Multi-task Inference and Planning in Board Games using Multiple Imperfect Oracles

Project Presentation – Graphs in ML & RL

Lilian Besson and Basile Clement

École Normale Supérieure de Cachan (Master MVA)

January 19th, 2016

Please contact us by email if needed: `first.last@ens-cachan.fr`
Our slides, report, code and examples are on `http://lbo.k.vu/gml2016`.
Grade: We got 18/20 for our project.

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of our project

Multi-task Inference and Planning in Board Games using Imperfect Oracles

Overview of the presentation

Board game inference

Hypotheses on the game: ($\Longrightarrow$ represented as a **MDP**)

- Two players,
- Discrete turns,
- Finite number of states $\mathcal{S}$ and actions $\mathcal{A}$ (can be big!)

This includes:

- Chess, Go, Checkers, Chinese Checkers, 4-in-a-Row, etc.
- **Tic-Tac-Toe**, $\longleftarrow$ used for our experiments

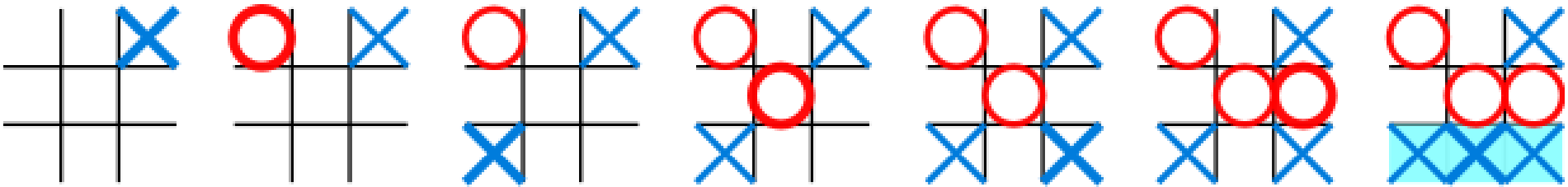
Goal: learn a good policy $\pi^\star$ to play the game.

A policy π is a distribution on action for each state: $\pi(s, a) = \mathbb{P}(a|s)$.

Board game inference

Hypotheses on the game: ($\Longrightarrow$ represented as a **MDP**)

- Two players,
- Discrete turns,
- Finite number of states $\mathcal{S}$ and actions $\mathcal{A}$ (can be big!)



Example of a game for 3-by-3 Tic-Tac-Toe, X winning against O. (from Wikimedia)

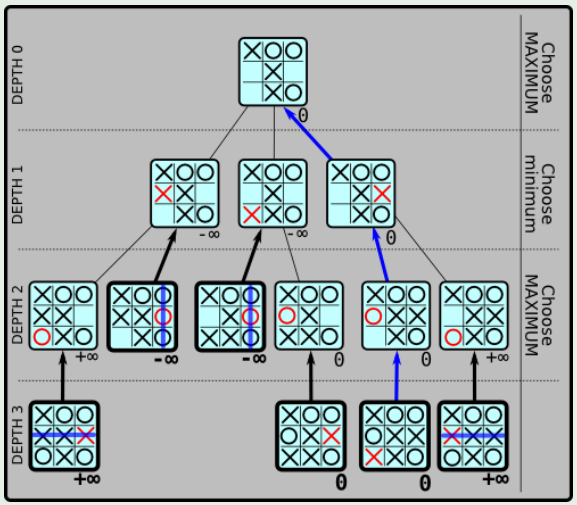
Minimax tree search:

Naive approach: *"Minimax"*

Complete tree search to select the move which maximizes the end-game score.

Quick and easy for the 3-by-3 Tic-Tac-Toe:

- We implemented it and used it for our experiments [a].
- Minimax is *optimal* here: it never looses (either win or draw).



[a] Figure from `beej.us/blog/data/minimax/`.

Minimax tree search: only for small games

But. . .

When there is too many policies

$\Longrightarrow$ Combinatorial explosion!

It only works for (very) small games!

One more hypothesis on the game:

$\Longrightarrow$ so we restrict to "linearly representable" games.

Inverse Reinforcement Learning

A few notations on multi-expert learning:

- M experts, $m = 1, \ldots, M$, all *independent*,
- They all play the *same game*,
- But may be against a different opponent,
- Each expert has some demonstrations $\mathcal{D}_m = \{d_m^{(i)}\}_i$.

