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

1. Presentation, hypotheses and notations

2. Starting with the single-expert setting
   - Learn to represent an expert linearly [TD13]
   - Implementation and results for single-expert

3. Extension to the multi-expert setting
   - Our first aggregation algorithm (using LSTD-Q) [TD13]
   - Combining experts, with a prior on their strength
   - Compute a distribution on expert a posteriori?

4. Infer the transitions for each expert
   - Intuition behind our second algorithm [PD15b]
   - Quick explanation of Pengkun’s algorithm [PD15b]
   - Combining two approaches [TD13, PD15b]
   - Implementation and results for multi-expert

5. Conclusion
Board game inference

**Hypotheses on the game:**

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

This includes:

- Chess, Go, Checkers, Chinese Checkers, 4-in-a-Row, etc.
- Tic-Tac-Toe, ← used for our experiments

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

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

Hypotheses on the game: (\(\implies\) represented as a MDP)

- Two players,
- Discrete turns,
- Finite number of states \(S\) and actions \(A\) (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

⇒ Combinatorial explosion!

It only works for (very) small games!

One more hypothesis on the game:

⇒ 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, \( m = 1, \ldots, M \), all independent,
– They all play the same game,
– But may be against a different opponent,
– Each expert has some demonstrations \( D_m = \{d_m^{(i)}\}_i \).

Basic idea:

**First:** Learn from the demonstrations

**Then:** Aggregate the policies \( \pi_m^* \)

\[ \Rightarrow \{w_m^*, \pi_m^*\}_m \]

\[ \Rightarrow w^*, \pi^* \]
“Linearly-representable” games

We focus on “linearly-representable” games:

Instead of discrete state $s \in S$ and action $a \in A$ 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!
We focus on “linearly-representable” games:

Instead of discrete state $s \in S$ and action $a \in A$ indexes …

Use a **features vector** $g(a, s) \in \mathbb{R}^r$. ⇒ work in a vector space!

⇒ Instead of combinatorial exploration, convex optimization can be used!

Hypothesis and usual RL notations:

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

\[ \]
“Linearly-representable” games

We focus on “linearly-representable” games:

Instead of discrete state \( s \in S \) and action \( a \in A \) indexes …

Use a features vector \( g(a, s) \in \mathbb{R}^r \). 

\[\Rightarrow \text{ work in a vector space!}\]

\[\Rightarrow \text{ Instead of combinatorial exploration, convex optimization can be used!}\]

Hypothesis and usual RL notations:

– Optimal \( Q \)-value function \( Q^* \) is linear wrt. features:

\[Q^*(a, s) = g_Q(a, s)^T \cdot w,\]

– The policy is obtained with a softmax, with an (inverse) temperature \( \beta > 0 \):

\[\pi_Q(a|s) = \text{softmax}_\beta(Q)(s, a) \overset{\text{def}}{=} \exp(\beta Q(s, a)) / \left( \sum_{a' \in A} \exp(\beta Q(s, a')) \right).\]
Ex.: Tic-Tac-Toe is a “linearly-representable” game

Example of board features for Tic-Tac-Toe:

- 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.

$\Leftrightarrow$ features should be rotation invariant.
Ex.: Tic-Tac-Toe is a “linearly-representable” game

Example of board features for Tic-Tac-Toe:

→ 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:

- 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]).

\[
\Rightarrow F_n = O(n^2) = O(\text{size of the board}): \text{good!}
\]
How to learn the weight vector $w_m$ of an expert?

$\implies$ Maximum a posteriori

Maximize the posterior distribution: $\xi(\beta, w_m | D_m) \propto P(D_m | \pi_{\beta, w_m}) \xi(\beta) \xi(w_m)$

Recall:
- $Q(s, a) = g_Q(s, a)^T \cdot w_Q$
- $\pi_Q(a|s) = \text{softmax}_\beta(Q)(s, a) = \exp(\beta Q(s, a)) / \left(\sum_{a' \in A} \exp(\beta Q(s, a'))\right)$
- Hyp: We chose a \textbf{uniform} and independent \textbf{prior} $\xi$ on $\beta$ and $w_m$. 
How to learn the weight vector $w_m$ of an expert?

