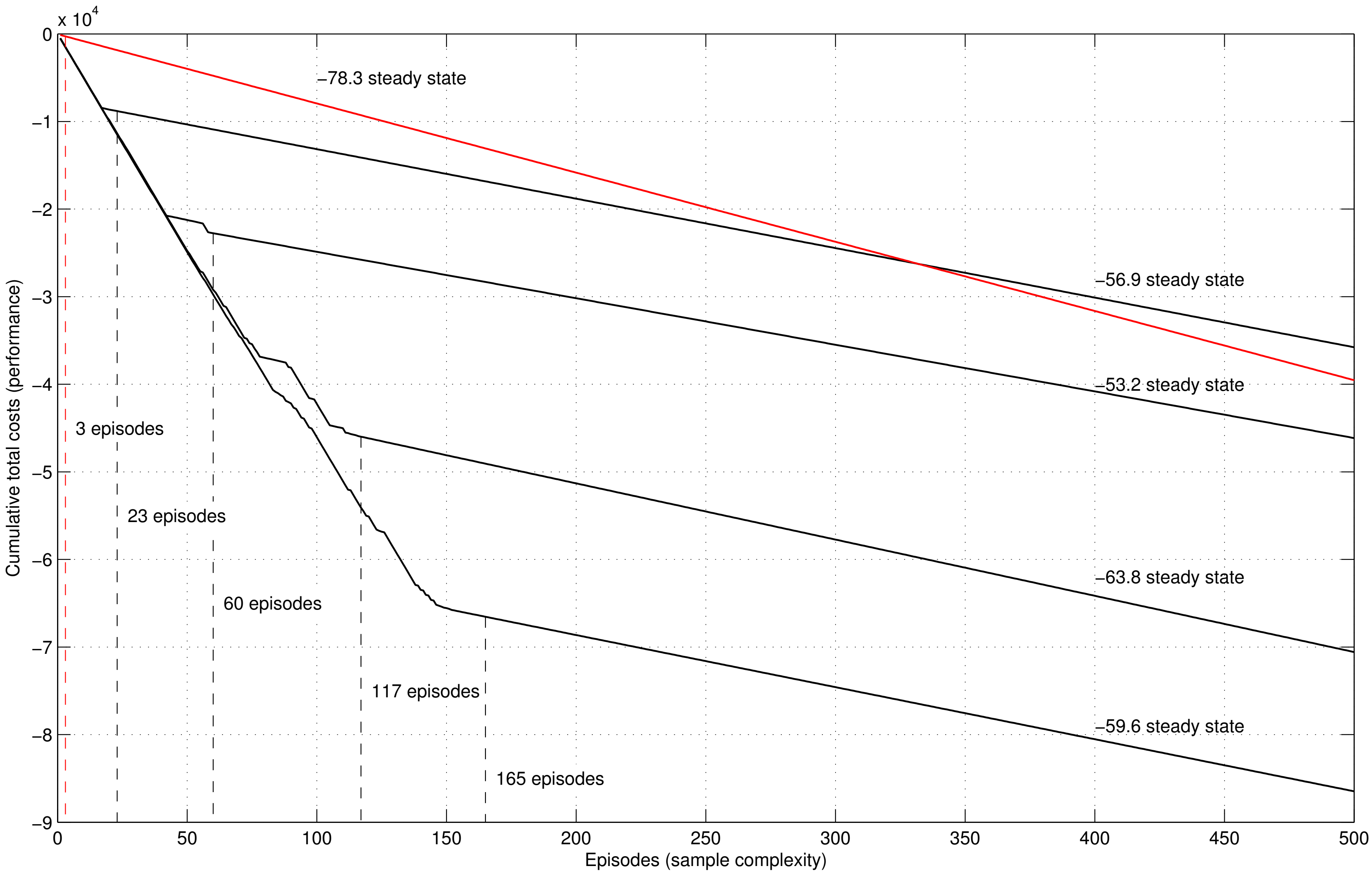
using empowerment to drive the exploration of the environment

