ACRL (16-899C, S09) Lecture #20 (March 31, 2009)

Improvements to Policy Gradient Method and Policy Iteration

Lecturer: Drew Bagnell *Scribe: Bryan Wagenknecht*

1 Recap REINFORCE

$$\nabla J_\pi = \sum_s P^\pi(s) \sum_a \pi(a|s) \cdot \nabla \log \pi(a_t|s_t) \cdot Q^\pi(s_t, a_t) \tag{1}$$

2 Problems with REINFORCE

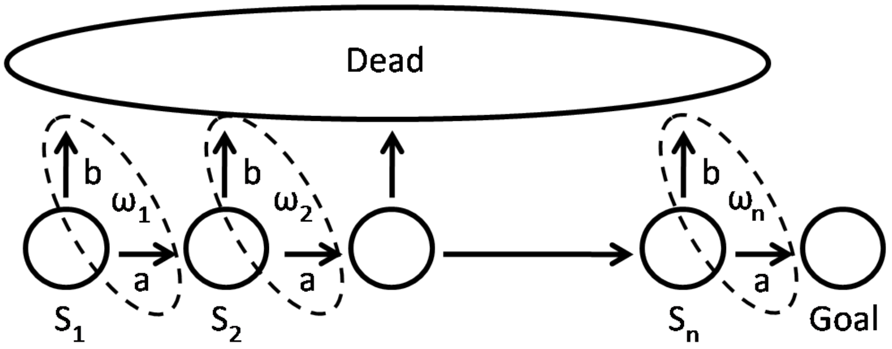

Figure 1: Diagram of cliff-walking with 2 possible actions and a parameter that determines the action at each state

- Need lots of samples to get reward (maybe exponentially many with length of process)
- With random policy, gradient is really small w.r.t. parameters (ω_n) affecting states near end of sequence because $P(s_n)$ is really small
- Gradient is scaled wrong!

3 Improvement: Natural REINFORCE

How do gradient updates work again?

- Within a small ball of parameter space, the gradient is the best direction to move to improve the reward

- Normally gradient seeks to make biggest increases to reward while making the smallest change to parameters as possible
- That makes less sense here where some parameters affect future states much more than others
  - It's actually ok to make big changes to parameters at the end of a sequence

What we really want is to make big increases to reward while making only small changes to *distribution of states* $P^\pi(s)$, we measure this change using KL-divergence.

$$\max J(\omega + \Delta\omega) \qquad s.t.\ KL(P^\omega(s)|P^{\omega+\Delta\omega}) \le \epsilon \tag{2}$$

$(\Delta\omega)^T G(\Delta\omega) \le \epsilon$ is a quadratic approximation to the KL-divergence condition where G is the Fisher Information Matrix (defined below).

Then set up Lagrange formulation of the optimization using linear expansion of $J(w + dw)$

$$L(\Delta\omega) = \nabla J^T \delta\omega - \lambda(\Delta\omega^T G \Delta\omega - \epsilon) \tag{3}$$

$$\frac{\partial L}{\partial \Delta\omega} = 0 = \nabla J^T - 2\lambda \Delta\omega^T G \tag{4}$$

So then direction of improvement (up to a scalar) is

$$\Delta\omega = -G^{-1}\nabla J \tag{5}$$

which is also known as "Riemannian", "Covariant", or "Natural" gradient ascent. If $\Delta\tilde{w}$ was the update from REINFORCE, Natural REINFORCE has the update $\Delta w = G^{-1}\Delta\tilde{w}$

Key:
- Make small changes to $P(s)$
- The less you change $P(s)$, the more you can trust Q, and the more accurate is the linearization of $J(w + \Delta w)$ used in the optimization

So what is G (Fisher Information Matrix)?

$$G = E_{P(s)\pi(a|s)}\left[\nabla \log \pi(a|s)\,[\nabla \log \pi(a|s)]^T\right] \tag{6}$$

Steps for computing G:

1. $G = I$
2. Loop T times:
   (a) Run policy π_w
   (b) $z = \nabla log \pi_w$ (vector)
   (c) $G+ = zz^T$
3. $G = G/T$

4 Actor - Critic Methods

Instead of using sum of actual rewards, we can estimate Q^π with our favorite methods

- TD(0) update
- Least squares fit: $\min\left(\sum \omega_i f(s,a) - [r_1 + r_2 + ...]\right)^2$

This used to cause problems because there were big errors in your learned Q at states you haven't visited. Now we use Q_est in the REINFORCE equation and as long as we have a good estimate of Q over the current $P(s)$, we have a good estimate of the gradient.

$$\nabla J = E_{P(s),\pi(a|s)}\left[\nabla log \pi(a|s) \cdot Q^\pi_{est}(s,a)\right] \tag{7}$$

We don't care that Q_{est} is bad in unvisited states because the ∇J equation takes the expectation over $P(s)$, which places very small weight on the bad sections of Q_{est}.

Natural Actor - Critic:

- Fusion of Actor-Critic and Natural Gradient
- Use estimates of Q rather than actual sum of future rewards
- Compute gradient direction using G matrix as before

5 Key Lessons from REINFORCE:

- State distribution matters a lot
  - API/AVI problems stem from arbitrarily large changes in $P(s)$
  - REINFORCE changes $P(s)$ slowly, acts only in "trust region" around current $P(s)$
- Downsides of REINFORCE
  - Requires stochastic policies
  - Can be slower than API, but more stable (regular REINFORCE can be painfully slow)
  - Not a batch algorithm, must continuously interact with the simulation
  - Can still get stuck in local minima

6 Improvement to Policy Iteration

Recap of old PI:

1. Start with π^0
2. Learn Q^{π^0} by:

   (a) Run π^0, accumulate sample data
   (b) Perform regression on sample data:
      i. $(s, a) \to \sum_t r_t$
      ii. $(s, a) \to r + E_{\pi(a'|s')}\left[Q^{\pi^0}(s', a')\right]$

3. Update: $\pi^{i+1} = \arg\max_a Q^{\pi^i}(s, a)$

New "Conservative Policy Iteration":

1. Start with baseline distribution $P_0(s)$
2. Learn Q^{π^0} by:
   (a) Starting from state sampled from $P_0(s)$
   (b) Run π^0 but sample a's at random, accumulate sample data
   (c) Perform regression
3. Update: $\pi^{i+1} = \alpha\left[\arg\max_a Q^{\pi^i}(s, a)\right] + (1 - \alpha)\pi^i$

This means the updated policy is to stochastically select the new optimized policy with probability alpha and otherwise use the previous policy (probability 1-α).

- This effectively limits the change in $P(s)$ when α is small
- In fact, for very small α ($\alpha \to 0$) we are guaranteed to be heading uphill
- Does add complexity of $O(N)$ because you ideally have to remember every previous policy