TTIC 31250
An Introduction to the Theory of Machine Learning

MDPs and Reinforcement Learning

Avrim Blum

05/21/18

MDPs and RL

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:

- Transitions are probabilistic (harder)
- Observation = state (easier)
- Goal is to collect reward rather than build a model (different)

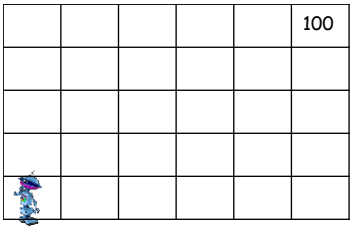
Reward is probabilistic function of current (state,action)

Next state is probabilistic function of current (state,action)

Typical Example

Imagine a grid world with walls.

- Actions left, right, up, down
- If not currently hugging a wall, then with some probability an action takes you to an incorrect neighbor
- Entering top-right corner gives you reward of 100 and then takes you to a random state



Nice features of MDPs

- Like DFA, an appealing model of agent trying to figure out actions to take in the world. Also, incorporates notion of actions being good or bad for reasons that will only become apparent later.
- Probabilities allow to handle imperfect actions or random events (someone picks up the agent and puts it down somewhere else)
- Natural learning algos that propagate "value" backward through state space. If you get reward 100 in state s, then perhaps give value 90 to state s' you were just in.
- Probabilities can to some extent model states that look the same, by merging them, though it's not great for that

What do we mean by a good strategy?

Several notions of what we might want to optimize

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

We will focus on the last one.

- Why γ^t and not, say, $1/t^2$?

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

So, we are looking for an optimal *policy* = an optimal mapping of states to actions.

Q-values

Goal is to maximize expected discounted reward:

$$E[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots]$$

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: $V(s) = \max_a \left[E[r(s,a)] + \gamma \sum_{s'} \Pr(s'|s,a)\, V(s')\right]$.

Why is this OK as a *definition*?

Ans: only one solution. Proof: If two solutions V_1, V_2 differ by at most Δ over all s', then in fact differ by at most $\gamma\Delta$ over all s.

How to solve for Q-values?

Suppose we are given the transition and reward functions. How to solve for Q-values? Two natural ways:

1. Dynamic Programming. Start with guesses $V_0(s)$ for all states s. Update using:

$$V_i(s) = \max_a \left[E[r(s,a)] + \gamma \sum_{s'} \Pr(s'|s,a)\, V_{i-1}(s') \right].$$

Get ϵ-close in $O\left(\frac{1}{1-\gamma}\log\frac{1}{\epsilon}\right)$ steps. If initialize $V_0 = 0$ then $V_i(s)$ represents max discounted reward if game ends in i steps.

2. Linear Programming. Replace "max" with "$\geq$" and then minimize $\sum_s V(s)$ subject to these constraints.

"Dynamic Programming"

Richard Bellman ("Eye of the Hurricane: An autobiography", World Scientific, 1984): An interesting question is, 'Where did the name, dynamic programming, come from?' The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, 'programming.' I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying—I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.

Q-learning

Approach for learning when do not have transition, reward functions. Like asynchronous dynamic programming. Start with initial guesses $\hat{Q}(s,a)$ and then update as you go:

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

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \equiv r + \gamma \hat{V}(s')$$

- Probabilistic world: On update t of $\hat{Q}(s,a)$:

$$\hat{Q}(s,a) \leftarrow (1-\alpha_t)\hat{Q}(s,a) + \alpha_t \left[r + \gamma \max_{a'} \hat{Q}(s',a') \right]$$

$\alpha_t = 1/t$ 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 rewards.

Proof of convergence (deterministic case)

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

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

Claim: after the ith Q-interval, $\max_{s,a} |\hat{Q}(s,a) - Q(s,a)| \leq \Delta\gamma^i$. Also, during the ith Q-interval, this max difference is at most $\Delta\gamma^{i-1}$.

Proof: Proof by induction. Base case (start of first interval) is OK. After an update in interval i of $\hat{Q}(s,a)$ where went to s' we have:

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

So, to get convergence, pick actions s.t. #intervals $\rightarrow \infty$

Does approximating V give an apx optimal policy?

Say we find $\hat{V}$ and use greedy policy π wrt $\hat{V}$. Does $\hat{V}(s) \approx V(s)\ \forall s$ imply that $V^{\pi}(s) \approx V(s)\ \ \forall s$?

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

Will show: $\Delta \leq 2\epsilon\gamma/(1-\gamma)$

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

Let s be a state where gap in quality is largest: $V(s) - V^{\pi}(s) = \Delta$.
Say $opt(s) = a$ but $\pi(s) = b$. $V^{\pi}(s) = E[r(s,b)] + \gamma \sum'_s \Pr(s'|s,b)\, V^{\pi}(s')$.

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

$\Delta(1-\gamma) \leq \left[E[r(s,a)] + \gamma \sum_{s'} \Pr(s'|s,a)\, V(s')\right] - \left[E[r(s,b)] + \gamma \sum_{s'} \Pr(s'|s,b)\, V(s')\right]$.

Step 2: But, since b looked better according to $\hat{V}$,

$0 \geq \left[E[r(s,a)] + \gamma \sum_{s'} \Pr(s'|s,a)\, \hat{V}(s')\right] - \left[E[r(s,b)] + \gamma \sum_{s'} \Pr(s'|s,b)\, \hat{V}(s')\right]$.

Step 3: Using $\left|V(s') - \hat{V}(s')\right| \leq \epsilon$ and subtracting, we get:

$\Delta(1-\gamma) \leq \gamma\left[\sum_{s'} \Pr(s'|s,a)\, \epsilon\right] + \gamma\left[\sum_{s'} \Pr(s'|s,b)\, \epsilon\right] \leq 2\gamma\epsilon$. So, $\Delta \leq 2\epsilon\gamma/(1-\gamma)$.

Other Issues

If state space is large (like a feature vector) might store V values in trained network rather than a lookup table.

Q-learning is like standard learning algo but with "hallucinated" feedback

Like training up an evaluation function in a game. TD-gammon in 1990s and AlphaZero this year.