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(\(\lambda\)), ...
- The policy uses this function for making actions
  - e.g. greedy or \(\varepsilon\)-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
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

- 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


13th move for black:
fxe5 a5 £xc2 a¥xc2

Karjakin, Sergey 2788 – Timofeev, Arty 2665 1–0
C10 64th ch-RUS (6) 14.08.2011
Example: Annotated Chess Games

- 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**
  (+− ± 2 3 µ + −)
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 $\pi$ 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 $\pi_0$
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)
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

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

PBPI
1. start with policy $\pi_0$
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 $(s, a_i \preceq a_j)$ if it is
   - use all training examples to train a policy $\pi: S \rightarrow \Pi(A)$
1. goto 2. (until stop)
Advantages of a preference-based framework

- 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:

\[
\begin{align*}
\text{a}_1 & \quad \text{a}_2 & \quad \text{a}_3 \\
\text{a}_1 & \quad \text{a}_2 & \quad \text{a}_3 \\
\end{align*}
\]

- \(a_1\) is not significantly better than \(a_2\)  
- but we know \(a_1 \notin a_3\) and \(a_2 \in a_3\)

\(\rightarrow\) no training example for API  
\(\rightarrow\) 2 training examples for PBPI
Case Study 1
Learning from Action Preferences

**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
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

- random policy
- preference-based policy
- constant policies (4 settings + convex hull)

Graph showing toxicity levels against tumor size for different policy types.
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
Open Questions

- 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**