Inverse Reinforcement Learning

A few notations on multi-expert learning:

- M experts, $m = 1, \dots, M$, all *independent*,
- They all play the *same game*,
- But may be against a different opponent,
- Each expert has some demonstrations $\mathcal{D}_m = \{d_m^{(i)}\}_i$.

Basic idea:

First: Learn from the demonstrations $\longrightarrow \{w_m^\star, \pi_m^\star\}_m$

Then: Aggregate the policies $\pi_m^\star$ $\longrightarrow w^\star, \pi^\star$

"Linearly-representable" games

We focus on "linearly-representable" games:

Instead of discrete state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ indexes ...
Use a features vector $g(a, s) \in \mathbb{R}^r$. $\Longrightarrow$ work in a vector space!

$\Longrightarrow$ Instead of combinatorial exploration, convex optimization can be used!

"Linearly-representable" games

We focus on "linearly-representable" games:

Instead of discrete state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ indexes ...
Use a features vector $g(a, s) \in \mathbb{R}^r$. $\Longrightarrow$ work in a vector space!

$\Longrightarrow$ Instead of combinatorial exploration, convex optimization can be used!

Hypothesis and usual RL notations:

- Optimal Q-value function $Q^\star$ is linear wrt. features:
  $Q^\star(a, s) = g_Q(a, s)^T \cdot w$,

"Linearly-representable" games

We focus on "linearly-representable" games:

Instead of discrete state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ indexes ...
Use a features vector $g(a, s) \in \mathbb{R}^r$. $\implies$ work in a vector space!

$\implies$ Instead of combinatorial exploration, convex optimization can be used!

Hypothesis and usual RL notations:

- Optimal Q-value function $Q^\star$ is linear wrt. features:
  $Q^\star(a, s) = g_Q(a, s)^T \cdot w$,
- The policy is obtained with a **softmax**, with an (inverse) temperature $\beta > 0$:

$$\pi_Q(a|s) = \mathrm{softmax}_\beta(Q)(s, a) \stackrel{\text{def}}{=} \exp\left(\beta Q(s, a)\right) / \Big(\sum_{a' \in \mathcal{A}} \exp\left(\beta Q(s, a')\right) \Big).$$

Ex.: Tic-Tac-Toe is a "linearly-representable" game

Example of board features for Tic-Tac-Toe:

$\hookrightarrow$ features should be *rotation invariant*.

- Number of "i-lets" for each player, i.e. lines/columns/diagonals with exactly i marks of the corresponding player and all other spaces blank,
- "i-diversity", the number of directions for i-lets for X and O,
- the number of marks on the diagonals for each player, etc.

Ex.: Tic-Tac-Toe is a "linearly-representable" game

Example of board features for Tic-Tac-Toe:

$\hookrightarrow$ features should be *rotation invariant*.

- Number of "i-lets" for each player, i.e. lines/columns/diagonals with exactly i marks of the corresponding player and all other spaces blank,
- "i-diversity", the number of directions for i-lets for X and O,
- the number of marks on the diagonals for each player, etc.

How many features?

- For 3-by-3 Tic-Tac-Toe, we first used 10, and then $F_3 = 55$ features!

Ex.: Tic-Tac-Toe is a "linearly-representable" game

Example of board features for Tic-Tac-Toe:

$\hookrightarrow$ features should be *rotation invariant*.

- Number of "i-lets" for each player, i.e. lines/columns/diagonals with exactly i marks of the corresponding player and all other spaces blank,
- "i-diversity", the number of directions for i-lets for X and O,
- the number of marks on the diagonals for each player, etc.

How many features?

- For 3-by-3 Tic-Tac-Toe, we first used 10, and then $F_3 = 55$ features!
- For n-by-n Tic-Tac-Toe, first $4n - 2$ simple features, and then $F_n = \frac{(4n-2)(4n-1)}{2}$ (with multi-variate binomial terms [KBB09, TD13]).
  $\implies F_n = O(n^2) = O(\text{size of the board})$: good!

How to learn the weight vector w_m of an expert?

