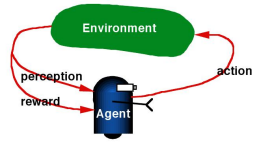
AIPOL: Anti Imitation-based Policy Learning

Michèle Sebag, Riad Akrour, Basile Mayeur, Marc Schoenauer

TAO, CNRS – INRIA – LRI, UPSud, Université Paris-Saclay, France

ECML PKDD 2016, Riva della Garda

Reinforcement Learning



The ultimate challenge

- Learning improves **survival expectation**

RL and the value function

- State space $\mathcal{S}$, action space $\mathcal{A}$
- transition $p(s, a, s')$
- reward function $R : \mathcal{S} \mapsto \mathbb{R}$
- policy $\pi : \mathcal{S} \mapsto \mathcal{A}$

For each π, define **reward expectation**

$$V_\pi(s) = R(s) + \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}) | s_0 = s, s_{t+1} \sim p(s_t, a_t = \pi(s_t), \cdot)\right]$$

1: Do we really need the value function ?

YES Bellman optimality equation

$$V^*(s) = max_\pi V_\pi(s)$$

$$Q^*(s,a) = R(s) + \gamma \mathbb{E}_{s' \sim p(s,a,\cdot)} V^*(s')$$

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

NO

Learning value function

$$Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$$

more complex than learning policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

Value function and Energy-based learning

Le Cun et al., 2006

Goal: Learn

$$h : \mathcal{X} \mapsto \mathcal{Y} \quad \text{e.g. } \mathcal{Y} \text{ structured}$$

Energy-based Learning

1. Learn

$$g : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R} \quad s.t.\ g(x, y_x) > g(x, y') \text{ for } y' \neq y_x$$

2. Set

$$h(x) = \arg\max_y g(x, y)$$

EbL pros and cons

- – more complex
- \+ more robust

2: Which human expertise required for RL ?

	Agent learns	Human designs / yields
RL	Pol. π^*	$\mathcal{S}$, $\mathcal{A}$, R
Inverse RL [1]	Reward R + RL	(optimal) trajectories
Preference RL [2,3,4,5]	Pol. Return + DPS	ranked trajectories

[1] Abbeel, P.: Apprenticeship Learning and Reinforcement Learning PhD thesis 08
[2] Frnkrantz, J. et al.: Preference-based reinforcement learning. MLJ 12M
[3] Wilson et al.: A Bayesian Approach for Policy Learning from Trajectory Preference Queries. NIPS 12
[4] Jain et al.: Learning Trajectory Preferences for Manipulators via Iterative Improvement NIPS 13
[5] Akrour et al. Programming by Feedback, ICML 14

This talk

Relaxing expertise requirement

Expert only required to know what can go wrong

Counter-trajectories

$$\text{CD} =_{def} (s_1, \ldots s_T) \text{ s.t. } V^*(s_t) < V^*(s_{t+1})$$

with V^* the (unknown) optimal value function.

Example

- Take a bicycle in equilibrium s_1
- Apply a random policy
- Bicycle soon falls down... s_T

Anti-Imitation Policy Learning 1/3

Given counter trajectories

$\mathcal{E} = \{(s_{i,1}, \ldots s_{i,T_i}), i = 1 \ldots n\}$

Learn pseudo-value U^* s.t. $U^*(s_{i,t}) > U^*(s_{i,t+1})$

Formally

$$U^* = \arg\min\{\text{Loss}(U, \mathcal{E}) + \mathcal{R}(U)\}$$

with

- $\text{Loss}(U, \mathcal{E}) = \Sigma_i \Sigma_{t<t'} [U(s_{i,t'}) - U(s_{i,t}) + 1]_+$
- $R(U)$ a regularization term

If transition model is known, AiPOL policy:

$$\pi_{U^*}(s) = \arg\max_a \mathbb{E}_{s' \sim p(s,a,\cdot)} U^*(s')$$

Anti-Imitation Policy Learning 2/3

If transition model is unknown

1. Given U^* pseudo-value function
2. Given $\mathcal{G} = \{(s_i, a_i, s'_i), i = 1, m,\ s.t.\ U^*(s'_i) > U^*(s'_{i+1})\}$

Learn pseudo Q-value s.t. $Q^*(s_i, a_i) > Q^*(s_{i+1}, a_{i+1})$

Formally **Learning to rank**

$$Q^* = \arg\min\{\text{Loss}(Q, \mathcal{G}) + \mathcal{R}(Q)\}$$

with

- $\text{Loss}(Q, \mathcal{G}) = \Sigma_{i<j}[Q(s_j, a_j) - Q(s_i, a_i) + 1]_+$
- $R(U)$ a regularization term

