"Singularity"

2024年4月22日 星期一 18:17

Again: MDP → Vector in $\mathbb{R}^n$

$s \in \{s_A, s_B\}$, $A \in \{a_A, a_B\}$

$V^\pi(s_A) = R(s_A) + \gamma \mathbb{E}_{s'}[V^\pi(s')]$

$$\begin{cases} V^\pi(s_A) = R(s_A) + \gamma\big(P(s_A|s_A,a_A)V^\pi(s_A) + P(s_B|s_A,a_A)V^\pi(s_B)\big) \\ V^\pi(s_B) = R(s_B) + \gamma\big(P(s_B|s_B,a_B)V^\pi(s_B) + P(s_A|s_B,a_B)V^\pi(s_A)\big) \end{cases}$$

$V^\pi = B^\pi(V^\pi)$ all "walkable" choices expression stacked together

The whole tree is infinite, ∞ # of τ because you can loop around (∞ values)

$V_0^\pi \xrightarrow{B^\pi} V_1^\pi = B(V_0^\pi) \xrightarrow{B^\pi} V_2^\pi = B(V_1^\pi) \to \dots$

(movement that we are gonna study)

$$\begin{aligned} V^\pi(s_0) &= \mathbb{E}_\tau[R(\tau)] && \text{Conceptual} \\ &= \mathbb{E}\Big[\sum_{i=0}^{\infty}\gamma^i R(s_i) \,\Big|\, s_0 = s\Big] && \text{Abstract} \\ &= R(s_0) + \gamma \sum_{i=1}^{\infty} V^\pi(s_i)\cdot \mathbb{P}(s_i|s_0,a_0) && \text{Less Abstract} \end{aligned}$$

Overview, Parallel MDP → Singularity

MDP (Recursive System of equation space)

one global state $\vec{V}$ in MDP

$V(s_1)$

each row is a unique time/point in the global state that may also be used in other part of the equation

$\vec{V}\begin{bmatrix} V(s_0) \\ V(s_1) \\ V(s_2) \\ \vdots \end{bmatrix} \quad B^\pi \quad B^\pi \quad \vec{V}'\begin{bmatrix} V'(s_0) \\ V'(s_1) \\ V'(s_2) \\ \vdots \end{bmatrix}$

Contraction Mapping

$\mathbb{R}^n$ Value Vector Space

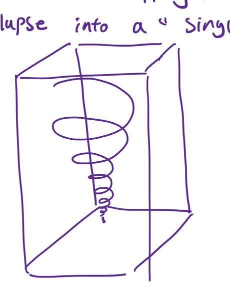
My understanding is actually correct! The wow moment, this is really really cool, once you understand it, you understand it

A different system of equation (or different values at different state)
- different global state
- can use the same policy with another different global state vector

"parallel MDP": potential state of system

"空间+时间" → Vector value space

Contraction mapping shows that all these "parallel universe MDP" actually collapse into a "singularity" that yields the optimal condition of all state globally



one value space vector, one global state that is optimal everywhere

Each row of value vector

Stacking all temporally different values together

J variable

$V(s_1) = R(s_1) + \gamma \mathbb{E}_{s\sim a,M}[V(s_j)]$

value s_1 — All other state

$\vdots$

$V(s_j) = R(s_j) + \gamma \mathbb{E}_{s\sim a,M}[V(s_{Any})]$

J variable big sum

$\vec{V} = B^\pi(\vec{V})$, $\vec{V}' = B^\pi(\vec{V}')$

- Vector of value is recursively defined, but the ultimate vector is values of all states

System of equation formed

This system of equation (value vector) considers all the possible value at each state, which does classify a potential global state condition (All is known)

Policy is a distribution over trajectory

Vector in value space considers all possible state value globally but each π is not a unique vector in value space

Intuition to Contraction mapping & Bellman Update in Vector space

$V \leftarrow B^\pi(V) \in \mathbb{R}^n$

more consistent

one potential/guessed global state

Should be consistent with update in one step (local wise)

make potential states more consistent with bellman update (global wise)

- Update globally in the recursive system → more consistent with bellman update rule's global state (one state to another)

Updating the global potential reality (finding in vector space a potential state that fit better) to more aligning with Bellman update state

Formal Proof: Contraction Mapping

① Can't solve this using this algorithm:

$X = 2x$

$x_0 \to x_1 \to x_2 \to \dots$

$10 \to 20 \to 40 \to \dots$ Blows up (exponentially)

Your algebra does much better, but if your equation is complicated, this algorithm helps

② Can solve this though:

$x = \frac{1}{2}x$

$x_0 \to x_1 \to x_2 \to \dots$

$10 \to 5 \to 2.5 \to \dots$ Converge to zero (pretty quickly, shrinking exponentially)

$f: x \to \frac{1}{2}x \quad \left(\frac{1}{2}\|x_1 - x_2\| \le \|x_1 - x_2\|\right)$

Defining Contraction Mapping

Given a space $\mathbb{R}^n$

$x \xrightarrow{f} f(x)$, $x_1 \xrightarrow{f} f(x_1)$

$k \in [0,1)$

It is contracting mapping when $\forall x, x' \in \mathbb{R}^n$

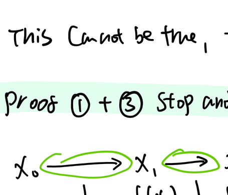
$\|f(x) - f(x')\| < k\|x - x'\|$

Distance between 2 points after mapping < Distance between 2 points before mapping

$V_0^\pi \to V_1^\pi \to \dots \quad B(V_{k+1}) \leftarrow B(V_k)$

Same with V_k, V_k must be the solution (one fixed point of this mapping)

Claim: As long as f is contraction mapping, you can always use this naive algorithm and converge



"Singularity!!"