$\Longrightarrow$ Maximum a posteriori [TD13]

Maximize the posterior distribution: $\xi(\beta, w_m|\mathcal{D}_m) \propto \mathbb{P}(\mathcal{D}_m|\pi_{\beta,w_m})\xi(\beta)\xi(w_m)$

Recall:

- $Q(s,a) = g_Q(s,a)^T \cdot w_Q$
- $\pi_Q(a|s) = \mathrm{softmax}_\beta(Q)(s,a) = \exp\left(\beta Q(s,a)\right) / \left(\sum_{a' \in \mathcal{A}} \exp\left(\beta Q(s,a')\right)\right)$
- Hyp: We chose a uniform and independent prior ξ on β and w_m.

How to learn the weight vector w_m of an expert?

$\Longrightarrow$ Maximum a posteriori [TD13]

Maximize the posterior distribution: $\xi(\beta, w_m|\mathcal{D}_m) \propto \mathbb{P}(\mathcal{D}_m|\pi_{\beta,w_m})\xi(\beta)\xi(w_m)$

Learning an expert policy: *LSTD-Q*

Maximize the (concave) log-likelihood $l_m(\beta w_m) \overset{\text{def}}{=} \log \mathbb{P}(\mathcal{D}_m|\pi_{\beta,w_m})$

$$= \frac{1}{|\mathcal{D}_m|} \sum_{d\in\mathcal{D}_m} \sum_{t=1}^{T_k} \left\{ \beta g_Q(a_{d,m}^{(t)}, s_{d,m}^{(t)})^T \cdot w_m - \ln\Big(\sum_{a'\in\mathcal{A}} \exp(\beta g_Q(a_{d,m}^{(t)}, s_{d,m}^{(t)})^T \cdot w_m) \Big) \right\}$$

$\hookrightarrow$ Learn $w_m^\star$ from the demonstrations $\mathcal{D}_m$, by **LSTD-Q**, as done by C. Dimitrikakis and A. Toussou [TD13].

- Homogeneous in βw_m, so let $\beta = 1$ (at first).
- Normalizing by $|\mathcal{D}_m|$ improves stability in practice.

First success: learning for one expert

First experiment, reproducing and extending [TD13]:

For 3-by-3 Tic-Tac-Toe, we can learn weight vectors for different kind of experts: random, optimal (minimax), or a convex combination of both ("ε-drunk" expert).

Player \ Opp.	Opt.	Rand.
Optimal	77%	93%
Random	58%	53%

(a) Learned *vs* Random.

Player \ Opp.	Opt.	Rand.
Optimal	100%	23%
Random	8%	0%

(b) Learned *vs* Optimal.

Table 1: Combined "Win" % rate (100 demonstrations, 100 tests)

Results:

Our LSTD-Q implementation worked well, and it confirmed [TD13] results.

Combining experts with a relative scoring

We assume to have a relative scoring on the experts, $e(m)$ (prior distribution).

Combine the learned weight vectors $w_m^\star$ linearly:

We simply set: $w^\star \overset{\text{def}}{=} \mathbb{E}_e\left[w_m^\star\right] = \sum_{m=1}^{M} e(m) w_m^\star$ (convex comb.).

Then use $w^\star$ to compute $Q^\star$, and finally $\pi^\star$ (as previously).

- *Example of* $e(m)$: "ELO" score for chess.
- Expectation on the weights $w_m^\star$ (or $Q_m^\star$), but NOT on the policies $\pi_m^\star$!

Combining experts with a relative scoring

We assume to have a relative scoring on the experts, $e(m)$ (prior distribution).

Combine the learned weight vectors $w_m^\star$ linearly:

We simply set: $w^\star \overset{\text{def}}{=} \mathbb{E}_e\left[w_m^\star\right] = \sum_{m=1}^{M} e(m) w_m^\star$ (convex comb.).

Then use $w^\star$ to compute $Q^\star$, and finally $\pi^\star$ (as previously).

- *Example of* $e(m)$: "ELO" score for chess.
- Expectation on the weights $w_m^\star$ (or $Q_m^\star$), but NOT on the policies $\pi_m^\star$!

Problem with this prior:

- Not realistic: what is a good prior? Where does it come from?
- Hard to test experimentally: no prior for our generated demonstrations.