AiPOL policy:

$$\pi_{Q^*}(s) = \arg\max_a Q^*(s, a)$$

Anti-Imitation Policy Learning 2/3

Proposition

If

i) V^* continuous on $\mathcal{S}$;

ii) U^* monotonous wrt V^* on $\mathcal{S}$

iii) with a margin between best and other actions

$$\forall a' \neq a = \pi_{U^*}(s), \mathbb{E}U^*(s'_{s,a}) > \mathbb{E}U^*(s'_{s,a'}) + M$$

iv) U^* Lipschitz with constant M;

v) transition model β-sub-Gaussian:

$$\forall t \in \mathbb{R}^+, \mathbb{P}(||\mathbb{E}s'_{s,a} - s'_{s,a}||_2 > t) < 2e^{-\beta t^2}$$

Then

if $2L < M\beta$, π_{U^*} is an optimal policy

Experimental validation

Goals of experiment

- How many CDs?
- How much expertise in generating CDs ? (starting state, controller)

Experimental setting

	Mountain	Bicycle	Pendulum
# CD	1	20	1
length CD	1,000	5	1,000
starting state	target st	random	target st.
controller	neutral	random	neutral

Experimental setting, 2

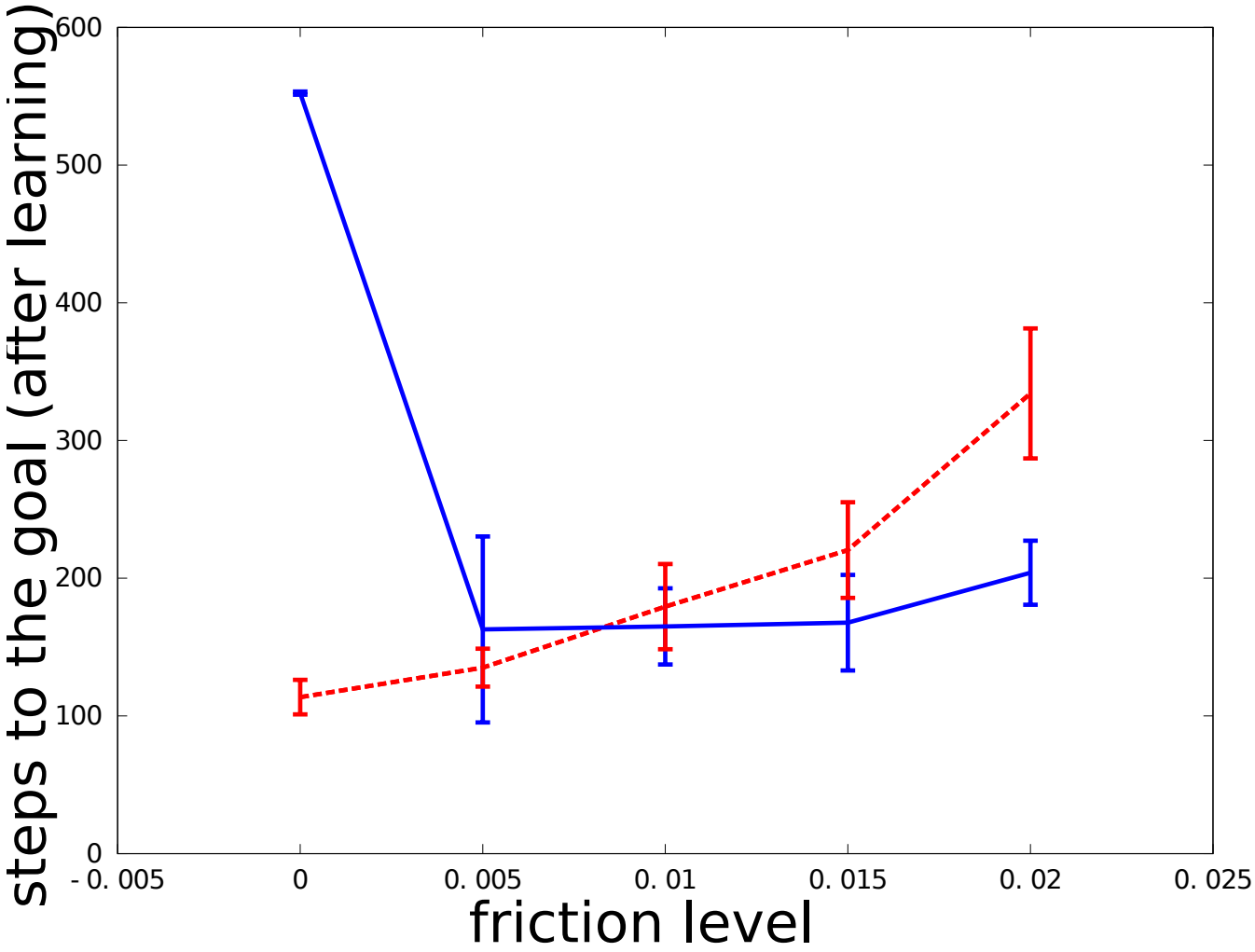
Learning to rank

Ranking SVM with Gaussian kernel — Joachims 06

		Mountain	Bicycle	Pendulum
U^*	C_1	10^3	10^3	10^{-5}
	$1/\sigma_1^2$	10^{-3}	10^{-3}	.5
Q^*	nb const	500	5,000	–
	C_2	10^3	10^3	–
	$1/\sigma_2^2$	10^{-3}	10^{-3}	–

Mountain Car, 1/3

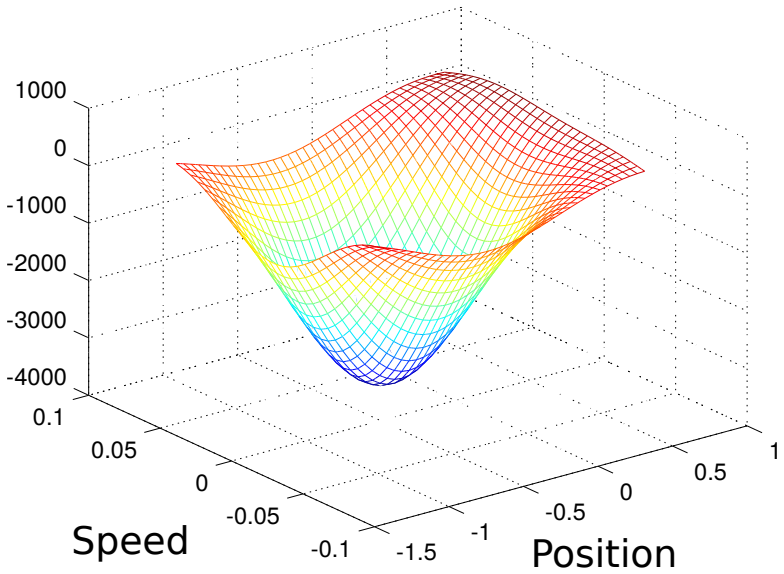
AiPOL vs **SARSA** depending on the friction

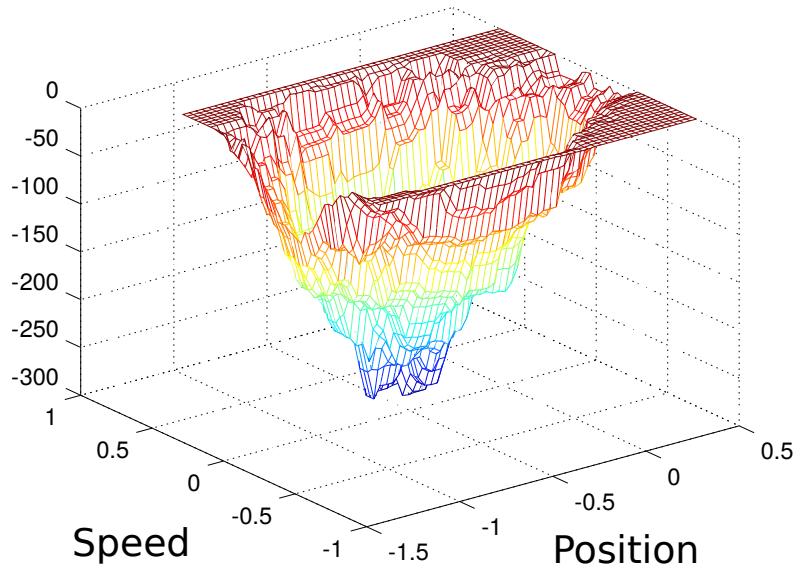


Mountain car (20 runs)

