Reinforcement learning and MDPs: recap and odds & ends

15-859(A) 03/30/04

RL and MDP recap

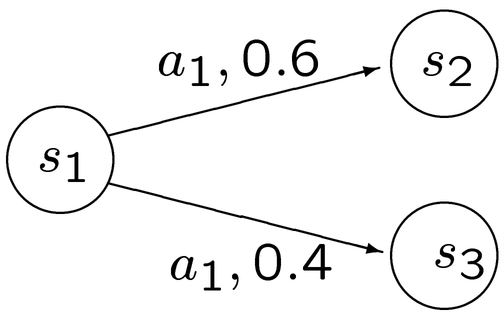
General scenario: We are an agent in some state. Have observations, perform actions, get rewards. (See lights, pull levers, get cookies)

Markov Decision Process: like DFA problem except we'll assume:

- Transitions are probabilistic. (harder than DFA)
- Observation = state. (easier than DFA)

  Assumption is that reward and next state are (probabilistic) functions of current observation and action only.

- Goal is to learn a good strategy for collecting reward, rather than necessarily to make a model. (different from DFA)



What exactly do we mean by a good strategy?

Several notions of what we might want a learned strategy to optimize:

- Expected reward per time step.
- Expected reward in first t steps.
- Expected discounted reward: $r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots$ ($\gamma < 1$)

We will focus on this last one.

- Why γ^i, and not, e.g., $1/i^2$?

One answer: makes it *time independent*. In other words the best action to take in state s doesn't depend on when you get there.

So, what we are looking an optimal *policy* (a mapping from states to actions).

Q-values

Goal is to maximize discounted reward:

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots,$$

Define: $Q(s,a)$ = expected discounted reward if perform a from s and then follow optimal policy from then on.

Define: $V(s) = \max_a Q(s,a)$.

Equivalent defn: $V(s) = \max_a [R(s,a) + \gamma \sum_{s'} \Pr(s')V(s')]$.

($R(s,a)$ = expected reward for doing action a in state s.)

Why is this OK as a definition?

A: Only one solution.

Can see this either by proving by contradiction, or noticing that if you're off by some amount Δ on the right-hand-side, then you'll be off by only $\gamma\Delta$ on the left-hand-side.

Q-learning

Start off with initial guesses $\widehat{Q}(s,a)$ for all Q-values. Then update these as you travel. Update rules:

- Deterministic world: in state s, do a, go to s', get reward r:

$$\widehat{Q}(s,a) \longleftarrow r + \gamma \max_{a'} \widehat{Q}(s',a')$$

- Probabilistic world: on the tth update of $\widehat{Q}(s,a)$:

$$\widehat{Q}(s,a) \longleftarrow (1-\alpha_t)\widehat{Q}(s,a) + \alpha_t[r + \gamma \max_{a'} \widehat{Q}(s',a')]$$

Idea: dampen the randomness.

$\alpha_t = 1/t$ or similar. With $\alpha_t = 1/t$, you get a fair average of all the rewards received for doing a in state s. If you make more slowly decreasing, you favor more recent r's.

Proof of convergence (deterministic case for simplicity)

Start with some $\hat{Q}$ values. Let: $\Delta_0 = \max_{s,a} |\hat{Q}(s,a) - Q(s,a)|$.

Define a "Q-interval" to be an interval in which every (s,a) is tried at least once.

Claim: after the ith Q-interval, $\max_{s,a} |\hat{Q}(s,a) - Q(s,a)| \leq \Delta_0 \gamma^i$. In addition, during the ith Q-interval, this maximum difference will be at most $\Delta_0 \gamma^{i-1}$.

Proof: Prove by induction. Base case (the very beginning) is OK. After an update in interval i of $\hat{Q}(s,a)$, (let's say that action a takes you from s to s') we have:

$$\begin{aligned}
|\hat{Q}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}(s',a')) - (r + \gamma \max_{a'} Q(s',a')| \\
&= \gamma | \max_{a'} \hat{Q}(s',a') - \max_{a'} Q(s',a')| \\
&\leq \gamma \max_{a'} |\hat{Q}(s',a') - Q(s',a')| \\
&\leq \gamma \cdot \gamma^{i-1} \Delta_0. \quad \blacksquare
\end{aligned}$$

So, to get convergence, pick actions so that #intervals $\rightarrow \infty$.

Does approximating V give an apx optimal policy?

Say we find $\hat{V}$ and use greedy policy π with respect to $\hat{V}$. Does $\hat{V}(s) \approx V(s)$ necessarily imply $V^{\pi}(s) \approx V(s)$ for all states s?

- $V^{\pi}(s)$ is value of s if we follow policy π.
- Let $\epsilon = \max_s |\hat{V}(s) - V(s)|$. Assuming this quantity is small.
- Let $\Delta = \max_s [V(s) - V^{\pi}(s)]$. Want to show this has to be small too.

Does apx V give an apx optimal policy? Yes.

Let s be a state where gap is largest: $V(s) - V^{\pi}(s) = \Delta$. Say opt$(s) = a$ but $\pi(s) = b$. $V^{\pi}(s) = R(s,b) + \gamma \sum_{s'} \mathrm{Pr}_b(s')V^{\pi}(s')$.

Step 1: Consider following π for 1 step and then doing opt from then on. At best this helps by $\gamma\Delta$.

Step 2: So, this implies that

$$\Delta(1-\gamma) \leq [R(s,a) + \gamma \sum_{s'} \mathrm{Pr}_a(s')V(s')] - [R(s,b) + \gamma \sum_{s'} \mathrm{Pr}_b(s')V(s')].$$

Step 3: But, since b looked better according to $\hat{V}$,

$$0 \geq [R(s,a) + \gamma \sum_{s'} \mathrm{Pr}_a(s')\hat{V}(s')] - [R(s,b) + \gamma \sum_{s'} \mathrm{Pr}_b(s')\hat{V}(s')].$$

Step 4: But since $|V(s') - \hat{V}(s')| \leq \epsilon$, by subtracting we get:

$$\Delta(1-\gamma) \leq \gamma[\sum_{s'} \mathrm{Pr}_a(s')\epsilon] + \gamma[\sum_{s'} \mathrm{Pr}_b(s')\epsilon] \leq 2\gamma\epsilon.$$

So, $\Delta \leq 2\epsilon\gamma/(1-\gamma)$.

What if state space is too large to write down explicitly?

In practice, often have large state space. Each state described by set of features.

Much like concept learning except:

- Next example is probabilistic function of action and previous example.
- Most examples don't have labels. Only get feedback infrequently (e.g., when you win the game, reach the goal, etc.).

What if state space is too large to write down explicitly?

Neat idea: Use Q-learning (or TD(λ)) to train standard learning algorithm (neural net or whatever) with "hallucinated" feedback.

Say we're in state s, do action a, get reward r, and go to s'. Use $(1-\alpha)\widehat{Q}(s,a)+\alpha(r+\gamma\widehat{V}(s'))$ as the label. Can think of like training up an evaluation function for chess by trying to make it be self-consistent.

Will this really work? "May work in practice but it will never work in theory".

Depends on how well your predictor can fit the value function. Even if it *can* fit it, you might still get into a bad feedback loop: work by Geoff Gordon on what conditions really ensure things will go well. But in practice, it can work great. E.g., TD-gammon backgammon player.