Learning for policy improvement

Geoffrey J. Gordon
Microsoft Research Montreal
Carnegie Mellon University

joint work w/ Wen Sun

Learning from $\ll 10^9$ steps

- Many successes of RL: need billions of training examples
- What if experience is expensive?
  - e.g., robot in real world
  - e.g., expert supervision
  - e.g., personalization
  - e.g., safety
- Idea: replace "SGD-like" algorithms (e.g., policy gradient) with algorithms based on *policy iteration*
- Different tradeoff between computation and data
  - compute better update directions, take bigger steps
  - "think before you act"

Motivation

- Consider TD(λ) (and later algorithms based on it, like DQN)
- Each iteration: one new experience, one step of SGD
  - couples optimization efficiency and sample efficiency
  - if optimizer is slow, uses more data
- What if we did more computation on a minibatch of samples to determine a better update direction?
  - if done well: bigger updates, fewer total samples
  - if done poorly: don't update policy often enough, collected data is less relevant

(Exact) policy iteration

- Do at least once:
  - for all states s, actions a
    - calculate current total exp. cost $Q^\pi(s, a)$, value $V^\pi(s) = E_{a\sim\pi(s)}[Q^\pi(s, a)]$, and (dis)advantage $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ *// evaluate*
  - choose $\pi^{new}(s) = \text{argmin}_a A^\pi(s, a)$ *// improve*

- Doesn't work in a real-size problem:
  - must sample (s, a) rather than iterating over all
  - can't calculate A^π exactly, must estimate somehow
  - can't choose new policy freely, must work in some hypothesis class

Approximate policy improvement

(meta-algorithm)

- Do at least once:
  - estimate $A^\pi(s, a)$ *// evaluate*
  - train π' to achieve low $E[A^\pi(s, a)]$
  - adjust π toward π' to get π^{new} *// improve*

- To instantiate: way to estimate $A^\pi(s, a)$, train π', update π^{new}
  - also starting π, stopping criterion

Simple analysis of approx PI

- Guarantee: cost of π^{new} is $V^{\pi}(s_0) + T\, E_{new}[A^{\pi}(s, a)]$
  - performance difference lemma
    - simple proof by telescoping sum: if we follow π we expect $V^{\pi}(s_0)$; each time we instead take an action $a \neq \pi(s)$ we gain/lose $A^{\pi}(s, a)$
    - improvement when $E_{new}[A^{\pi}(s, a)] < 0$

      (i.e., π improvable, hypothesis class rich enough, and training succeeds)

- Difficulty: expectation is under distribution of (s, a) from π^{new} (not the distribution we used to collect data)
- Can we develop algorithms that guarantee improvement (w/ assumptions) despite this difficulty? Yes…

Two routes to improvement

- Small updates: if new policy is close enough to old, then difference between new/old gradients doesn't matter
  - "SGD-like" analysis
  - pro: frequent updates mean we can get to relevant policies faster
- Big updates: if we can guarantee $E_{new}[A^{\pi}(s, a)] < 0$ whp, doesn't matter how close new/old policies are
  - pro: might be able to do better optimizing each minibatch
- Range of algorithms, from aggressive to deliberate updates

Ex: natural policy gradient

- Exponential family policy $\pi(s, a) \sim \exp(\phi(s, a) \cdot w)$
  - features $\phi(s, a)$, weights w, probs. normalized to sum to 1 at each s
- Evaluation step: fit $A^\pi(s, a) \approx \phi(s, a) \cdot v$ by regression
  - compatible function approximation (same ϕ)
  - data: run π, collect $\phi(s, a) \rightarrow$ [total cost after taking (s, a)] $- \hat{V}^\pi(s)$
- Improvement/update step: boosting
  - $w \leftarrow w - \eta v$ (step size η)
  - i.e., scale $\pi(s, a)$ by $\exp(-\eta \phi(s, a) \cdot v)$, then renormalize
- Analysis: v estimates *natural gradient* of total cost wrt w
  - improvement for small enough η, vs. noise level (batch size) and smoothness

