Towards Learning Representations for Efficient Reinforcement Learning

Emma Brunskill
Carnegie Mellon University
Joint work with Lihong Li, Yun-En Liu, Travis Mandel, & Zoran Popovic
If you invent a breakthrough in AI, so that machines can learn, that is worth 10 Microsofs
– Bill Gates

If you can invent an AI that helps us all learn to be as motivated, knowledgeable & intelligent as Bill Gates, that is worth 7.1 Billion Microsofs
– me
(Machine)
Learning to
Improve Learning
Efficient learning important in high stakes domains

Abstractions can help speed learning
Challenge: Abstraction Sufficient To Code Optimal Policy May Not Allow Learning That Policy

Also see Li, Walsh, Littman 2006
Little Prior Work on Intersection

Abstraction Learning

Efficient Exploration

Very little theoretical work
Towards Learning Representations for Efficient Reinforcement Learning

• Learning options to speed learning
• Learning state abstractions to speed learning

Speed = Amount of data need to learn to make near optimal decisions
Options / Macro-Actions
But Do Options Really Help Speed* Learning?

• Prior evidence is mixed
• Sometimes accelerated learning, and sometimes slow learning (Jong, Hester, Stone 2008)
Options Discovery?

- Where do these options (if helpful!) come from?
- Encouraging empirical benefit but heuristic
  - Maximize “compression” (Thrun & Schwartz, Pickett & Barto)
  - Sub-goal discovery (Stolle & Precup, Mannor et al)
  - Homomorphisms (Soni & Singh) & shared features (Konidaris & Barto)
Contributions

1. How and when options speed* reinforcement learning
2. Discover options across tasks to provably accelerate* RL in future tasks

* As measured by sample complexity of learning.
Background: SMDP & Options

Figure from Sutton, Precup & Singh 1999
Background: SMDP & Options

\[ P(s', \tau | s, a) \]

waiting time

Figure from Sutton, Precup & Singh 1999
Background: SMDP & Options

\[ P(s', \tau | s, a) \]

waiting time

Figure from Sutton, Precup & Singh 1999
Background: SMDP & Options

Bellman operator for SMDPs:

\[ Q(s, a) = r(s, a) + \sum_{s'} \left[ \sum_{\tau} p(s', \tau | s, a) \gamma^\tau \right] \max_{a'} Q(s', a') \]

Expected discount factor for (s,a) to s'
Contributions

1. How and when options speed* reinforcement learning

2. Discover options across tasks to provably accelerate* RL in future tasks

* As measured by sample complexity of learning.
Prior: RL Sample Complexity of Exploration in MDPs

• Number of sub e-optimal decisions

\[ \sum_t I \left( V_{A_t} (s_t) \leq V^*(s_t) - e \right) \]

• RL algorithm is PAC-MDP (Kearns & Singh, Brafman & Tennenholtz) if:
  – Sample complexity poly func of MDP params with high probability
New: Sample Complexity Of Exploration in SMDPs

\[
\sum_t \tau_t \cdot \mathbb{1} \left( V^{A_t}(s_t) \leq V^*(s_t) - \epsilon \right)
\]

- RL algorithm PAC-SMDP if polynomial in SMDP params with high probability

Weighed by waiting time (# steps till choose new action)
Condition on SMDP for Any Algorithm to be PAC

- Unbounded waiting time \( \mathcal{T} \)
  - Could never return from a bad decision!
  - SC infinite!
Condition on SMDP for Any Algorithm to be PAC

- Unbounded waiting time $\mathcal{T}$
  - Could never return from a bad decision!
  - SC infinite!

