Preference-Based Policy Iteration

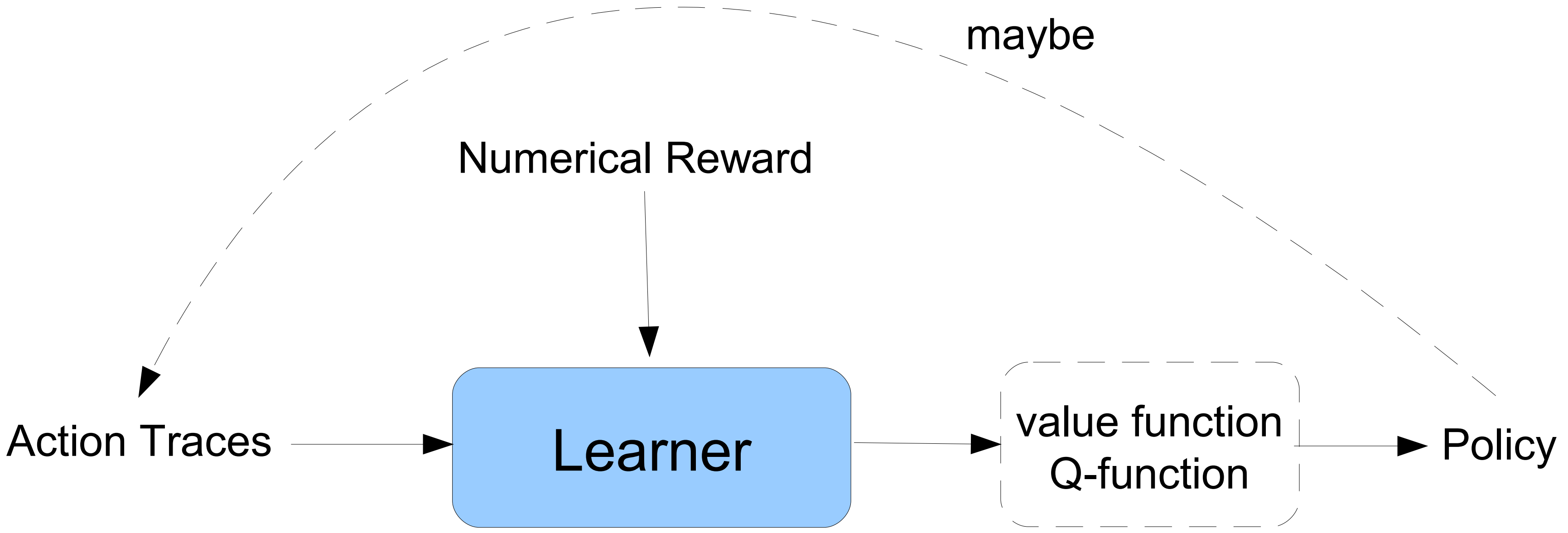
Leveraging Preference Learning for Reinforcement Learning

Weiwei Cheng **Johannes Fürnkranz**

Eyke Hüllermeier **Sang-Hyeun Park**

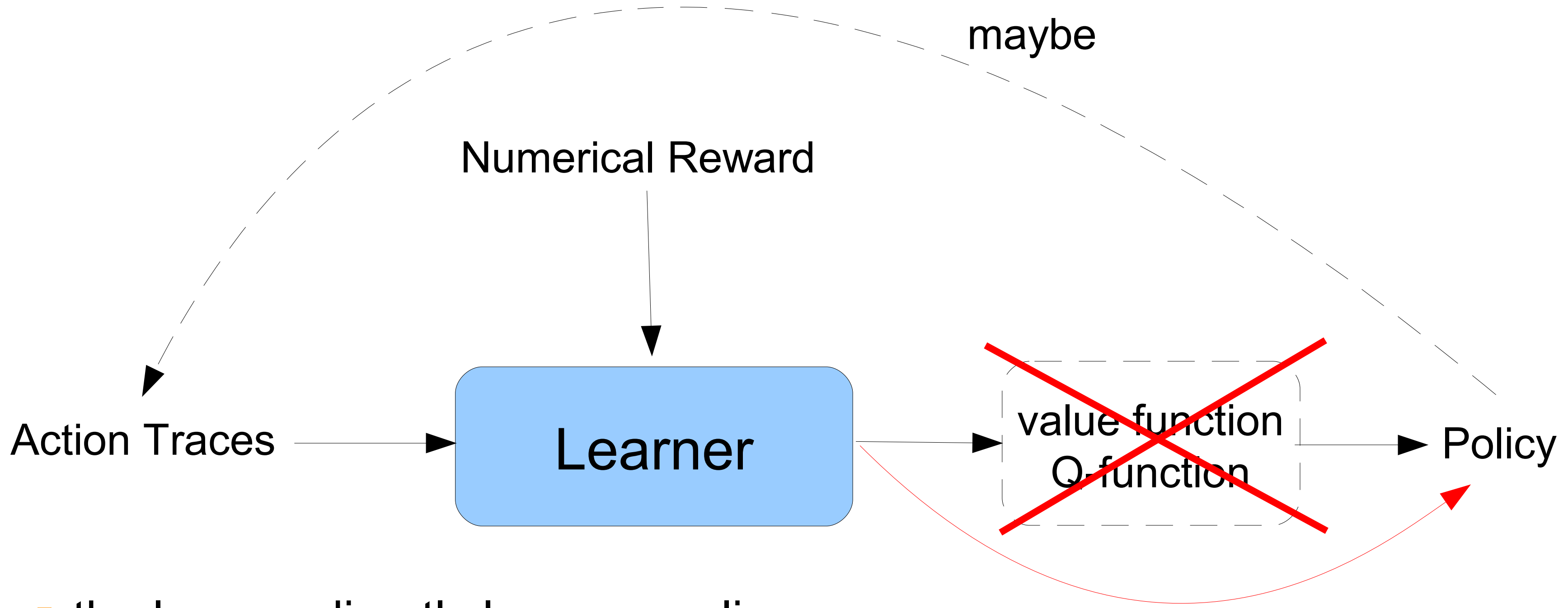
Philipps-Universität Marburg TU Darmstadt

