Reinforcement Learning by the People and for the People: With a Focus on Lifelong / Meta / Transfer Learning

Emma Brunskill

Stanford
CS234
Winter 2018
Overview

– Last time: Monte Carlo Tree Search
– This time: Human focused RL
– Next time: Quiz
Some Amazing Successes
What About People?
Reinforcement Learning for the People and By the People

Observation → Action

Reward

Policy: Map Observations → Actions
Goal: Choose actions to maximize expected rewards
– Transfer learning / meta-learning / multi-task learning / lifelong learning for people focused domains
  • Small finite set of tasks
  • Large / continuous set of tasks
Provably More Efficient Learners

– 1\(^{st}\) (to our knowledge) Probably Approximately Correct (PAC) RL algorithm for discrete partially observable MDPs (Guo, Doroudi, Brunskill)
  • Polynomial sample complexity

– Near tight sample complexity bounds for finite horizon discrete MDP PAC RL (Dann and Brunskill, NIPS 2015)
Limitations of Theoretical Bounds

• Even our recent tighter bounds suggest need ~1000 samples per state—action pair
• And state—action space can be big!

\[2^{100}\]
Possible knowledge states
Types of Tasks: All Different
Types of Tasks: All the Same -- Can Share Experience!
Transfer / Lifelong Learning
Finite Set of Tasks: Can Also Share Experience Across Tasks
1st: If Know New Task is 1 of M Tasks, Can That Speed Learning?
Approach 1: Simple Policy Class: Small Finite Set of Models or Policies

- If set is small, finding a good policy is much easier.
RL with Policy Advice

\[ \pi_1 \quad \pi_2 \quad \pi_3 \]
RL with Policy Advice

- Keep upper bound on avg. reward per policy
- Use to optimistically select policy

Azar, Lazaric, Brunskill, ECML 2013
RL with Policy Advice

- Keep upper bound on avg. reward per policy
- Use to optimistically select policy
- Regret bounds indp of S-A space, sqrt(# policies)

Azar, Lazaric, Brunskill, ECML 2013
Alternative Idea: Learning as Classification

- Track L2 error of model predictions of observed transitions \((s,a,r,s')\) in current task
- Use to identify current task as 1 of M tasks

\[<s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, \ldots s_H>\]

Brunskill & Li, UAI 2013
But Where Do These Clustered Tasks Come From?
Personalization & Transfer Learning for Sequential Decision Making Tasks

Possible to guarantee learning speed increases across tasks?
Why is Transfer Learning Hard?

• What should we transfer?
  ○ Models?
  ○ Value functions?
  ○ Policies?
Why is Transfer Learning Hard?

• What should we transfer?
  ○ Models?
  ○ Value functions?
  ○ Policies?

• The dangers of negative transfer
  ○ What if prior tasks are unrelated to current task, or worse, misleading
  ○ **Check your understanding:** Can we ever guarantee that we can avoid negative transfer without additional assumptions? (Why or why not?)
Sample complexity:
number of actions may choose whose value is potentially far from optimal action’s value

Can sample complexity get smaller by leveraging prior tasks?
Example: Multitask Learning Across Finite Set of Markov Decision Processes

Sample a task from finite set of MDPs

Brunskill & Li, UAI 2013
Example: Multitask Learning Across Finite Set of Markov Decision Processes

Act in it for H steps

\langle s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, \ldots s_H \rangle
Example: Multitask Learning Across Finite Set of Markov Decision Processes

Again sample a MDP…

Brunskill & Li, UAI 2013
Example: Multitask Learning Across Finite Set of Markov Decision Processes

Act in it for H steps

$<s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, \ldots s_H>$
Example: Multitask Learning Across Finite Set of Markov Decision Processes

Series of tasks
Act in each task for $H$ steps
Example: Multitask Learning Across Finite Set of Markov Decision Processes

- MDP R
  - $T_R$, $R_R$

- MDP G
  - $T_G$, $R_G$

- MDP Y
  - $T_Y$, $R_Y$

...
Example: Multitask Learning Across Finite Set of Markov Decision Processes

MDP R
T=? R=?