- Assume $\mathcal{T}$
  - Has expected value $< L$
  - Distribution sub-Gaussian with parameter $C$
Algorithms For PAC-SMDP

• Ala MDPs, drive exploration towards unknown s-a by making reward for unknown s-a large in alternate SMDP
Marginal Waiting Time

\[ P(\tau \mid s, a) \]
Expected Discount Factor

\[ \bar{\gamma}_{sa} = \sum_{\tau} \gamma^{\tau} P(\tau \mid s, a) \]

Marginal (over s') expected discount factor for (s,a)
SMDP-Rmax Sample Complexity

\[ \bar{\gamma}_{sa} = \sum_{\tau} \gamma^{\tau} P(\tau \mid s, a) \]

Marginal (over \(s')\) expected discount factor for \((s,a)\)

\[ \frac{V_{\text{max}}^3}{\epsilon^3} \sum_{sa} \frac{N_{sa}}{(1 - \bar{\gamma}_{sa})^3} \left( \frac{1}{1 - \gamma} + L + \frac{1}{\sqrt{C}} \right) \]
SMDP-Rmax vs Rmax

\[
\frac{V_{\text{max}}^3}{\varepsilon^3} \sum_{sa} \frac{N_{sa}}{(1 - \bar{\gamma}_{sa})^3} \left( \frac{1}{1 - \gamma} + L + \frac{1}{\sqrt{C}} \right)
\]

\[
\frac{V_{\text{max}}^3}{\varepsilon^3} \left| S \right| \left| A_{\text{prim}} \right| \frac{N_{sa}}{(1 - \gamma)^3}
\]
Benefit* If Less Pairs To Learn

\[
\frac{V^3}{\epsilon^3} \sum_{s,a} \frac{N_{sa}}{(1 - \gamma_{sa})^3} \left( \frac{1}{1 - \gamma} + L + \frac{1}{\sqrt{C}} \right)
\]

* Not quite: slightly different notions of near optimality
**Duration/Discount**

\[
\frac{1}{1 - \gamma} + L + \frac{1}{\sqrt{C}} \leq \frac{(1 - \bar{\gamma})^2}{(1 - \gamma)^3}
\]

- Benefit* when
  - Waiting time not much longer than \( \frac{1}{1 - \gamma} \) compared to how much smaller effective discount factor is than discount factor

* Not quite: slightly different notions of near optimality
Consistent With Empirical Results of Jong et Al.

• Options + primitive actions can be worse than primitive only
  – SC expected to increase use all
Consistent With Empirical Results of Jong et Al.

• Options + primitive actions can be worse than primitive only

• Limiting some states to options & others to primitive can speed learning
  – SC 9.6 *10^6 all primitive > 1.8*10^6 limiting
Contributions

1. How and when options speed* reinforcement learning

2. Discover options across tasks to provably accelerate* RL in future tasks

* As measured by sample complexity of learning.
Lifelong Learning Setup

Set of MDPs $M$

$M_{t1} \rightarrow M_{t2} \rightarrow M_{t3} \rightarrow M_{t4} \rightarrow \ldots$

Exists options set $O$ that allows $\varepsilon$-optimal policies for all $M$
Lifelong RL With Options

Set of MDPs $M$

$M_{t1} \rightarrow M_{t2} \rightarrow \ldots$

Phase 1:
Run $E^3$ with primitive $a$ &
find
e-optimal policies
Lifelong RL With Options

Set of MDPs $M$

$M_t_1 \rightarrow M_{t_2} \rightarrow ...$

Phase 1:
Run $E^3$ with primitive $a$ &
find
e-optimal policies

Discover $O^*$
to represent e-optimal policies for all $M$
Lifelong RL With Options

Set of MDPs $M$

Phase 1:
Run $E^3$ with primitive $a$ & find e-optimal policies

Discover $O^\sim$ to represent e-optimal policies for all $M$

Phase 2:
Run SMDP-Rmax with $O^\sim$

$M_{t1} \rightarrow M_{t2} \rightarrow ...$

$M_{t20} \rightarrow M_{t21} \rightarrow ...$
New Option Discovery Alg

• At least as hard as set-covering

• Instead, propose greedy approach that constructs options to reduce SC of covering MDPs in phase 1
Simulation

from Sutton, Precup, Singh 1999.
104 states, 8 actions
## Significantly & Substantially Better

<table>
<thead>
<tr>
<th></th>
<th># State-Options</th>
<th>Sample Complexity Bound</th>
<th>Avg. Reward Phase 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Only</td>
<td>832</td>
<td>832000</td>
<td>10470</td>
</tr>
<tr>
<td>PolicyBlocks</td>
<td>985</td>
<td>942450</td>
<td>11229</td>
</tr>
<tr>
<td>(Pickett &amp; Barto)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAC-Inspired</td>
<td>550</td>
<td>511605</td>
<td>13145</td>
</tr>
</tbody>
</table>
Performance Quite Close to Hand Designed Options