Classical Reinforcement Learning



- the learner produces a function which estimates the value of states or state/action pairs
  - e.g., Q-learning, TD(λ), ...
- the policy uses this function for making actions
  - e.g. greedy or ε-greedy policies

Policy learning



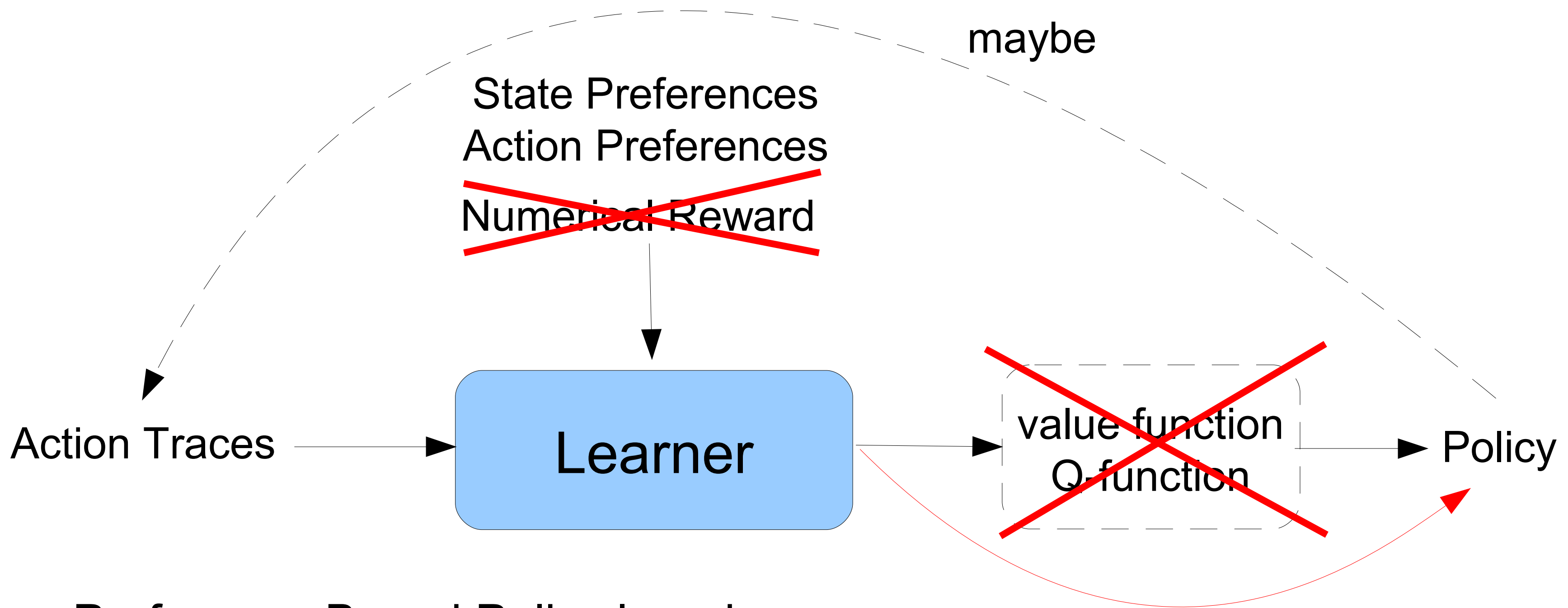
- the learner directly learns a policy
  - *actor-critic methods* learn both a value function (critic) and a policy (actor)
  - *policy gradient methods* search in the space of parametrized policies
    - e.g., a policy is a linear function that maps a state description to continuous actions
- estimation of expected reward may not be necessary

Numerical Reward is not needed

- Numerical Reward is not directly needed for learning
  - it is only needed for determining the best action in a given state
- Numerical Reward is not directly needed for acting
  - the learned policy is directly computed from the state description
  - no estimation of expected reward
- Instead we could have an oracle giving the best action in each state (→ supervised learning)
- But the best action in a state may be
  - unknown
  - infeasible to determine