Experiment #3: framework

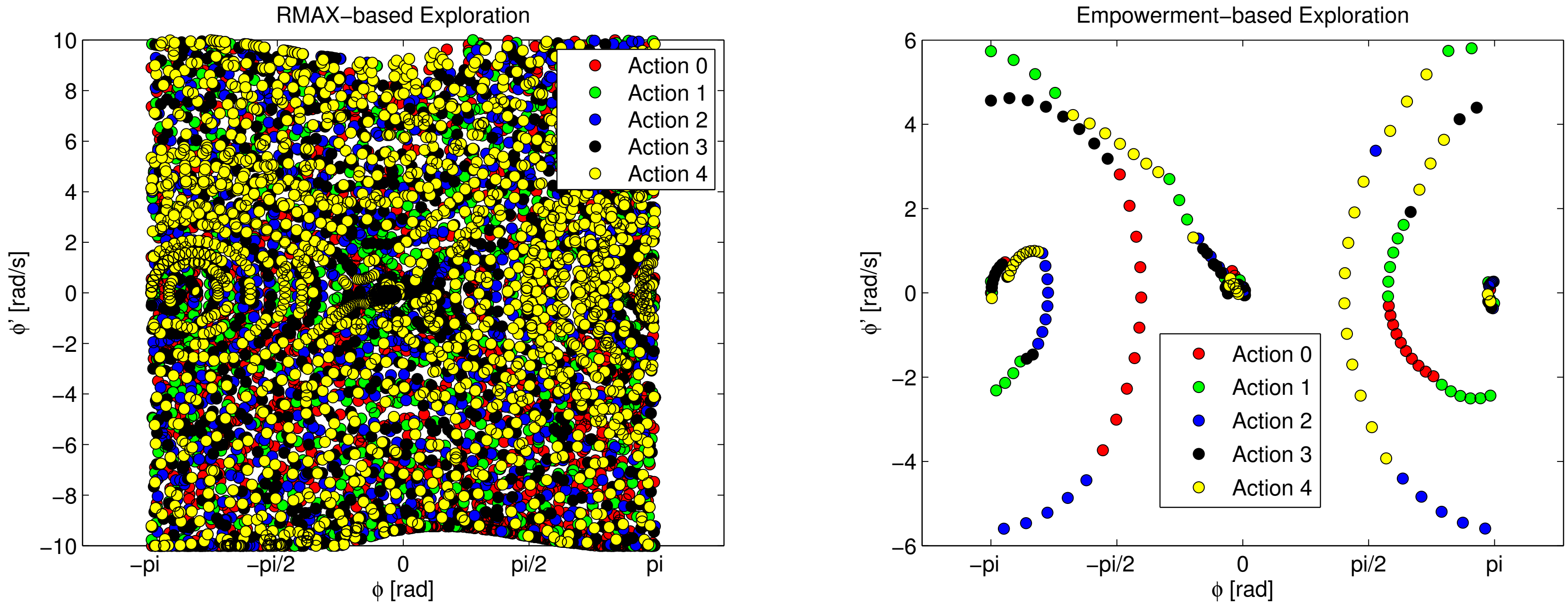


Experiment #3: empowerment vs. RMAX

Compare: RMAX with empowerment in the inverted pendulum domain:



Experiment #3: which states were visited?



(a) RMAX (b) Empowerment

Discussion

Results:

- One
- Two
- Three

Open technical questions:

- Is there a more efficient way to approximately compute empowerment?
- How can we deal with a larger number of n-step actions? Right now we use all possible sequences of length n, such that computational complexity scales with $|A|^{2n}$, where $|A|$ is number of 1-step actions.
- How can we deal with continuous actions?

Open conceptual questions:

- How can we benefit from empowerment? Why should we care about computing it?

Further reading

References:

T. Jung, D. Polani, and P. Stone. Empowerment for continuous agent-environment systems. Adaptive Behavior 19(1), pp. 16-39, 2011

Acknowledgements:

This research was partially funded by the European Commission as part of the FEELIX GROWING project under contract FP6 IST-045169.