Algo 1: Multi-expert aggregation with a prior

Data: g_Q: board features function,
Data: Number M, and a database $\mathcal{D}_m$ of demonstrations for each expert m,
Data: A prior $e(m)$ on the experts strength,
Data: An inverse temperature β for the softmax ($\beta = 1$ works, because no constraint).
`/* (For each expert, separately) */`
for $m = 1$ *to* M **do**
  `/* Learn` $\pi^\star_m$ `from` $\mathcal{D}_m$ `the LSTD-Q algorithm */`
  Compute the log-likelihood $w \mapsto l_m(w)$; `/* As done before */`
  Compute its gradient $w \mapsto \nabla l_m(w)$; `/* cf. report */`
  Chose an arbitrary starting point, let $w_m^{(0)} = [0, \ldots, 0]$;
  $w^\star_m \leftarrow$ L-BFGS$(l_m, \nabla l_m, w_m^{(0)})$; `/* 1-st order concave optimization */`
end
$w^\star = \mathbb{E}_m\left[w^\star_m\right]$, $Q^\star = g \cdot w^\star$ (expectation based on the distribution $e(m)$);
Result: $\pi^\star = \mathrm{softmax}_\beta\left(Q^\star\right)$ the aggregated optimal policy we learn.

Algorithm 1: Naive multi-task learning algorithm for imperfect oracles, with a prior on their strength.

Computing the distribution *a posteriori*?

Key idea

Use the $w_m^\star$ or $\pi_m^\star$ to compare the M experts.

Instead of relying on a prior $e(m)$, can we compute a distribution a posteriori?

Computing the distribution *a posteriori*?

Key idea

Use the $w_m^\star$ or $\pi_m^\star$ to compare the M experts.

Instead of relying on a prior $e(m)$, can we compute a distribution a posteriori?

1st idea : using temperatures

Intuition: as the max-likelihood problem is homogeneous in βw,
a temperature can be set with $\beta_m \overset{\text{def}}{=} \|w_m^\star\|$
($\equiv$ considering the constrained problem $\|w_m\| = 1$).
"Cold" $\beta_m \implies$ expert m "confident" in his result $\implies$ higher score $e_\beta(m)$?
We tried, but... we could not achieve satisfactory results:
confident $\nleftrightarrow$ efficient!

Computing the distribution *a posteriori*?

Key idea

Use the $w_m^\star$ or $\pi_m^\star$ to compare the M experts.

Instead of relying on a prior $e(m)$, can we compute a distribution a posteriori?

2nd idea : test all the experts on a fixed opponent

To try to evaluate their (relative) strengths, make them play against a common opponent π_0, e.g. fully random.
Problem: not advisable against a good opponent.

Computing the distribution *a posteriori*?

Key idea

Use the $w_m^\star$ or $\pi_m^\star$ to compare the M experts.

Instead of relying on a prior $e(m)$, can we compute a distribution a posteriori?

2nd idea : test all the experts on a fixed opponent

To try to evaluate their (relative) strengths, make them play against a common opponent π_0, e.g. fully random.
Problem: not advisable against a good opponent.
If we have a good opponent to test against... use it instead of learning!

A different approach – infer the transitions?

What if we use LSTD-Q to learn the opponents O_m?

Instead of learning the experts policies, learn the MDP they were playing against.

A different approach – infer the transitions?

What if we use LSTD-Q to learn the opponents O_m?

Instead of learning the experts policies, learn the MDP they were playing against.

Why ?

Then we can perform a "clever" tree search against each opponents, to learn how to beat the best one.