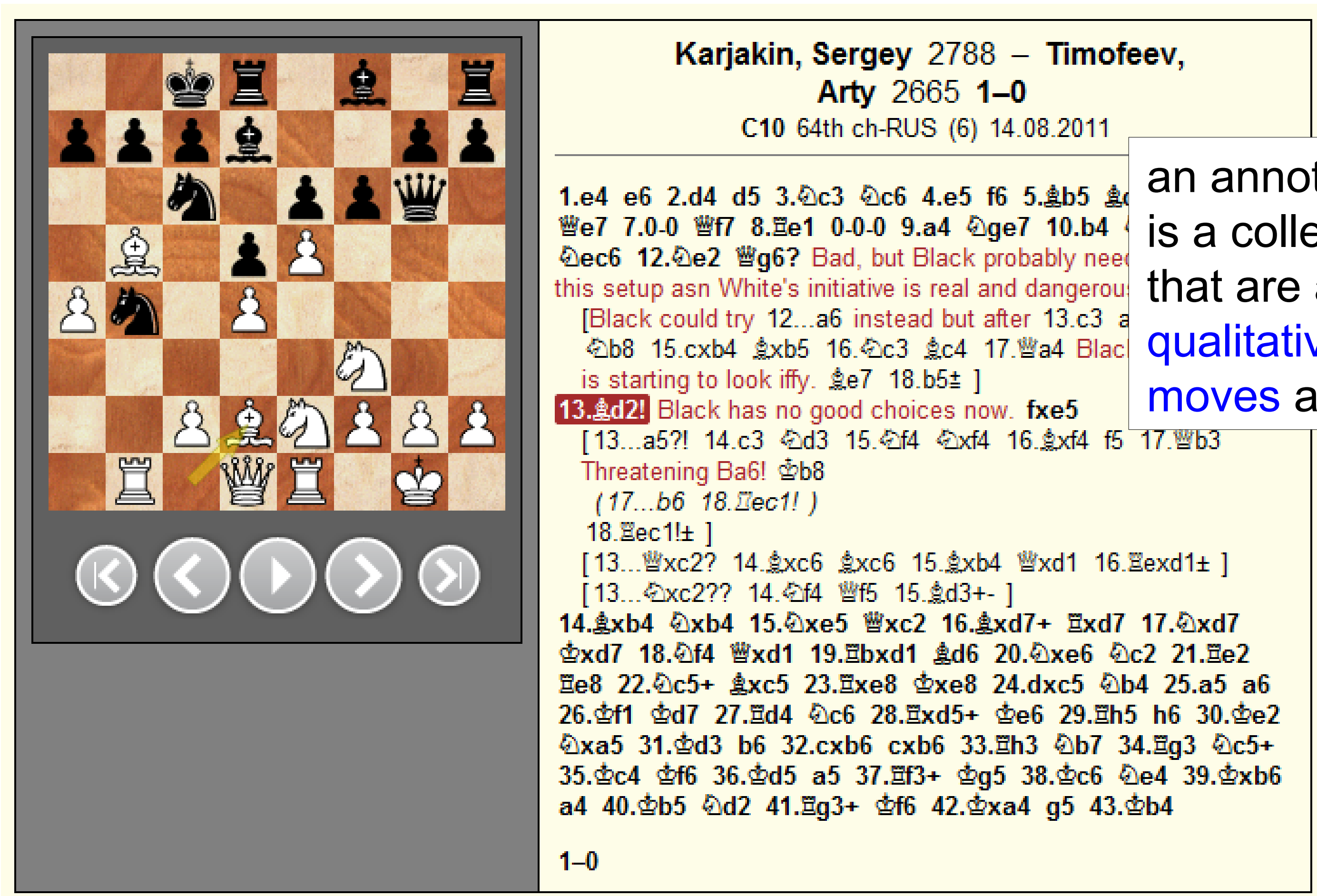
  (essentially it requires to sample the optimal policy)

Vision: Preference-Based Reinforcement Learning



- Preference-Based Policy learning:
  - the policy function is a label ranker that ranks all actions in a given state
  - we know their order (best to last) but not their value
- Training information:
  - Action preferences and State preferences

Example: Annotated Chess Games



an annotated chess game is a collection of trajectories that are annotated with qualitative rewards for moves and states

Example: Annotated Chess Games

TECHNISCHE UNIVERSITÄT DARMSTADT

Karjakin, Sergey 2788 – **Timofeev, Arty** 2665 **1–0**
C10 64th ch-RUS (6) 14.08.2011

1.e4 e6 2.d4 d5 3.♘c3 ♘c6 4.e5 f6 5.♗b5 ♗d7 6.♘f3 ♕e7 7.0-0 ♕f7 8.♖e1 0-0-0 9.a4 ♘ge7 10.b4 ♘xb4 11.♖b1 ♘ec6 12.♘e2 ♕g6? Bad, but Black probably needs to rethink this setup asn White's initiative is real and dangerous anyhow.

[Black could try 12...a6 instead but after 13.c3 axb5 14.axb5 ♘b8 15.cxb4 ♗xb5 16.♘c3 ♗c4 17.♕a4 Black's king safety is starting to look iffy. ♗e7 18.b5±]

13.♗d2! Black has no good choices now. **fxe5**

[13...a5?! 14.c3 ♘d3 15.♘f4 ♘xf4 16.♗xf4 f5 17.♕b3 Threatening Ba6! ♔b8
(17...b6 18.♖ec1!)
18.♖ec1±]

[13...♕xc2? 14.♗xc6 ♗xc6 15.♗xb4 ♕xd1 16.♖exd1±]

[13...♘xc2?? 14.♘f4 ♕f5 15.♗d3+-]