MDP G
T=? R=?
2 Key Challenges in Multi-task / Lifelong Learning Across Decision Making Tasks

1. How to summarize past experience in old tasks?
2. How to use prior experience to accelerate learning / improve performance in new tasks?
Summarizing Past Task Experience

• Assume a finite (potentially large) set of sequential decision making tasks
• Learn models of tasks from data
Latent Variable Modeling

Observation data:

\[ <s_{11}, a_{11}, r_{11}, s_{12}, a_{12}, r_{12}, s_{13}, a_{13}, \ldots, s_{1H}> \]

\[ <s_{21}, a_{21}, r_{21}, s_{22}, a_{22}, r_{22}, s_{23}, a_{23}, \ldots, s_{2H}> \]

\[ <s_{31}, a_{31}, r_{31}, s_{32}, a_{32}, r_{32}, s_{33}, a_{33}, \ldots, s_{3H}> \]

\[ <s_{41}, a_{41}, r_{41}, s_{42}, a_{42}, r_{42}, s_{43}, a_{43}, \ldots, s_{4H}> \]

MDP R

\[ T_R, R_R \]

MDP Y

\[ T_Y, R_Y \]

MDP G

\[ T_G, R_G \]
Latent Variable Modeling

\[ \langle s_{11}, a_{11}, r_{11}, s'_{12}, a_{12}, r_{12}, s'_{13}, a_{13}, \ldots \rangle \]

\[ \langle s_{21}, a_{21}, r_{21}, s'_{22}, a_{22}, r_{22}, s'_{23}, a_{23}, \ldots \rangle \]

\[ \langle s_{31}, a_{31}, r_{31}, s'_{32}, a_{32}, r_{32}, s'_{33}, a_{33}, \ldots \rangle \]

\[ \langle s_{41}, a_{41}, r_{41}, s'_{42}, a_{42}, r_{42}, s'_{43}, a_{43}, \ldots \rangle \]

Oberved data

Latent variable:
Underlying MDP identity

MDP R
\[ T_{R'}, R_R \]

MDP Y
\[ T_Y, R_Y \]

MDP G
\[ T_G, R_G \]
Latent Variable Modeling Background

- Formally hard problem
- Expectation Maximization has weak theoretical guarantees
- Recent finite sample bounds on learned parameter estimates
Separability for Latent Variable Modeling

Assume for any 2 finite state—action MDPs $M_i$ & $M_j$, there exists at least one state—action pair such that

$$||\theta_i(\cdot|s,a) - \theta_j(\cdot|s,a)|| > \Gamma$$

Vector of transition & reward parameters for $(s,a)$ for MDP $M_j$

Note: to guarantee $\epsilon$-optimal performance, very small differences in models are irrelevant. *Implies above property always holds in discrete MDPs for some $\Gamma = f(\epsilon)$*
Implications of Separability for Learning & Representing Task Knowledge

• Assume can visit any part of the decision making task an unbounded number of times
• If time horizon per task sufficiently long, can learn $O(\Gamma)$-accurate task parameters with high probability
→ Can correctly cluster tasks
Recall: Using Task Models to Accelerate Learning in New Task*

- Track L2 error of model predictions of observed transitions \((s,a,r,s')\) in current task
- Use to identify current task as 1 of M tasks

Act in it for H steps

\[<s_1,a_1,r_1,s_2,a_2,r_2,s_3,a_3,\ldots,s_H>\]
Sample Complexity Substantially Improved

Theorem 1 Given any $\epsilon$ and $\delta$, run Algorithm 1 for $T$ tasks, each for $H = O\left(DSA(\max(\frac{1}{\Gamma^2} \log \frac{T}{\delta}, SD^2))\right)$ steps. Then, the algorithm will select an $\epsilon$-optimal policy on all but at most $\tilde{O}\left(\frac{C V_{max}}{\epsilon(1-\gamma)}\right)$ steps, with probability at least $1 - \delta$, where

$$\zeta = O\left(T_1 \zeta_s + \bar{C} \zeta_s + (T - T_1) \frac{\bar{C} D}{\Gamma^2}\right),$$

and $\zeta_s = \tilde{O}\left(\frac{NSAV^2_{max}}{\epsilon^2(1-\gamma)^2}\right)$, with probability at least $1 - \delta$.