Maximize the posterior distribution: $\xi(\beta, w_m | D_m) \propto P(D_m | \pi_\beta, w_m) \xi(\beta) \xi(w_m)$

Learning an expert policy: $LSTD-Q$

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

$= \frac{1}{|D_m|} \sum_{d \in D_m} \sum_{t=1}^{T_k} \left\{ \beta g_Q(a_d^{(t)}, s_d^{(t)})^T \cdot w_m - \ln \left( \sum_{a' \in A} \exp(\beta g_Q(a_d^{(t)}, s_d^{(t)})^T \cdot w_m) \right) \right\}$

→ Learn $w_m^*$ from the demonstrations $D_m$, by $LSTD-Q$, as done by C. Dimitrikakis and A. Toussou [TD13].

- Homogeneous in $\beta w_m$, so let $\beta = 1$ (at first).
- Normalizing by $|D_m|$ 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 ("$\varepsilon$-drunk" expert).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>77%</td>
<td>93%</td>
</tr>
<tr>
<td>Random</td>
<td>58%</td>
<td>53%</td>
</tr>
</tbody>
</table>

(a) Learned vs Random.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>100%</td>
<td>23%</td>
</tr>
<tr>
<td>Random</td>
<td>8%</td>
<td>0%</td>
</tr>
</tbody>
</table>

(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 $w_m^*$ linearly:

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

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

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

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

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

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

**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 $D_m$ of demonstrations for each expert $m$,
Data: A prior $e(m)$ on the experts strength,
Data: An inverse temperature $\beta$ for the softmax ($\beta = 1$ works, because no constraint).

/* (For each expert, separately) */
for $m = 1$ to $M$ do
  /* Learn $\pi^*_m$ from $D_m$ 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 $w^{(0)}_m = [0, \ldots, 0];$
  $w^*_m \leftarrow$ L-BFGS($l_m, \nabla l_m, w^{(0)}_m$); /* 1-st order concave optimization */
end

$w^* = \mathbb{E}_m [w^*_m], Q^* = g \cdot w^*$ (expectation based on the distribution $e(m)$);
Result: $\pi^* = \text{softmax}_\beta (Q^*)$ 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 \textit{a posteriori}?

**Key idea**

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

Instead of relying on a prior $e(m)$, can we compute a distribution \textit{a posteriori}?
Computing the distribution \textit{a posteriori}?

\textbf{Key idea}

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

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

\textbf{1st idea: using temperatures}

Intuition: as the max-likelihood problem is homogeneous in $\beta w$, a temperature can be set with $\beta_m \overset{\text{def}}{=} ||w_m^*||$ ($\equiv$ 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... \textbf{we could not achieve satisfactory results:}

\textbf{confident} $\Leftrightarrow$ \textbf{efficient}!
Computing the distribution \textit{a posteriori}?

**Key idea**

Use the $w^*_m$ or $\pi^*_m$ to compare the $M$ experts.

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

**2\textsuperscript{nd} idea : test all the experts on a fixed opponent**

To try to evaluate their (relative) strengths, make them play against a common opponent $\pi_0$, e.g. fully random.

Problem: not advisable against a good opponent.
Computing the distribution *a posteriori*?