14.♗xb4 ♘xb4 15.♘xe5 ♕xc2 16.♗xd7+ ♖xd7 17.♘xd7 ♔xd7 18.♘f4 ♕xd1 19.♖bxd1 ♗d6 20.♘xe6 ♘c2 21.♖e2 ♖e8 22.♘c5+ ♗xc5 23.♖xe8 ♔xe8 24.dxc5 ♘b4 25.a5 a6 26.♔f1 ♔d7 27.♖d4 ♘c6 28.♖xd5+ ♔e6 29.♖h5 h6 30.♔e2 ♘xa5 31.♔d3 b6 32.cxb6 cxb6 33.♖h3 ♘b7 34.♖g3 ♘c5+ 35.♔c4 ♔f6 36.♔d5 a5 37.♖f3+ ♔g5 38.♔c6 ♘e4 39.♔xb6 a4 40.♔b5 ♘d2 41.♖g3+ ♔f6 42.♔xa4 g5 43.♔b4

1–0

- it is hard to give an exact reward signal for a move
- it is easier to specify which of two moves is better

→ ***Action Preferences***

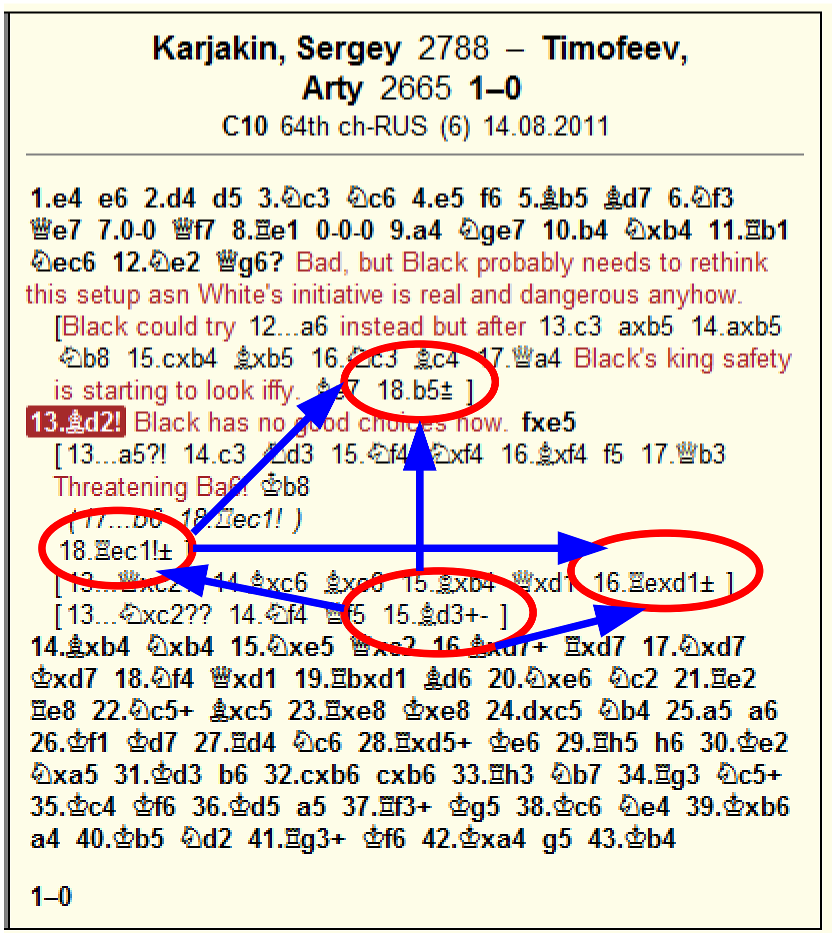
$$(!! \succ\ !\ \succ\ !? \succ\ ?! \succ\ ? \succ\ ??)$$

13th move for black:

fxe5 ≻ a5 ≻ ♕xc2 ≻ ♘xc2

Example: Annotated Chess Games

TECHNISCHE UNIVERSITÄT DARMSTADT



Karjakin, Sergey 2788 – **Timofeev, Arty** 2665 **1–0**
C10 64th ch-RUS (6) 14.08.2011

1.e4 e6 2.d4 d5 3.♘c3 ♘c6 4.e5 f6 5.♗b5 ♗d7 6.♘f3 ♕e7 7.0-0 ♕f7 8.♖e1 0-0-0 9.a4 ♘ge7 10.b4 ♘xb4 11.♖b1 ♘ec6 12.♘e2 ♕g6? Bad, but Black probably needs to rethink this setup asn White's initiative is real and dangerous anyhow.
[Black could try 12...a6 instead but after 13.c3 axb5 14.axb5 ♘b8 15.cxb4 ♗xb5 16.♘c3 ♗c4 17.♕a4 Black's king safety is starting to look iffy. ♔e7 18.b5±]
13.♗d2! Black has no good choices now. **fxe5**
[13...a5?! 14.c3 ♘d3 15.♘f4 ♘xf4 16.♗xf4 f5 17.♕b3 Threatening Ba6! ♔b8
(17...b6 18.♖ec1!)
18.♖ec1!±]
[13...♕xc2?! 14.♗xc6 ♗xc6 15.♗xb4 ♕xd1 16.♖exd1±]
[13...♘xc2?? 14.♘f4 ♕f5 15.♗d3+-]
14.♗xb4 ♘xb4 15.♘xe5 ♕xe2 16.♗xd7+ ♖xd7 17.♘xd7 ♔xd7 18.♘f4 ♕xd1 19.♖bxd1 ♗d6 20.♘xe6 ♘c2 21.♖e2 ♖e8 22.♘c5+ ♗xc5 23.♖xe8 ♔xe8 24.dxc5 ♘b4 25.a5 a6 26.♔f1 ♔d7 27.♖d4 ♘c6 28.♖xd5+ ♔e6 29.♖h5 h6 30.♔e2 ♘xa5 31.♔d3 b6 32.cxb6 cxb6 33.♖h3 ♘b7 34.♖g3 ♘c5+ 35.♔c4 ♔f6 36.♔d5 a5 37.♖f3+ ♔g5 38.♔c6 ♘e4 39.♔xb6 a4 40.♔b5 ♘d2 41.♖g3+ ♔f6 42.♔xa4 g5 43.♔b4