Mountain Car, 2/3

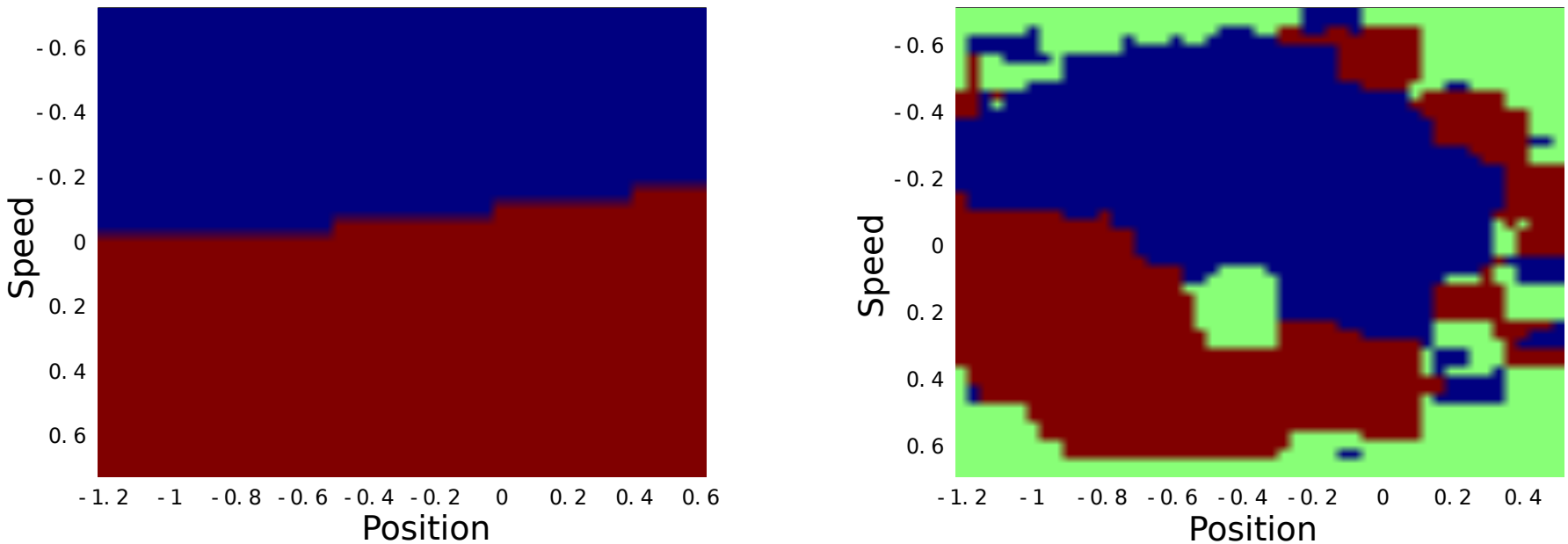
AiPOL pseudo-value vs **SARSA value (1,000 iter)**