• 1\textsuperscript{st} result, to our knowledge, that multi-task learning can provably speed learning in later sequential decision making tasks

Brunskill & Li, UAI 2013 & in prep
Concurrent RL

Or all customers using Amazon, or patients, or robot farm...
Concurrent but Independent
Concurrent but Independent

• Very little prior work on concurrent RL
• Except encouraging empirical paper that might be very useful for customers (Silver et al. 2013)
Concurrent RL in Same MDP

- N copies of same task
- Best possible improvement in how long takes to learn a good policy?

Guo and Brunskill, AAAI 2015
Concurrent RL in Same MDP

• N copies of same task
• Best possible improvement in how long takes to learn a good policy?
• Linear improvement
• Proved this for sample complexity (within minor restrictions)
• Interesting:
  • Needed no explicit coordination
  • Algorithm: concurrent MBIE

Guo and Brunskill, AAAI 2015
Concurrent RL in Finite Set of MDPs

- Task identity unknown
- Just know there is a finite set
- Latent variable modeling!
- Assume separability again

Guo and Brunskill, AAAI 2015
Concurrent RL in Finite Set of MDPs

• For $t=1:T$ steps
  • Explore state-action space in each MDP

• Cluster tasks

• Run concurrent MBIIE in each cluster for all future time steps

Guo and Brunskill, AAAI 2015
• If samples to cluster $<<$ samples to learn optimal policy
  $\approx$ Linear speedup*

*Sample complexity over not sharing data

Guo and Brunskill, AAAI 2015
2 Key Challenges in Multi-task / Lifelong Learning Across Decision Making Tasks

1. How to summarize past experience in old tasks? Latent variable modeling
   - Separability assumption
   - Alternate assumptions?

2. How to use prior experience to accelerate learning / improve performance in new tasks?
Method of Moments for Latent Variable Modeling

• Required # of interaction steps per task is very short

• Need to be able to visit all relevant state/actions during that time
Regret Bounds for Multitask Learning across Latent Bandits

Act in it for $H$ steps

$<a_1, r_1, a_2, r_2, a_3, \ldots s_H>$

Azar, Lazaric & Brunskill, NIPS 2013
Method of Moments to Learn Multitask Latent Bandit Parameters

• Used robust tensor power method (Anandkumar et al. 2014)
• Yields confidence bounds over latent bandit parameters

Azar, Lazaric & Brunskill, NIPS 2013
Using Prior Information to Speed Learning in Latent Bandits

Azar, Lazaric & Brunskill, NIPS 2013
Active Set is Models Compatible with Current Task’s Data

Reward for arm 1

μ

Current task

M1  M2  M3
Latent models

Azar, Lazaric & Brunskill, NIPS 2013
Active Set is Models Compatible with Current Task’s Data

Reward for arm 1

\( \mu \)

Current task  M1  M2  M3

Latent models

Azar, Lazaric & Brunskill, NIPS 2013
Upper Bound is Now Upper Bound of Active Set

Azar, Lazaric & Brunskill, NIPS 2013
Theorem. If tUCB is run over J tasks of n steps, where each task is drawn from a set of models \( \Theta \), then with probability at least \( 1 - \delta \), its cumulative regret is

\[
\mathcal{R}_J \leq JK + \sum_{j=1}^{J} \sum_{i \in A_1^j} \min \left\{ \frac{2 \log \left( \frac{2mKn^2}{\delta} \right)}{\Delta_i(\bar{\theta}^j)^2}, \frac{\log \left( \frac{2mKn^2}{\delta} \right)}{2 \min_{\theta \in \Theta_i^j,+(\bar{\theta}^j)} \hat{\Gamma}_i^j(\theta; \bar{\theta}^j)^2} \right\} \Delta_i(\bar{\theta}^j)
\]

\[
+ \sum_{j=1}^{J} \sum_{i \in A_2^j} \frac{2 \log \left( \frac{2mKn^2}{\delta} \right)}{\Delta_i(\theta^j)},
\]

where \( K = \# \) arms, \( A_1^j \) = the set of best arms of models that can be discarded during task j, \( A_2^j \) = the set of best arms of models that cannot be discarded during task j, \( \delta = \# \) of models

Azar, Lazaric & Brunskill, NIPS 2013
Converges to Regret as if Knew Models!