<table>
<thead>
<tr>
<th>Method</th>
<th># State-Options</th>
<th>Sample Complexity Bound</th>
<th>Avg. Reward Phase 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Only</td>
<td>832</td>
<td>832000</td>
<td>10470</td>
</tr>
<tr>
<td>PolicyBlocks (Pickett &amp; Barto)</td>
<td>985</td>
<td>942450</td>
<td>11229</td>
</tr>
<tr>
<td>PAC-Inspired</td>
<td>550</td>
<td>511605</td>
<td>13145</td>
</tr>
<tr>
<td>Hand Coded</td>
<td>189</td>
<td>85765</td>
<td>14718</td>
</tr>
</tbody>
</table>
Summary

1. Options can speed* reinforcement learning if reduce pairs to learn and/or reduce effective discount factor without too long of an additional waiting period

2. Can discover options across tasks to provably accelerate* RL in future tasks

* As measured by sample complexity of learning.
Towards Learning Representations for Efficient Reinforcement Learning

• Learning options to speed learning
• Learning state abstractions to speed learning

* With a focus on approaches with guarantees
Approach

• Efficient exploration by representing uncertainty over (model) parameter values
Approach

• Efficient exploration by representing uncertainty over model & parameter values

• Adapt representation based on data
  – Bayesian posterior

• Reduce computation by considering particular forms of state abstractions
Setting

• Discrete state and action MDPs
• Relative outcome dynamics
  – $s + \text{outcome} \rightarrow \text{next state } s'$
  – Know set of outcomes
  – Don’t know probability distribution over outcomes
Approach: Cluster States By Relative Dynamics to Speed Learning

• Intuition: many states may have same relative dynamics
• If knew which states had same relative dynamics, can provably speed learning (Leffler et al 2007, Brunskill et al. 2008/2009)
• But we don’t...
• Want to cluster states into those with similar dynamics, but don’t know dynamics of states
Idea: Change Abstraction Based on Data

• Little data, more states clumped together
  – Can’t tell if states are different
• More data, split states with different dynamics

* In a way that doesn’t prevent us from learning optimal policy.
Prior Work: Thompson Sampling for Reinforcement Learning
(Osband, Russo, Van Roy 2013, Osband and Vany Roy 2014)

• Define MDP
• Prior over MDP model parameters
• Sample from prior
• Compute optimal policy for those parameters
• Act
• Update posterior over parameters given data
New Work: Thompson **Clustering** for Reinforcement Learning

- Define original state and action space
- Prior over state clusters/aggregations & parms
- Sample state aggregation for each action from prior and parameters for aggregations
  - Intuitively, sample model and model parameters
- Compute optimal policy for those parameters
- Act
- Update posterior over parameters given data
TCRL Could Speed Learning

• Define original state and action space
• Prior over state clusters/aggregations & parms
• Sample state aggregation for each action from prior and parameters for aggregations
  → If aggregate a lot of states, share their data, get better model of dynamics if states are the same
• Compute optimal policy for those parameters
• Act
• Update posterior over parameters given data
Involves Sampling & Updating Distribution over Abstractions

- Define original state and action space
- Prior over state clusters/aggregations & parms
- **Sample state aggregation for each action from prior and parameters for aggregations**
- Compute optimal policy for those parameters
- Act
- Update posterior over parameters given data
Conceptually Appealing But Prior Updating and Sampling Expensive

- # clusterings = $n^n$ where $n = \# \text{ states}$
- Introduce two algorithms that are (fairly) computationally tractable
  - TCRL-Relaxed
  - TCRL-Theoretic
- Use specific priors over state-action dynamics clusterings
- And sometimes approximation over sampling
Consider Clustering “Nearby” States, Likely to Have Same Dynamics
TCRL-Relaxed

• Consider fairly flexible way of clustering states
• But sample from this in an approximate way
TCRL-Relaxed Procedure