Mountain Car, 3/3

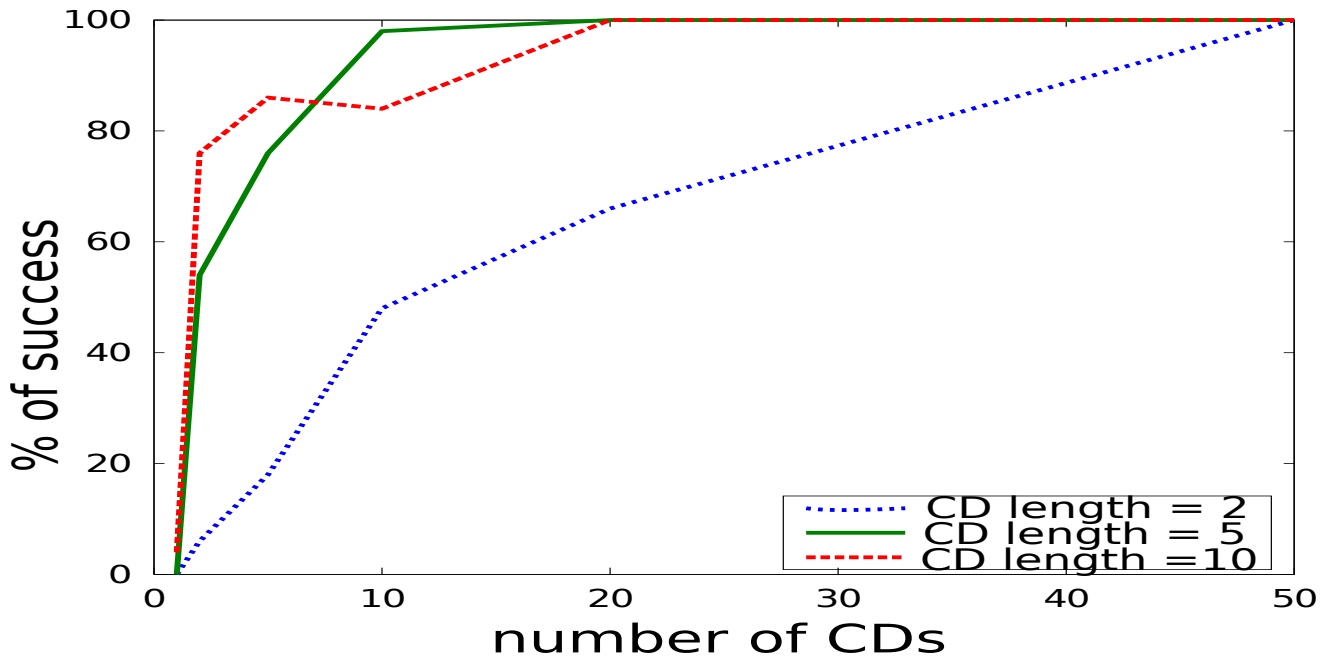
AiPOL policy vs **SARSA policy**



Action: forward, backward, neutral.

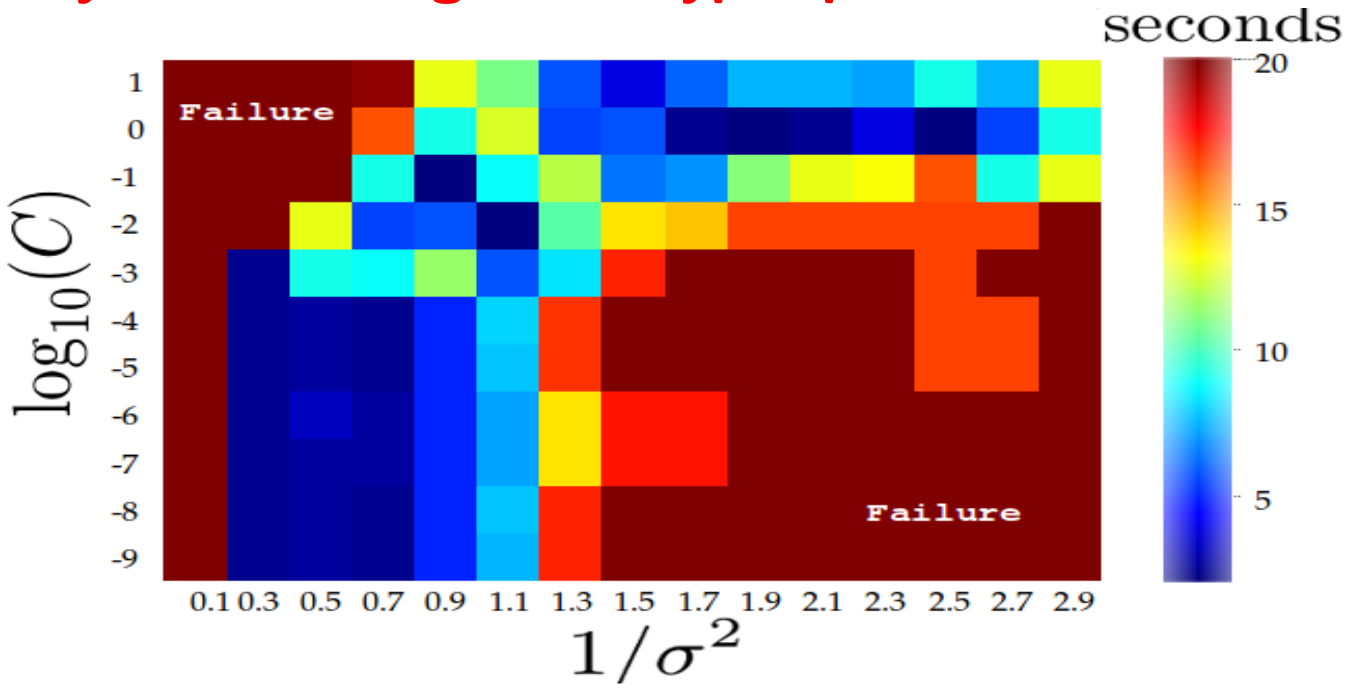
Bicycle

Sensitivity wrt number and length of CDs



Inverted Pendulum

Sensitivity wrt Ranking-SVM hyper-parameters



Interpretation

- kernel width too small, no generalization (doesn't reach the top)
- too large, U^* imprecise (goes to the top and falls on the other side)

AiPOL: Discussion

Pro

- Compared to Inverse RL, AiPOL involves relaxed expertise requirements, with lesser computational requirements (greedification as opposed to RL)

Limitations

- Latency of transitions: (e.g. bicycle) $(s_i, a_i, s'_i, a'_i, s\text{“}_i)$ $Q(s_i, a_i) > Q(s_j, a_j)$ if $U^*(s''_i) > U^*(s\text{“}_j)$
- Cost of learning Q^* quadratic in number of triplets.

Further work

- Non reversible MDP needs be addressed.