**Key idea**

Use the $w_m^*$ or $\pi_m^*$ 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 $\pi_0$, 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

Score $g_i \sim \mathcal{N}(g_{\pi_i}, 1)$
Infer the distribution with $\text{sgn}(g) = \text{win, loss on leaves}$
Algorithm 2: multi-opponent algorithm combining [TD13] and [PD15b]

Data: $g_ℚ$: board features function,
Data: $M$, and a database $𝒟_ℌ$ of demonstration for each opposing expert $ℌ$,

/* 1. Off-line learning step */

for $ℌ = 1$ to $M$ do
    Learn $π^*_ℌ$ for the opposing player from $𝒟_ℌ$ using LSTD-ℚ;  /* As above */
end

/* 2. Play step (on-line during the game) */

Data: $b$: the current board state

for $ℌ = 1$ to $M$ do
    /* Use the coherent inference algorithm from [PD15b] */
    Learn the $G$ values starting from $b$, using $π^*_ℌ$ for the opponent’s distribution;
    Sample $r_ℌ$ from the distribution of $G$ at $b$;  /* Reward */
    for $a ∈ A$ do
        Sample $c_a$ from the distribution of $G$ at $b + a$ (state after playing move $a$).
    end
    Let $a_ℌ$ be arg max $a c_a$ be the best answer to $π^*_ℌ$;
end

Let $ℌ^*$ be arg min $ℌ r_ℌ$ be the strongest opponent;

Return $a^* = a_{ℌ^*}$ the best answer to the strongest opponent.

Algorithm 2: Multi-task algorithm for imperfect opposing experts.
Results for our multi-expert algorithm 2

<table>
<thead>
<tr>
<th>Player and opponent</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opposing Expert 1</td>
<td>25%</td>
<td>34%</td>
<td>37%</td>
<td>44%</td>
</tr>
<tr>
<td>Opposing Expert 2</td>
<td>34%</td>
<td>74%</td>
<td>27%</td>
<td>13%</td>
</tr>
<tr>
<td>Aggregated 1 and 2</td>
<td>29%</td>
<td>64%</td>
<td>41%</td>
<td>41%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Player and opponent</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opposing Expert 1</td>
<td>25%</td>
<td>34%</td>
<td>37%</td>
<td>44%</td>
</tr>
<tr>
<td>Opposing Expert 2</td>
<td>34%</td>
<td>74%</td>
<td>27%</td>
<td>13%</td>
</tr>
<tr>
<td>Aggregated 1 and 2</td>
<td>29%</td>
<td>64%</td>
<td>41%</td>
<td>41%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Player and opponent</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opposing Expert 1</td>
<td>25%</td>
<td>34%</td>
<td>37%</td>
<td>44%</td>
</tr>
<tr>
<td>Opposing Expert 2</td>
<td>34%</td>
<td>74%</td>
<td>27%</td>
<td>13%</td>
</tr>
<tr>
<td>Aggregated 1 and 2</td>
<td>29%</td>
<td>64%</td>
<td>41%</td>
<td>41%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Player and opponent</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregated 1 and 2</td>
<td>29%</td>
<td>64%</td>
<td>41%</td>
<td>41%</td>
</tr>
<tr>
<td>Pengkun’s Coherent Inference</td>
<td></td>
<td></td>
<td>Average = 40%</td>
<td></td>
</tr>
</tbody>
</table>

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 $\beta$-temperatures,
- experimented on the MDP transition learning.
Thank you for your attention.
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:

- 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,
– 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).*

- Christos Dimitrakakis and Constantin A. Rothkopf (2012).
  *Bayesian Multitask Inverse Reinforcement Learning.*

- Wolfgang Konen and Thomas Bartz-Beielstein (2009).
  *Reinforcement Learning for Games: Failures and Successes.*
  URL http://doi.acm.org/10.1145/1570256.1570375.
Liu Pengkun and Christos Dimitrakakis (June 2015). 
*Code for Implementation and Experimentation of Coherent Inference in Game Tree (GitHub repository).* 

Liu Pengkun and Christos Dimitrakakis (June 2015). 
*Implementation and Experimentation of Coherent Inference in Game Tree.* 
Master’s thesis, Chalmers University of Technology.

*Probabilistic Inverse Reinforcement Learning in Unknown Environments.* 
Open-Source Licensed

License

These slides and our article (and the additional resources – including code, images, etc), are open-sourced under the terms of the MIT License.

Copyright 2015-2016, © Lilian Besson and Basile Clement.