1–0

- it is hard to give an exact reward signal for a move
- it is easier to specify which of two moves is better

→ ***Action Preferences***

$$(!! > ! > !? > ?! > ? > ??)$$

- it is hard to give an exact numerical score for a position
- it is easier to give a qualitative evaluation for a position

→ ***State Preferences***

$$(+- > \pm > ⩲ > ⩱ > \mp > -+)$$

Approximate Policy Iteration with Roll-Outs
(Lagoudakis & Parr, ICML-03)

- Assumption:
  - we have a generative model of the underlying Markov process
  - we can use this model for sampling action traces and reward signals

  → we can perform *roll-outs* (generate action traces / trajectories)

Roll-Out

- Estimate the value $Q^{\pi}(s,a)$ for performing action a in state s and following policy π thereafter
- by performing the action and then repeatedly following the policy for at most T steps
- and returning the average of the observed rewards

- and use these roll-outs for training a policy...

Approximate Policy Iteration with Roll-Outs
(Lagoudakis & Parr, ICML-03)

- Key idea:
  - determine the best action in each state
  - train a conventional classifier (e.g., decision tree) as a policy

API

1. start with policy π_0
2. for each state s
   - evaluate all actions with **Roll-Out**
   - determine the best action a^* (the one with highest estimated Q-value)
   - generate a training example (s,a^*) if a^* is significantly better than all other actions in state s
3. use all training examples to train a policy $\pi: S \rightarrow A$
4. goto 2. (until stop)

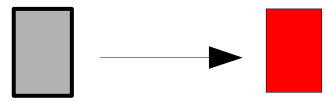
Classifier

Label Ranking

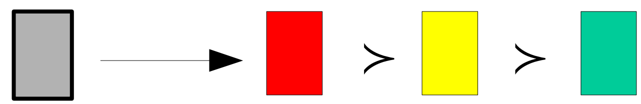
(e.g., Hüllermeier, Fürnkranz, Cheng, Brinker, AIJ 2008)

The task in label ranking is to order a set of labels

- **Classification:**
  - pick one of a set of items



- **(Label) Preference Learning:**
  - predict a (partial or total) order $\Pi(A)$ relation on a set of items A



Label rankers can be trained with label preferences

- In our case we want to rank all actions based on the state description
- trained on action preferences of the type $(s, a_i \succ a_j)$

Preference-Based Policy Iteration

TECHNISCHE UNIVERSITÄT DARMSTADT

- Key idea:
  - compute preferences between pairs of actions
  - train a label ranker as a policy

PBPI

1. start with policy π_0

2. for each state s
   - evaluate all actions with **Roll-Out**
   - for all action pairs (a_i, a_j) determine if a_i is significantly better than a_j
   - generate a training example $(s, a_i \succ a_j)$ if it is
- use all training examples to train a policy $\pi: S \to \Pi(A)$

1\. goto 2. (until stop)

Label Ranker

Advantages of a preference-based framework

TECHNISCHE UNIVERSITÄT DARMSTADT

- Often there is no natural numerical value
  - a preference-based formulation allows to deal with qualitative feedback
- It is difficult to optimize multiple objectives
  - a preference-based framework allows to flexibly define preferences over states according to multiple criteria (e.g., Pareto dominance)
- It may impossible or infeasible to determine the best action
  - but it is often easier to compare two actions
  - in the case of roll-outs:

a_1 is not significantly better than a_2
→ no training example for API

but we know $a_1 \succ a_3$ and $a_2 \succ a_3$
→ 2 training examples for PBPI

Case Study 1
Learning from Action Preferences

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithms: each using a Neural Network as a base classifier

- **API**: Approximate Policy Iteration (Lagoudakis & Parr, ICML-03)
  - uses roll-outs to determine the best action
- **PAPI**: Pairwise Approximate Policy Iteration
  - uses all preferences that involve the best action (pairwise classification)
- **PBPI**: Preference-Based Policy Iteration
  - uses all preferences (also those involving suboptimal actions)