The Coherent Inference algorithm, quick explanation

Tree search and message passing algorithm

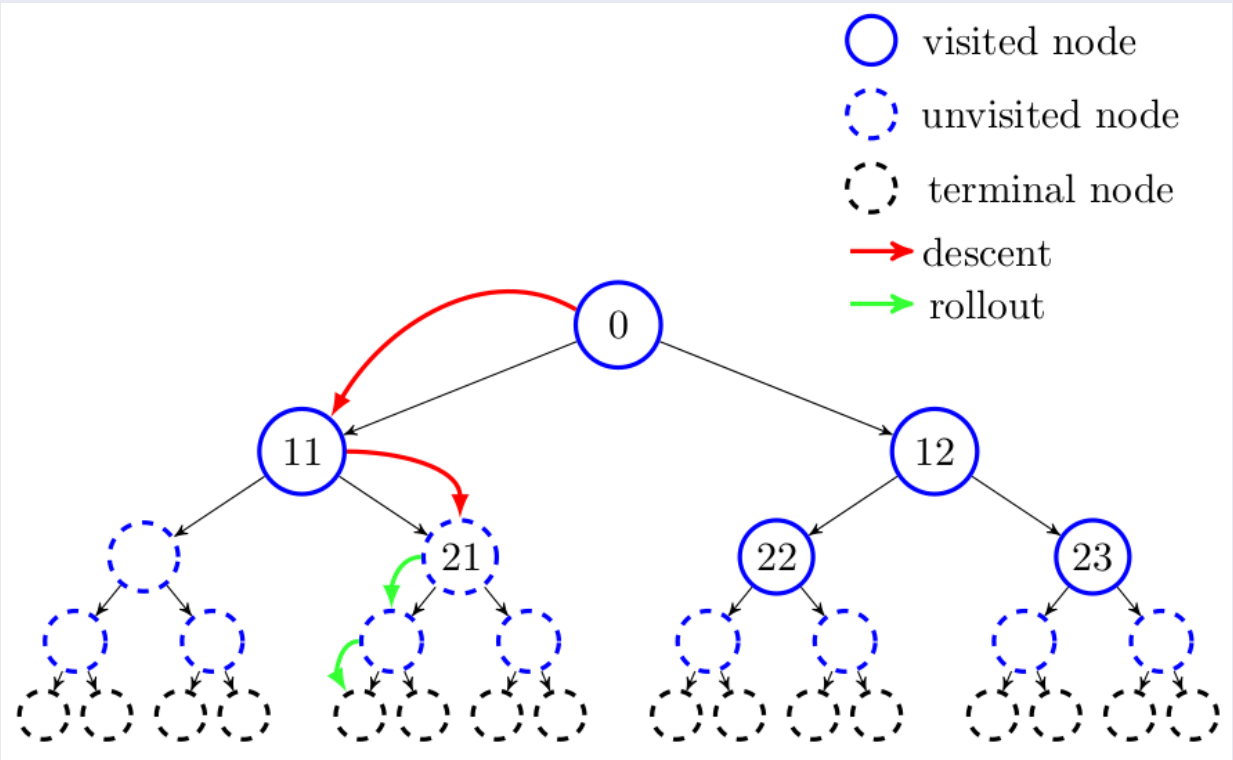


Illustration of the Coherent Inference algorithm

Score $g_i \sim \mathcal{N}(g_{\pi_i}, 1)$
Infer the distribution with $\mathrm{sgn}(g) = \text{win, loss}$ on leaves

Algo 2: multi-opponent algorithm combining [TD13] and [PD15b]

Data: g_Q: board features function,
Data: M, and a database $\mathcal{D}_m$ of demonstration for each opposing expert m,
`/* 1. Off-line learning step */`
for $m = 1$ *to* M **do**
  Learn $\pi_m^{\star,O}$ for the opposing player from $\mathcal{D}_m$ using LSTD-Q; `/* As above */`
end
`/* 2. Play step (on-line during the game) */`
Data: b: the current board state
for $m = 1$ *to* M **do**
  `/* Use the coherent inference algorithm from [PD15b] */`
  Learn the G values starting from b, using $\pi_m^{\star,O}$ for the opponent's distribution;
  Sample r_m from the distribution of G at b; `/* Reward */`
  for $a \in A$ **do**
    Sample c_a from the distribution of G at $b + a$ (state after playing move a).
  end
  Let a_m be $\arg\max_a c_a$ be the best answer to $\pi_m^{\star,O}$;
end
Let $m^\star$ be $\arg\min_m r_m$ be the strongest opponent;
Return $a^\star$ be $a_{m^\star}$ the best answer to the strongest opponent.

Algorithm 2: Multi-task algorithm for imperfect opposing experts.

Results for our multi-expert algorithm 2

Player and opponent	Run 1	Run 2	Run 3	Run 4
Opposing Expert 1	25%	34%	37%	44%
Opposing Expert 2	34%	74%	27%	13%
Aggregated 1 and 2	29%	64%	41%	41%

Table 2: Draw % rate using different opposing experts (against optimal).

Combining several experts: (for 3-by-3 Tic-Tac-Toe)

Results for our multi-expert algorithm 2

Player and opponent	Run 1	Run 2	Run 3	Run 4
Opposing Expert 1	25%	34%	37%	44%
Opposing Expert 2	34%	74%	27%	13%
Aggregated 1 and 2	29%	64%	41%	41%

Table 2: Draw % rate using different opposing experts (against optimal).