try all acts, or try a random one and use IPS

ok to use any estimate, even a biased one

Variation: policy boosting

- Functional gradient version of same algorithm
- Instead of linear model $\boldsymbol{\phi}(s, a) \cdot v$, fit $A^{\pi}(s, a)$ from any function class (decision trees, deep nets, …) — say, at iteration t, $A_t(s, a)$
- Policy proportional to $\exp(-\boldsymbol{\eta}_1 A_1 - \boldsymbol{\eta}_2 A_2 - \ldots)$

Ex: conservative policy iteration

- Policy: mixture of classifiers from hypothesis space *H*
- Evaluation/improvement step: cost-sensitive classification
  - train π' from *H* to minimize E[A^{π}(s, a)] with s ~ π, a ~ π'
  - sample s ~ π, a ~ arbitrary, record advantage A^{π}(s, a) and score 1 / P(a | s)
    - e.g., a ~ π and argmax regression: same setup as for natural gradient
- Update step: mixture
  - $\pi(s, a) \leftarrow (1-\eta)\ \pi(s, a) + \eta\ \pi'(s, a)$
- Analysis: π' minimizes dot product w/ functional policy gradient
  - so, update step is Frank-Wolfe
  - nonconvex objective, but convergence guaranteed with, e.g., $\eta = 1/\text{iter}$

Connection: imitation learning

- Let π be the expert policy, run one iteration of approx. policy improvement to find π^{new} that does at least as well
- Note: if expert is suboptimal, learner might do strictly better
  - in fact, a single step of policy iteration is often very powerful
  - e.g., suggests that a *random* expert can be good enough for successful IL, as long as it reaches goals occasionally

Ex: AggreVaTe

- Policy: mixture of classifiers from hypothesis space *H*
- Evaluation/improvement step: cost-sensitive *no-regret* classification
  - train π' from $\Delta(H)$ to minimize $E_{\pi'}[A^{\pi}(s, a)]$: i.e., on its *own* distribution
- Update step:
  - slowly mix from expert to learned policy (original paper)
  - or wait and set $\pi \leftarrow \pi$' when confident of improvement (justified by perf. diff. lemma)

Update step

- If we mix over to new policy slowly enough, improvement is guaranteed (analysis in original AggreVaTe paper)
- If we switch all at once, can be confident of improvement when observed advantage is sufficiently negative
  - compare no-regret guarantee to performance difference lemma

