Reinforcement Learning (Planning Algos)

Friday, March 27, 2020 3:05 PM

Planning Algorithms: given knowledge of $\mathbb{P}(s'|s,a)$ and $\mathbb{E}\, r(s,a)$ $\forall a \in A, s,s' \in S$
how do we compute the optimal policy π^*?

Sometimes called "Dynamic Programming" used very broadly. Means different things to different people

① Value Iteration

In mid '90s, Puterman said this algo was most widely used, due to its simplicity, but he advises against using it (due to slow convergence)

Bellman's Optimality Eq'n:

not a scalar, so 'max' seems ill-defined, but π decouples across s, so interpret component-wise

$$(\forall s \in S) \quad V_{\pi^*}(s) = \max_{a \in A} \left(\mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}(s'|s,a) \cdot V_{\pi}(s') \right)$$

$$= \max_{\pi} \left(R_\pi + \gamma P_\pi \cdot V \right)$$ since choosing action a, based on s, is same as a policy

$$:= \Phi(V)$$ ← looks funny since V depends on π, but OK if we require...

Fixed Pt. Eq'n $\quad V^* = \Phi(V^*)$ (Bellman's Eq'n, restated)

non-linear operator

Algorithm: Value Iteration aka Picard iteration

Initialize V_0 arbitrarily

Iterate $V_{t+1} = \Phi(V_t)$ for $t=1,2,\ldots$

(stopping criteria: $\|V_{t+1} - \Phi(V_t)\| < \frac{1-\gamma}{\gamma} \cdot \varepsilon$

Thm (17.11) Value iteration converges to an optimal V^*

proof Use contraction-mapping (aka Banach or Banach-Picard fixed pt.) theorem

i.e., show $\exists c<1$ st $\forall v,u$ $\|\Phi(v) - \Phi(u)\| \le c\|v-u\|$

(any norm, as long as it's a Banach space... in fact complete metric space works)

We'll show Φ is Lipschitz cts w/ constant $c = \gamma$, and use $\|\cdot\| = \|\cdot\|_\infty$

I'm going to abuse notation to simplify it... you can make it rigorous by fixing a row (i.e., fix s)

$$\Phi(v) - \Phi(u) = \max_{\pi} (R_\pi + \gamma P_\pi v) - \max_{\pi'} (R_{\pi'} + \gamma P_{\pi'} u)$$

$$= R_{\overset{\text{(argmax...)}}{\pi}} + \gamma P_\pi v - \max_{\pi'} (R_{\pi'} + \gamma P_{\pi'} u)$$

$$\le R_\pi + \gamma P_\pi v - (R_\pi + \gamma P_\pi u)$$ since π suboptimal

$$= \gamma P_\pi (v - u)$$

and similarly $\Phi(u) - \Phi(v) \le \gamma P_{\pi'} (u - v)$

so $\|\Phi(v)-\Phi(u)\|_\infty \le \gamma \|P_\pi (v-u)\|_\infty$

$\le \gamma \|P_\pi\|_\infty \cdot \|v-u\|_\infty$ by def. operator norm

$\|P_\pi\|_\infty = 1$ regardless of which π

$= \gamma \cdot \|v-u\|_\infty$ $\square$

How do we get π^* back from V^* ?

Bellman Eq'n,

$$V^*(s) = \max_{a\in A} \underbrace{\mathbb{E}\, r(s,a)}_{\text{known}} + \gamma \sum_{s'\in S} \underbrace{\mathbb{P}(s'|s,a)}_{\text{known}} \cdot \underbrace{V^*(s)}_{\text{known}}$$

assuming $|A|<\infty$

$\Rightarrow$ we can solve the maximization problem to find a and this a is $\pi(s)$.

Value iteration w/ Gauss-Seidel acceleration

if you don't know this, just ignore this part

$\approx$ same cost per iteration, faster convergence

Always converges in our case due to properties of transition matrices, c.f. Puterman Thm 6.3.7

② Policy Iteration

Instead of solving for V and π is implicit, now work directly w/ π

ALGORITHM: Policy Iteration

Initialize π_0 arbitrarily

For $t=1,2,\ldots$*

* Policy iter. terminates in a finite # of steps

Find V_{π_t} by solving $(I-\gamma P_{\pi_t})V = R_{\pi_t}$ // expensive: solve a linear system of equations

if $|S|=n$, policy iter. is $O(n^3)$/iter. value iter. is $O(n^2\cdot|A|)$/iter

$\pi_{t+1} = \arg\max_\pi (R_\pi + \gamma P_\pi \cdot V_{\pi_t})$ // "greedy" update

same as "How do we get back π^* from V^*"

break if $\pi_{t+1}=\pi_t$

You can rewrite as $V_{t+1} = V_t - (\gamma P_{\pi_t} - I)^{-1}(\Phi(V_t) - V_t)$ cf. Puterman

vague for now

so at a fixed pt, $0 = \underbrace{(\gamma P_\pi - I)^{-1}}_{\text{nonsingular}}(\Phi(v) - v) \Rightarrow \Phi(v) = v$ (Bellman's Eq)

also like a Newton method $f(v) = \Phi(v) - v$ ... and in practice, policy iter. converges much faster than value iter.