Theorem. If tUCB is run over J tasks of n steps, where each task is drawn from a set of models $\Theta$, then with probability at least $1 - \delta$, its cumulative regret is

$$\mathcal{R}_J \leq JK + \sum_{j=1}^{J} \sum_{i \in A_1^j} \min \left\{ \frac{2 \log \left( \frac{2mKn^2}{\delta} \right)}{\Delta_i(\bar{\theta}^j)^2}, \frac{\log \left( \frac{2mKn^2}{\delta} \right)}{2 \min_{\theta \in \Theta_i} \Gamma_i^j(\theta; \bar{\theta}^j)^2} \right\} \Delta_i(\bar{\theta}^j)$$

where $K = \#$ arms, $A_1^j = \{ \text{the set of best arms of models that can be discarded during task } j \}$, 

$\& m = \#$ of models

Azar, Lazaric & Brunskill, NIPS 2013
Multitask Learning & Partial Personalization: Additional Work

• Separability assumptions
  – Concurrent RL (Guo & B., AAAI 2015)
  – Multi-task RL options learning (Li & B. ICML 2014)
  – Continuous-state multi-task RL (Liu, Guo & B. AAMAS 2016 16)

• Method of moments
  – Contextual latent bandits
Offline Evaluation of Online Latent Contextual Bandit for News Personalization

Zhou and Brunskill IJCAI 2016
Allow for smooth linear parameterization of dynamics model

\[(s'_{d} - s_{d}) \sim \sum_{k}^{K} z_{kad} w_{kb} f_{kad}(s) + \epsilon\]

\[\epsilon \sim N(0, \sigma_{nad}^{2})\]
Hidden Parameter MDPs ++

- Use Bayesian Neural Nets for dynamics
- Benefits for HIV Treatment simulation
- Each episode new patient

Transfer / meta learning is useful broadly in tasks involving people
Deep reinforcement learning to find good shared representation (Finn, Abbeel, Levine ICML 2017)
Fast transfer by encouraging shared representation learning across tasks
Open Issues

• What if the domains have different state or action spaces?
• When do we need new models or policies?
  • How do we identify when not to transfer?
• A number of the algorithms & results above combined ideas from multiple parts of the class
  • Sample efficient learning
  • Batch reinforcement learning
  • Generalization
• Many important additional challenges, in particular for human focused RL
  • What is the reward?
  • Moving beyond expectation / safe RL
  • Trustworthy and interpretable RL
What You Should Know From Today

- List the terms used to describe sharing knowledge as learn across tasks (transfer / lifelong / meta learning)
- Define negative transfer
- Be able to give at least one example application where transfer learning could be useful
Summary

Next time: quiz

2 sided page of notes allowed
Q. Thinking about Reinforcement Learning (select which ones are true):
(a) The maximization of the future cumulative reward allows to Reinforcement Learning to perform global decisions with local information
(b) Q-learning is a temporal difference RL method that does not need a model of the task to learn the action value function
(c) Reinforcement Learning only can be applied to problems with a finite number of states
(d) In Markov Decision Problems (MDP) the future actions from a state depend on the previous states

Q. Thinking about reinforcement learning which one (only 1) of the following statements is true:
(a) Estimation using Dynamic Programming is less computational costly than using Temporal Difference Learning
(b) Estimating using Montecarlo methods has the advantage that it is not needed to have absorbent states in the problem
(c) Temporal Difference learning allows on-line learning and Montecarlo methods need complete training sequences for estimation
(d) Dynamic Programming and Montecarlo methods only work if we know the transitions probabilities for the actions and the reward function
Q. In RL the discount factor (select all that are true)
A. Is specified in the interval $[-1,0]$
B. Is important for convergence
C. Adjusts the balance between immediate and delayed rewards
Problems in Related Classes (Of Similar Difficulty Level)

Pacman is the model of rationality and seeks to maximize his expected utility, but that doesn’t mean he never plays games.