1. Build DAG
TCRL-Relaxed:

2. Sample Clustering Given Data D

Note: Greedy in the sense that future clusterings are not considered.

\[ P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)} \]

\[ P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n) d\theta \]

C = binary variable
1 if cluster states
0 if not

Sample C given P(C|D)

Easy to compute for Dirichlets
TCRL-Relaxed:

2. Proceed Breadth First

\[ P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)} \]

\[ P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n)\,d\theta \]

C = binary variable
1 if cluster states
0 if not
TCRL-Relaxed:

2. Proceed Breadth First

\[ P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)} \]

\[ P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n) d\theta \]

Sample C given \( P(C|D) \)

C = binary variable
1 if cluster states
0 if not
TCRL-Relaxed:

2. First Consider Immediate Ancestor

\[ P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)} \]

\[ P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n) \, d\theta \]

Sample C given P(C|D)

C = binary variable
1 if cluster states
0 if not
TCRL-Relaxed: If Cluster, Consider Next Ancestor

C = binary variable
1 if cluster states
0 if not

\[
P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)}
\]

\[
P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n) d\theta
\]
TCRL-Theoretic:
Restrict Clusterings Considered, Strong Guarantees
TCRL-Theoretic:

1. Build Balanced Tree of Domain
TCRL-Theoretic:
2. Consider State Dynamics Aggregation
Only By Depth
TCRL-Theoretic:
2. Consider State Dynamics Aggregation Only By Depth

1. No clustering

\[
P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)}
\]

\[
P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, \ldots, \alpha_n) d\theta
\]
TCRL-Theoretic:
2. Consider State Dynamics Aggregation
   Only By Depth

1. No clustering
2. Clustered by Parents

\[
P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)}
\]

\[
P(D|C) = \int P(D|\theta)P(\theta|\alpha_1, ..., \alpha_n) d\theta
\]
TCRL-Theoretic:

2. Consider State Dynamics Aggregation
Only By Depth

1. No clustering
2. Clustered by Parents
3. Clustered by Grandparents
...

Choose among this logarithmic number of options using Bayes’ Rule

But, **not a greedy approximation** as clustering decisions are independent.
Thompson Clustering for Reinforcement Learning

• Define original state and action space
• Prior over state clusters/aggregations & params
• Sample state aggregation for each action from prior (using Theoretic or Relaxed approach) and sample parameters for aggregations
• Compute optimal policy for those parameters
• Act
• Update posterior over parameters given data
Thompson Clustering for RL: TCRL-Theoretic has Bounded Bayesian Regret

• Episodic regret definition

\[ R(T) = \sum_{e=1}^{[T/\tau]} V^* - V_{\pi_e} \]

• Thm: TCRL-Theoretic has Bayesian regret \( \leq \)

\[ O((r_{max} - r_{min})\tau |S| \sqrt{|A|T \log(|S||A|T))} \)
Thompson Clustering for RL: TCRL-Relaxed Guaranteed to Still Asymptotically Converge to Optimal Policy
Alternatives

- Best of Sampled Set, BOSS (Asmuth et al. 2009)
  - Bayesian prior
  - Solve with MCMC
  - Very general, computationally expensive, so get approximate solution
TCRL-Relaxed $\geq$ MCMC Approach & Computationally Cheaper
6 Arms Domain
All States are Different for >= 1 Action
All States are Different for >= 1 Action But Never Converge to 6 State Rep. Why?
All States are Different for $\geq 1$ Action
But Never Converge to 6 State Rep. Why?

Never worth separating s-a pairs that don’t yield high reward!
Thompson Clustering for RL Summary

• Dynamically change abstraction for data have
• Do efficient exploration given explicit representation of uncertainty over abstraction
• Accomplish this by using specific set of reasonable but efficiently to compute relative dynamics outcome abstractions
• For more, see our IJCAI 2016 paper
Still Lots of Work to do on Learning Abstractions to Provably Reduce Data
Need to do Reinforcement Learning in Big Spaces
Summary: Combining Abstraction Learning & Efficient Exploration for RL

• Learning options to speed learning
• Learning state abstractions to speed learning

• Data-dependent abstraction
• Leverage uncertainty over abstraction to reduce data needed to get near-optimal performance