Combining several experts: (for 3-by-3 Tic-Tac-Toe)

- usually improving performance over using a single expert – the presence of a good opposing expert is reducing the penalty from having a bad opposing expert,

Results for our multi-expert algorithm 2

Player and opponent	Run 1	Run 2	Run 3	Run 4
Opposing Expert 1	25%	34%	37%	44%
Opposing Expert 2	34%	74%	27%	13%
Aggregated 1 and 2	29%	64%	41%	41%

Table 2: Draw % rate using different opposing experts (against optimal).

Combining several experts: (for 3-by-3 Tic-Tac-Toe)

- usually improving performance over using a single expert – the presence of a good opposing expert is reducing the penalty from having a bad opposing expert,
- and having several good opposing experts will usually improve over using a single one of them (e.g. in run 3).

Results for our multi-expert algorithm 2

Player and opponent	Run 1	Run 2	Run 3	Run 4
Aggregated 1 and 2	29%	64%	41%	41%
Pengkun's Coherent Inference	Average = 40%			

Table 2: Draw % rate using different opposing experts (against optimal).

Unfortunately:

- Usually does not improve over the performance (although it does not loses much from it either) of simply using the Coherent Inference algorithm (from [PD15b]),
- and is dependent on the performance of the opposing vector learned (high variance).

Quick sum-up

We studied. . .

- Policy learning for board games,
- Inverse Reinforcement Learning (IRL).

Quick sum-up

We showed how to. . .

- represent value functions using feature vectors,
- use LSTD-Q to learn the feature weights for a single expert,
- combine weights with a good *prior* estimate of the experts strength,
- (try to) estimate the experts strength *a posteriori*,
- learn the MDPs' transitions & explore them with Coherent Inference.

Quick sum-up

Experimentally, we. . .

- wrote an optimized implementation of both LSTD-Q for one expert [TD13] and multi-expert,
- and of Coherent Inference [PD15b],
- experimented with prior distributions and β-temperatures,
- experimented on the MDP transition learning.

Thank you!

Thank you for your attention.

Questions?

Questions?

Questions?

Questions?

Want to know more?

↪ Explore the references, or read our project report,

↪ And contact us by e-mail if needed (first.last@ens-cachan.fr).

Main references:

- Liu Pengkun, *"Implementation and Experimentation of Coherent Inference in Game Tree"* (2015). Master's thesis, Chalmers University of Technology.
- Aristide Toussou and Christos Dimitrakakis (2013). *"Probabilistic Inverse Reinforcement Learning in Unknown Environments"*.
- Christos Dimitrakakis and Constantin Rothkopf (2012). *"Bayesian Multi-Task Inverse Reinforcement Learning"*.

Appendix

Outline of the appendix:

- More references given below,
- Code and raw results from some experiments:
  $\longrightarrow$ `http://lbo.k.vu/gml2016`.
- MIT License.

More references I

Our main references are the work of Liu Pengkun in 2015 [PD15b, PD15a], and the previous work of Christos Dimitrikakis [TD13, DR12, Dim15].

Christos Dimitrakakis (December 2015).

BeliefBox, a Bayesian framework for Reinforcement Learning (GitHub repository).

URL `https://github.com/olethrosdc/beliefbox`, online, accessed 20.12.2015.

Christos Dimitrakakis and Constantin A. Rothkopf (2012).

Bayesian Multitask Inverse Reinforcement Learning.

In *Recent Advances in Reinforcement Learning*, pages 273–284. Springer.

URL `http://arxiv.org/abs/1106.3655v2`.

Wolfgang Konen and Thomas Bartz-Beielstein (2009).

Reinforcement Learning for Games: Failures and Successes.

In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2641–2648. ACM.

URL `http://doi.acm.org/10.1145/1570256.1570375`.

More references II

Liu Pengkun and Christos Dimitrakakis (June 2015).

Code for Implementation and Experimentation of Coherent Inference in Game Tree (GitHub repository).

URL `https://github.com/Charles-Lau-/thesis`, online, accessed 20.12.2015.

Liu Pengkun and Christos Dimitrakakis (June 2015).

Implementation and Experimentation of Coherent Inference in Game Tree.

Master's thesis, Chalmers University of Technology.

Aristide C. Y. Toussou and Christos Dimitrakakis (2013).

Probabilistic Inverse Reinforcement Learning in Unknown Environments.

arXiv preprint arXiv:1307.3785.

URL `http://arxiv.org/abs/1307.3785v1`.

Open-Source Licensed

License