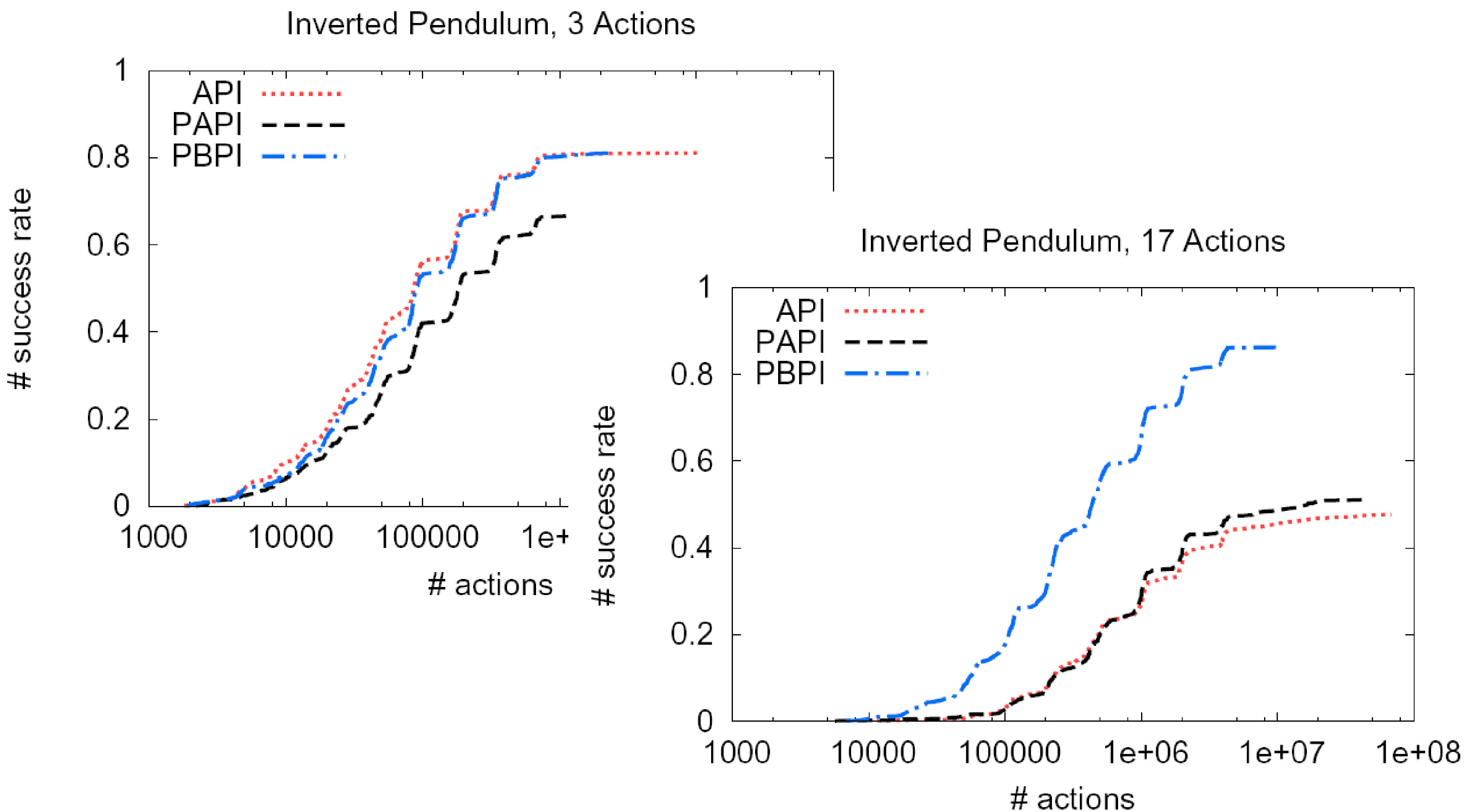
Domains: Standard RL benchmarks, each with 3, 5, 9, 17 actions

- Inverted Pendulum
- Mountain Car

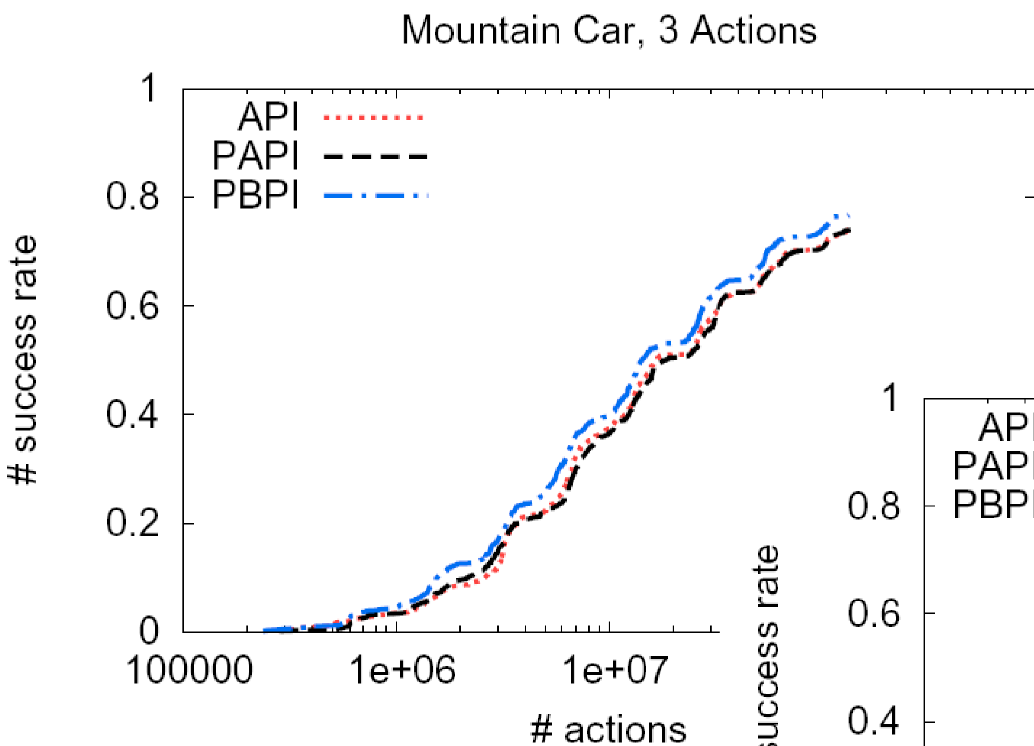
Evaluation: following (Lagoudakis & Parr, ICML-03)