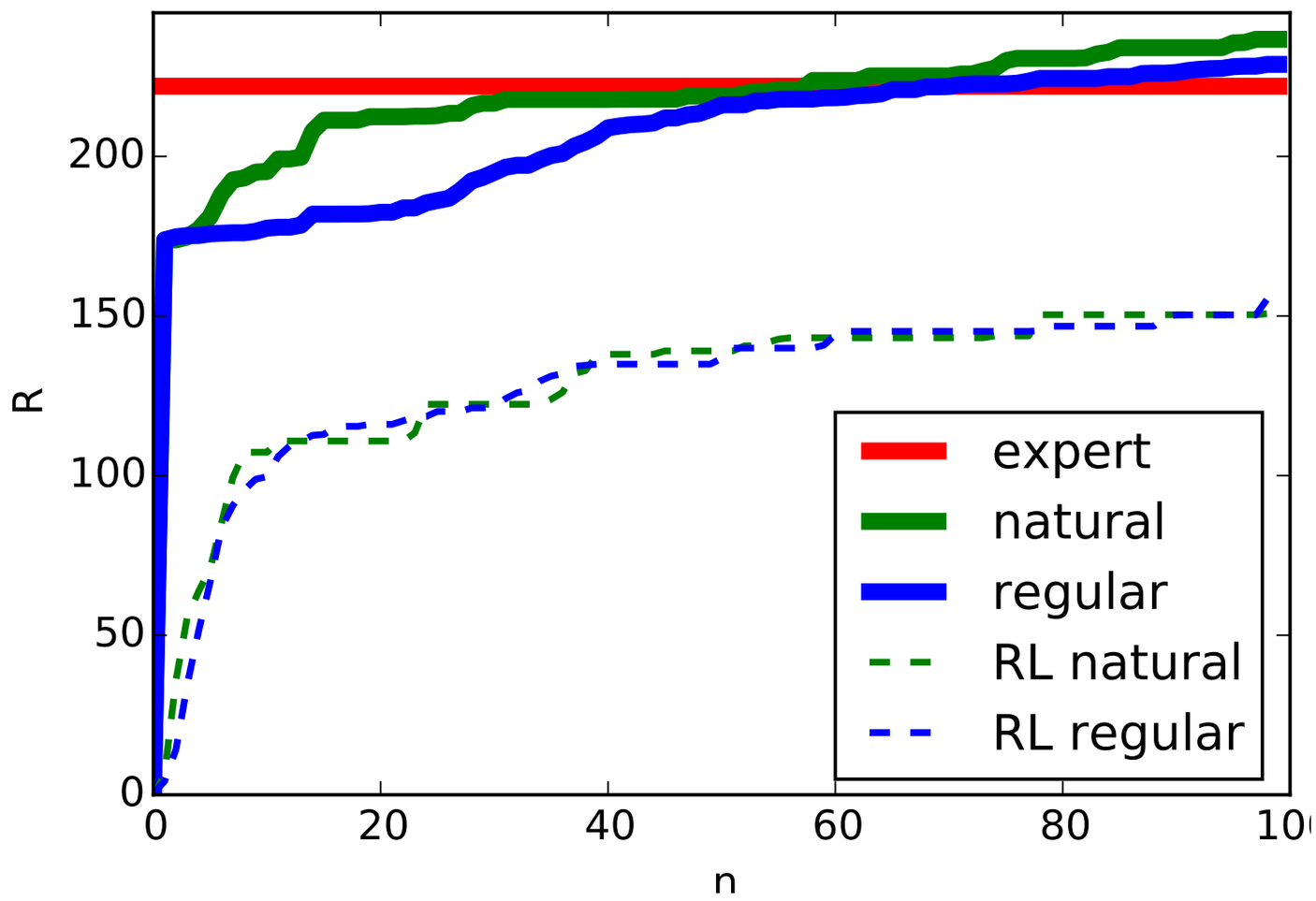
    $V_{\pi'}(s_0) - V_{\pi}(s_0) = T\, E_{\pi'}[A^{\pi}(s, a)]$
- Either way, don't have to use mixture policy (random selection from iterations of no-regret)
  - best component of the mixture is at least as good as the mixture itself

Variation: AggreVaTeD

- Pick online gradient descent as no-regret learner
- Pick a deep net as function representation (even though no-regret property is not guaranteed in this case)
- Ignore the expert after initialization
  - learn initial policy by behavioral cloning
  - then aggressively update expert to be current policy (small minibatches)