$f'(v) = (\gamma P_\pi - I)$

A common practical version "modified policy iteration"

solve $(I-\gamma P_{\pi_t})v = R_{\pi_t}$ approximately, using the fact

$(I-\gamma P)^{-1} = \sum_{k=0}^{\infty} (\gamma P)^k$ (Neumann Series)

so $(I-\gamma P)^{-1} R \approx \sum_{k=0}^{d} (\gamma P)^k R$

compute $S = R$
$R_1 = \gamma P\cdot R$
$S \leftarrow S + R_1$
$R_2 = \gamma P \cdot R_1$
$S \leftarrow S + R_2$
...

Monotonicity* property

If $V \ge U$ (meaning $V(s) \ge U(s)\ \forall s$)

then $\Phi(v) \ge \Phi(u)$

* this generalizes 1D monotonicity ($u, v \in \mathbb{R}$, $v \geq u \Rightarrow \Phi(v) \geq \Phi(u)$)

but in a different way than "monotonicity" for operators as used in optimization

monotone: $\langle v-u, \Phi(v)-\Phi(u) \rangle \geq 0$

For our proof, we'll need $V \geq U \Rightarrow (I-\gamma P_\pi)^{-1} V \geq (I-\gamma P_\pi)^{-1} U$ (for $\pi = \pi_t$)

proof: since $\forall \pi$, $\|P_\pi\|_\infty = 1 \Rightarrow$ (Neumann Series converges) (since $\|\gamma P_\pi\|_\infty = \gamma < 1$)

$$(I-\gamma P_\pi)^{-1} = \sum_{k=0}^{\infty} (\gamma P_\pi)^k$$

so $$(I-\gamma P_\pi)^{-1}(v-u) = \sum \underbrace{(\gamma P_\pi)^k}_{\geq 0 \text{ entries}} (v-u) \geq 0 \quad \square$$

Lemma 17.12 If (V_t) constructed via policy iteration, then $V_t \leq V_{t+1} \leq V^*$ $(\forall t)$

proof: because π_{t+1} chosen greedily (to maximize $R_\pi + \gamma P_\pi \cdot V_t$)

$$R_{\pi_{t+1}} + \gamma P_{\pi_{t+1}} \cdot V_t \geq R_{\pi_t} + \gamma P_{\pi_t} \cdot V_t = V_t$$

so $$R_{\pi_{t+1}} \geq (I - \gamma P_{\pi_{t+1}}) V_t$$

hence $$\underbrace{(I-\gamma P_{\pi_{t+1}})^{-1} R_{\pi_{t+1}}}_{V_{t+1}} \geq (\ldots)^{-1}(\ldots) V_t = V_t$$ via monotonicity

$\square$

Thm Policy Iteration converges to an optimal policy (for a finite MDP)

proof: By greedy update, if $V_{t+1} = V_t \Rightarrow$ satisfy Bellman's Eq'n so optimal

By lemma above, $V_t \leq V_{t+1}$, so

V_t not optimal $\Rightarrow V_t < V_{t+1}$

Note for a finite MDP, # possible policies $= |A|^{|S|} < \infty$

and we can't repeat any policies since $V_t < V_{t+1}$

$\Rightarrow \exists$ maximal policy V_{t^*}, and $V_{t^*+1} = V_{t^*} \Rightarrow$ it's optimal. $\square$

Corollary: converge in $\leq |A|^{|S|}$ iterations

or $O\left(\frac{|A|^{|S|}}{|S|}\right)$ using better proof techniques.

③ Linear Programming

Bellman Eq'n: $(\forall s)$ $$V^*(s) = \max_{a \in A} \underbrace{R_a(s)}_{\mathbb{E}\, r(s,a)} + \gamma \sum_{s' \in S} \underbrace{P_a(s'|s)}_{\mathbb{P}(s'|s,a)} \cdot V^*(s')$$

i.e., $\forall a$,
$$V^*(s) \geq R_a(s) + \gamma \sum_{s' \in S} P_a(s'|s) V^*(s')$$

or in matrix notation

$$\vec{V} \geq \vec{R}_a + \gamma P_a \cdot \vec{V} \quad (\forall a)$$

a set of $|S|$ linear inequalities

so $|A|$ of these $|S|$ sets of ineq.

So

Find $\vec{V} \in \mathbb{R}^{|S|}$ st. $\vec{V} \geq \vec{R}_a + \gamma P_a \cdot \vec{V}$ $\forall a \in A$

is a Linear Program (if a finite MDP).

$\exists$ weakly polynomial time algo to solve LP (1980's, ellipsoid, Karmarkar, etc.)

and in fact $\exists$ strongly polynomial time algo to solve the LP arising from MDP (Yinyu Ye 2000's)

In practice, due to extremely large state space S, LP formulation usually not efficient

Summary

1. For small MDP ($|S|, |A|$ reasonably sized),
   we can satisfactorily solve the planning problem

2. Value Iteration: many iterations, cost $O(|S|^2 \cdot |A|)$ / iteration

   Policy Iteration: few iterations, cost $O(|S|^3)$ / iteration

   LP formulation: LP w/ $|S|$ variables, $|S| \cdot |A|$ constraints

   complexity of LP something like variables^3 · constraints^2) I forget exactly (not relevant since many special tricks/algos)

3. As $|S| \to \infty$, need special tricks.
   Obviously still ongoing research

   Backgammon: $|S| = 10^{20}$

   Chess: $|S| = 10^{47}$

   Go: $|S| = 10^{170}$ for 19 × 19 board

   See wikipedia "Game complexity"