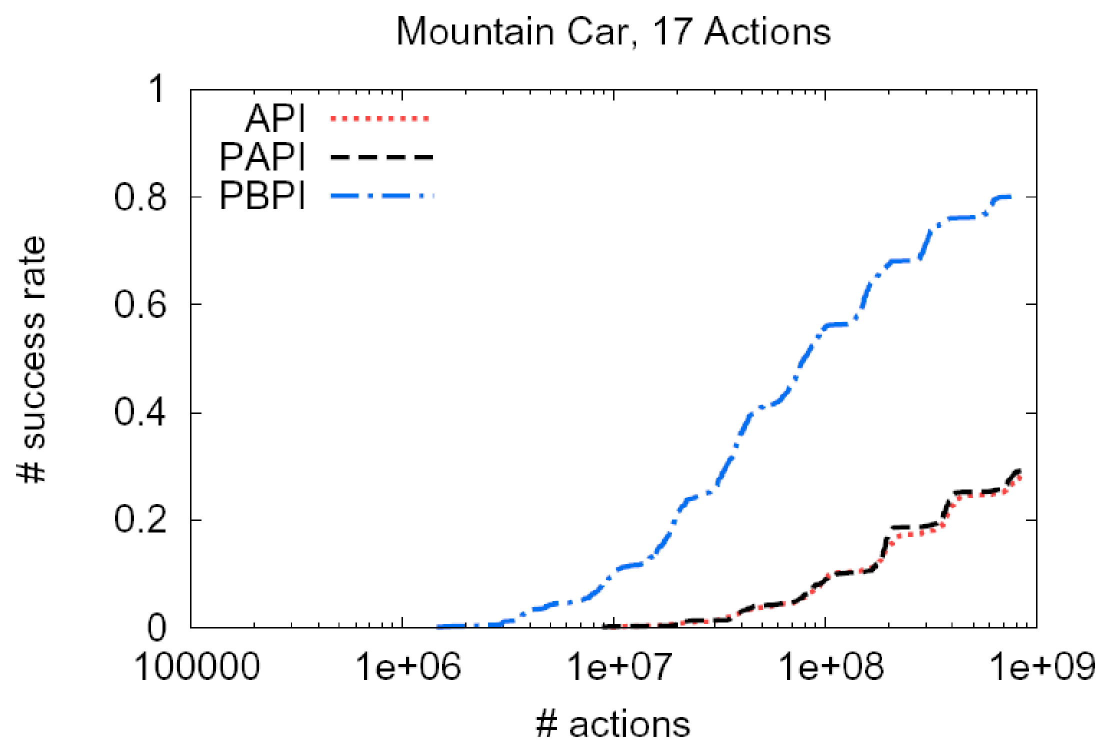
- try a variety of different parametrizations (starting states etc.)
- run each until successful or at most 10 policy iterations
- plot cumulative distribution of success rate over total number of actions taken to reach this success rate

Results: Inverted Pendulum

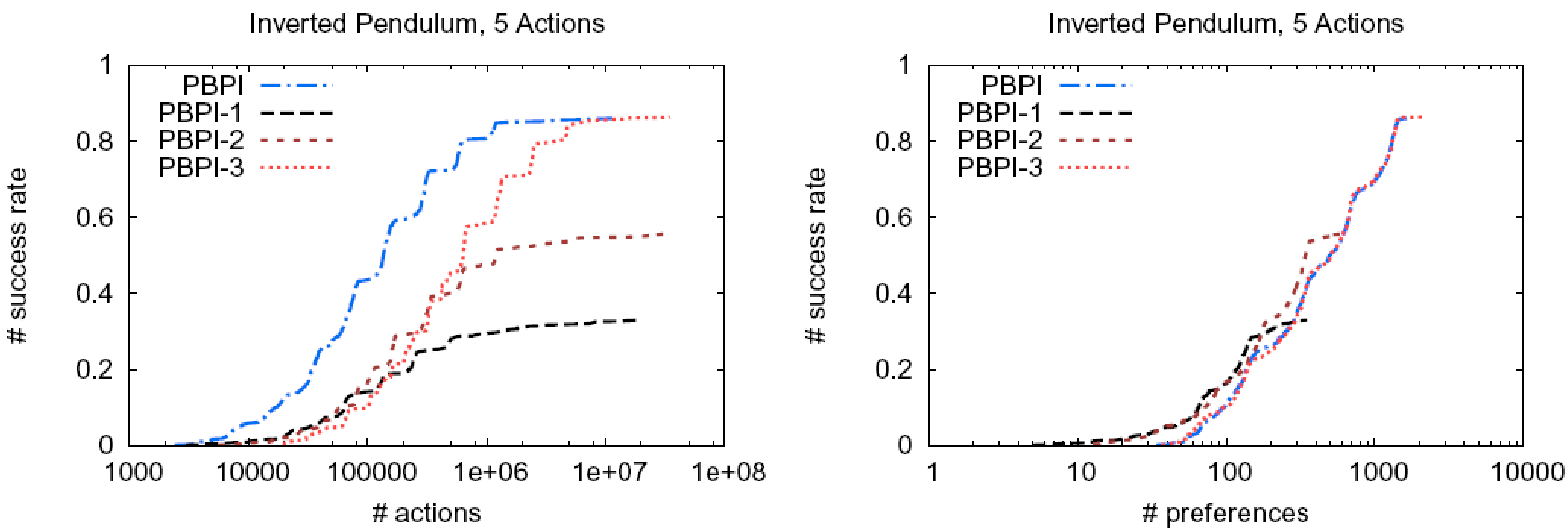


Results: Mountain Car





Complete vs. Partial State Evaluation



In each case PBI-i does only generate one preference per state

- PBI-1: visits the same number of states as PBI
- PBI-2: visits k/2 as many states (2 roll-outs vs. k roll-outs)
- PBI-3: visits k(k-1)/2 as many states (generates the same #preferences)