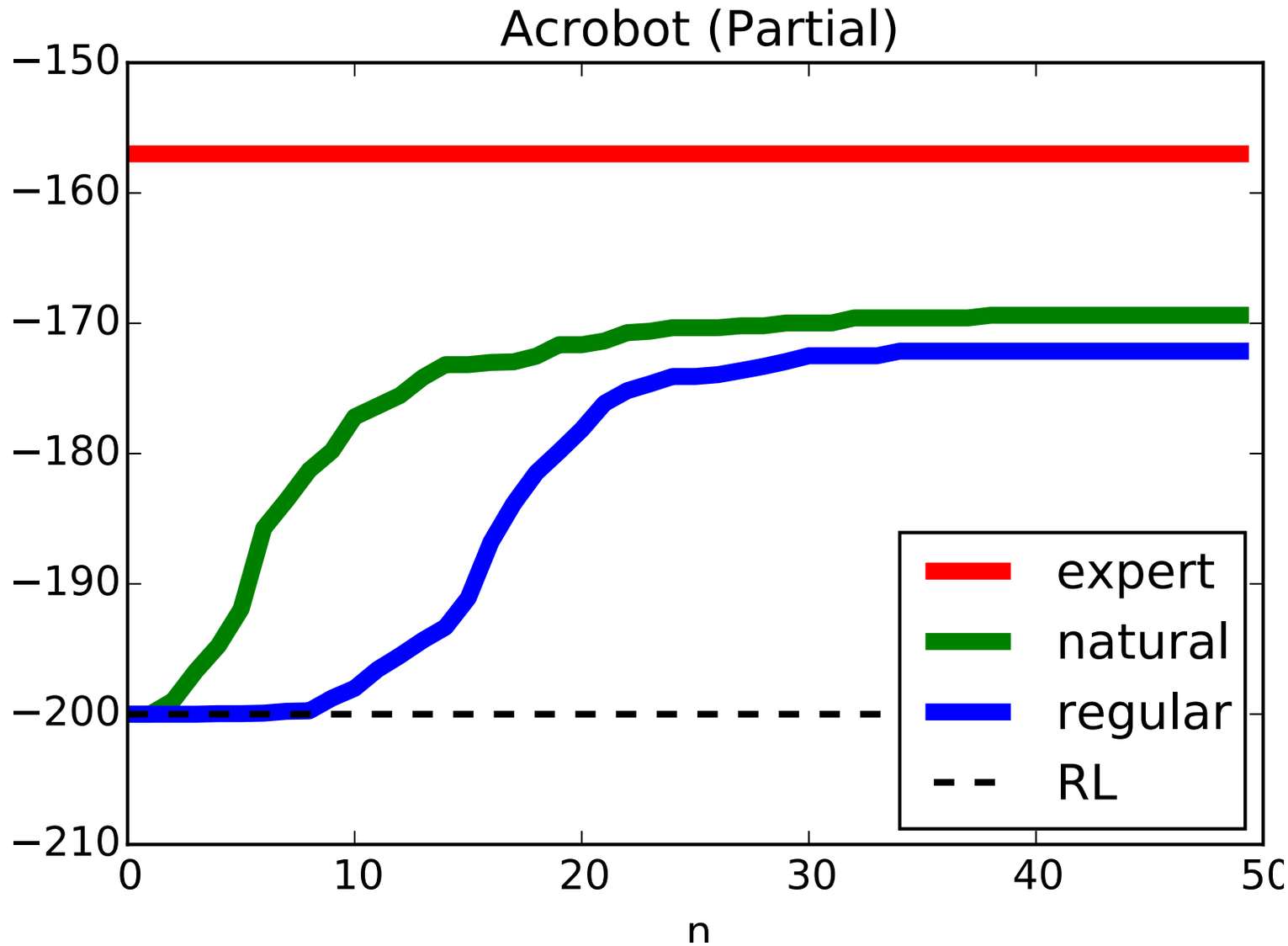
AggreVaTeD on AI gym 2d walker

- Single PI step, starting from policy from TRPO (overnight)
- AggreVaTeD (regular and natural gradient versions) improves much faster than from-scratch RL, beats original expert



AggreVaTeD on POMDP

- Fun use of IL: let the expert cheat
- POMDP version of acrobot (can see only joint angles not angular velocities)
- Expert sees everything



Summary

- Sample-efficient RL via approximate policy improvement
- Family of RL algorithms (different ways to estimate advantages, represent step directions, improve policies; aggressive to deliberate updates)
  - gave several examples
  - many more seem possible
- Future work
  - more thorough exploration and analysis of possibilities
  - combine these algorithms with exploration mechanisms