(a) [3 pts] Q-Learning to Play under a Conspiracy. Pacman does tabular Q-learning (where every state-action pair has its own Q-value) to figure out how to play a game against the adversarial ghosts. As he likes to explore, Pacman always plays a random action. After enough time has passed, every state-action pair is visited infinitely often. The learning rate decreases as needed. For any game state $s$, the value $\max_a Q(s, a)$ for the learned $Q(s, a)$ is equal to (for complete search trees)

- The minimax value where Pacman maximizes and ghosts minimize.
- The expectimax value where Pacman maximizes and ghosts act uniformly at random.
- The expectimax value where Pacman plays uniformly at random and ghosts minimize.
- The expectimax value where both Pacman and ghosts play uniformly at random.
- None of the above.

(b) [3 pts] Feature-based Q-Learning the Game under a Conspiracy. Pacman now runs feature-based Q-learning. The Q-values are equal to the evaluation function $\sum_{i=1}^n w_i f_i(s, a)$ for weights $w$ and features $f$. The number of features is much less than the number of states. As he likes to explore, Pacman always plays a random action. After enough time has passed, every state-action pair is visited infinitely often. The learning rate decreases as needed. The value $\max_a Q(s, a)$ for the learned $Q(s, a)$ is equal to (for complete search trees)

- The minimax value where Pacman maximizes and ghosts minimize and the same evaluation function is used at the leaves.
- The expectimax value where Pacman maximizes and ghosts act uniformly at random and the same evaluation function is used at the leaves.
- The expectimax value where Pacman plays uniformly at random and ghosts minimize and the same evaluation function is used at the leaves.
- The expectimax value where both Pacman and ghosts play uniformly at random and the same evaluation function is used at the leaves.
- None of the above.
(c) [2 pts] A Costly Game. Pacman is now stuck playing a new game with only costs and no payoff. Instead of maximizing expected utility $V(s)$, he has to minimize expected costs $J(s)$. In place of a reward function, there is a cost function $C(s, a, s')$ for transitions from $s$ to $s'$ by action $a$. We denote the discount factor by $\gamma \in (0, 1)$. $J^*(s)$ is the expected cost incurred by the optimal policy. Which one of the following equations is satisfied by $J^*$?

- $J^*(s) = \min_a \sum_{s'} [C(s, a, s') + \gamma \max_{a'} T(s, a', s') \cdot J^*(s')]$
- $J^*(s) = \min_{s'} \sum_a T(s, a, s') [C(s, a, s') + \gamma \cdot J^*(s')]$
- $J^*(s) = \min_a \sum_{s'} T(s, a, s') [C(s, a, s') + \gamma \cdot \max_{s'} J^*(s')]$
- $J^*(s) = \min_{s'} \sum_a T(s, a, s') [C(s, a, s') + \gamma \cdot \max_{s'} J^*(s')]$
- $J^*(s) = \min_a \sum_{s'} T(s, a, s') [C(s, a, s') + \gamma \cdot J^*(s')]$
- $J^*(s) = \min_a \sum_{s'} \sum_a [C(s, a, s') + \gamma \cdot J^*(s')]$

(d) [2 pts] It’s a conspiracy again! The ghosts have rigged the costly game so that once Pacman takes an action they can pick the outcome from all states $s' \in S'(s, a)$, the set of all $s'$ with non-zero probability according to $T(s, a, s')$. Choose the correct Bellman-style equation for Pacman against the adversarial ghosts.

- $J^*(s) = \min_a \max_{s'} T(s, a, s') [C(s, a, s') + \gamma \cdot J^*(s')]$
- $J^*(s) = \min_{s'} \sum_a T(s, a, s') \cdot [\max_{s'} C(s, a, s') + \gamma \cdot J^*(s')]$
- $J^*(s) = \min_a \min_{s'} [C(s, a, s') + \gamma \cdot \max_{s'} J^*(s')]$
- $J^*(s) = \min_a \max_{s'} [C(s, a, s') + \gamma \cdot J^*(s')]$
- $J^*(s) = \min_{s'} \sum_a T(s, a, s') \cdot [\max_{s'} C(s, a, s') + \gamma \cdot \max_{s'} J^*(s')]$
- $J^*(s) = \min_a \min_{s'} T(s, a, s') [C(s, a, s') + \gamma \cdot J^*(s')]$