Case Study 2
Learning from Qualitative Feedback

Domain: Clinical trials of cancer treatment (Zhao et al. 2009)

- the goal is to devise a treatment policy for cancer patients
- action is the amount of medication that the patient is given

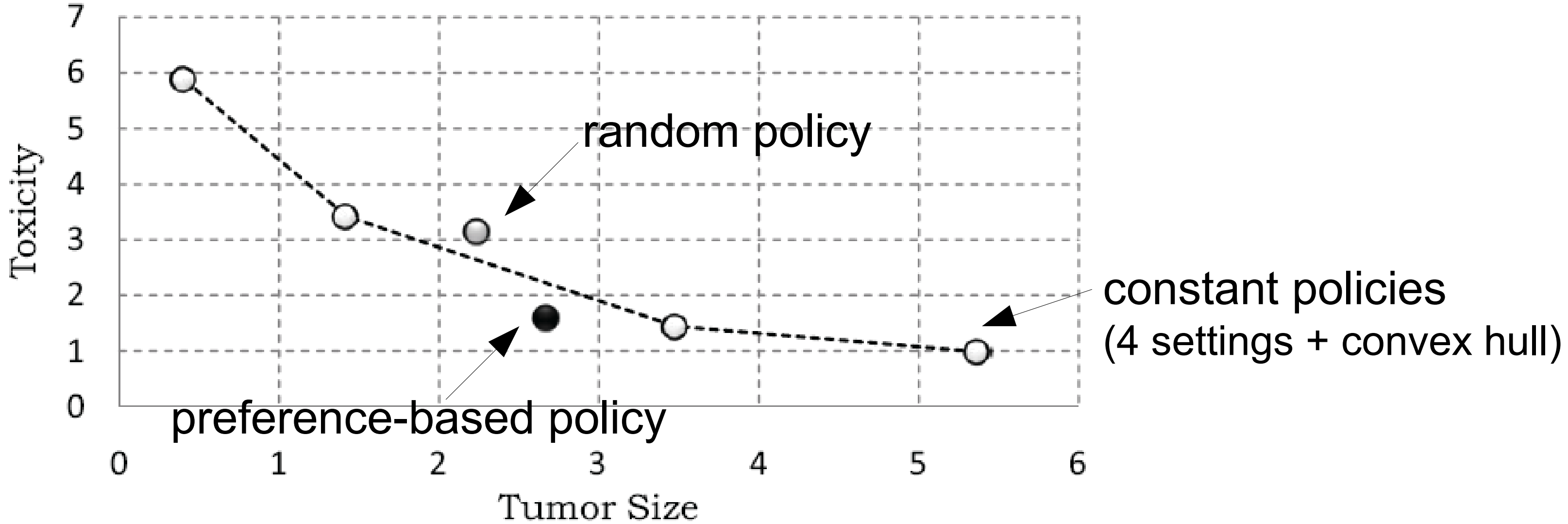
Characteristics:

- Numerical reward functions are artificial
  - The death of a patient is worse than all other results but cannot be given a reasonable number
- Multi-Objective definition of state preferences (Pareto-dominance)

Treatment A is better than Treatment B if

- *at every time point, the patient treated with A feels better than the patient treated with B and*
- *the patient treated with A is more healthy than patient B at the end*

Case Study 2
Learning from Qualitative Feedback



Conclusions

- First step towards a framework that lifts conventional reinforcement learning into a qualitative setting
  - where reward is not absolute but relative in comparison to alternatives
- We proposed a preference-based extension of approximate policy iteration
  - which we evaluated on 2 case studies
- Case Study 1 demonstrated the utility of using additional preferences
  - a label ranker can use more information and produce better results than a classifier
- Case Study 2 demonstrated an application where
  - numerical reward signals are somewhat artificial and
  - multiple objectives can be formulated in the form of preferences

Conclusions

- Formulated the problem of reinforcement learning from non-numeric (qualitative) rewards
  - Preferences are a natural way of formulating qualitative feedback
- First results
  - training on all action preferences in each state yields better results than using only the best action (makes better use of information)
  - proof of concept that learning in a domain where numeric feedback is not available works

Open Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

- How can we unify state and action preferences?
  - Key idea: Preferences over trajectories
- How can we integrate (qualitative) preference information and (quantitative) reward signals?
- How can we integrate off-line experience (annotated games) with on-line experience?
- Is there an on-line version of preference-based RL?
- Can we back up rankings of actions between states? What if we don't have a generative model?
- Can we really do this for chess?

While you ask questions...

Special issue of *Machine Learning* on **Preference Learning**

Editors: Eyke Hüllermeier and Johannes Fürnkranz

Submission Deadline: **December 31, 2011**