1. Stop exist
2. Stop at unique point (one vector in value space)
3. Convergence is fast

Proof ② unique stop

If it exist, it has to be unique

Assume is there are 2 unique point x and y, then

$\|f(x) - f(y)\| = \|x - y\| < k\|x - y\|$

This cannot be true, the space does not hold, so proof by contradiction exist

Proof ① + ③ Stop and stop fast

$x_0 \to x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to \dots$

$f(x_0) \quad f(f(x_0))$

$\|x_0 - x_1\| \ge \|x_1 - x_2\| = \|f(x_0) - f(x_1)\|$

Distance must be shrinking, a sequence that converges that is exponentially converging (keep multiplying < 1 number)

Bellman Update in Contraction mapping

Contraction → Best in MDP → Best in Casino

Goal: $V(s_0) \leftarrow B^\pi(V)$ is a contraction

Show $\|B^\pi(V') - B^\pi(V'')\| \le k\|V' - V''\|$

$$\|x\| = \|(x_1,\dots,x_n)\| = \begin{cases} \sqrt{\sum x_i^2} & \|\cdot\|_2 & \text{Euclidean} \\ \sum |x_i| & \|\cdot\|_1 & \text{Manhattan} \\ \max|x_i| & \|\cdot\|_\infty & \text{Infinite} \end{cases}$$

L_2, L_1, x

$\|\cdot\|_p = \left(\sum |x_i|^p\right)^{\frac{1}{p}}$ ← get back

once in the sky

Largest dimension directly sky rock to ∞, then take back

In $\|\cdot\|_{max}$ norm, it is a contraction

2 different V is 2 different policy in MDP

$\|B^\pi(V') - B^\pi(V'')\|_\infty$ (In value space) $\Rightarrow \left[\left|\left(\cancel{R(s_A)} - \gamma \mathbb{E}_{s'}[V'(s')]\right) - \left(\cancel{R(s_A)} + \gamma \mathbb{E}_{s'}[V''(s')]\right)\right| \;\vdots\; \vdots\right]$

In $\mathbb{R}^n$ Parallel MDP space

< 1 value

V is all the value at temporal dynamical state performing the same update B

$\Rightarrow \left[\gamma\left|\sum P(s'|s,\pi(s))\cdot(V'(s') - V''(s'))\right| \;\vdots\; \vdots\right] \le \gamma \max\big(V'(s') - V''(s')\big) = \gamma\|V' - V''\|_\infty$

Returning the B^π equation back to update format, same update, same transition function

γ carries over!

No deducted!

Deducted distance must ≤ max non deducted distance

Always talking about pairs not 1 vector.

The machinery takes in 2 state and see differences

⇒ Intuitively, prob is smaller than zero value, squeezing things together (Gamma makes sure definitely < 1)

⇒ Reward is adding a constant, doesn't change distance

All you're doing is solving $y = 1/2x$ on many dimension

Bellman Max Update is a Contraction Mapping

$V^*(s) \leftarrow R(s) + \gamma \max_{a\in A(s)} \mathbb{E}[V^*(s')]$

⇒ max over bunch of action, each action contracting, max of each action contracting is a contraction

⇒ 2 max operator would yield 2 different things, but all of the space that the max (action not same, can't do previous proof, take max action) can choose from is contraction, so max of it would be contraction

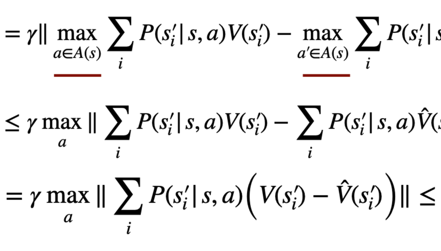
"Max" power is very limited

Formally speaking:

$\|B(V) - B(\hat{V})\|_\infty = \left\|\left(\cancel{R(s)} + \gamma \max_{a\in A(s)} \sum_i (s_i'|s,a)V(s_i')\right) - \left(\cancel{R(s)} + \gamma \max_{a'\in A(s)} \sum_i P(s_i'|s,a')\hat{V}(s_i')\right)\right\|$

How to do with 2 different state? → Actions are different because for it to be max and they are existing in different reality of the MDP parallel condition

$|\max_x f(x) - \max_y g(y)| \le \max_x |f(x) - g(x)|$



Mathematical property: max of $f(x)$ − max $g(x)$ would always be smaller than or equal to max of $f(x) - g(x)$

may not be max!

Real max difference — upper bound

$\|B(V) - B(\hat{V})\|_\infty$

$= \|\left(R(s) + \gamma \max_{a\in A(s)} \sum_i P(s_i'|s,a)V(s_i')\right) - \left(R(s) + \gamma \max_{a'\in A(s)} \sum_i P(s_i'|s,a')\hat{V}(s_i')\right)\|$

$= \gamma\|\max_{a\in A(s)} \sum_i P(s_i'|s,a)V(s_i') - \max_{a'\in A(s)} \sum_i P(s_i'|s,a')\hat{V}(s_i')\|$

$\le \gamma \max_a \|\sum_i P(s_i'|s,a)V(s_i') - \sum_i P(s_i'|s,a)\hat{V}(s_i')\|$

$= \gamma \max_a \|\sum_i P(s_i'|s,a)\left(V(s_i') - \hat{V}(s_i')\right)\| \le \gamma\|V(s_i') - \hat{V}(s_i')\|$

Convergence Rate

$D \to \gamma D \to \gamma^2 D \to \gamma^3 D \to \gamma^{\dots} D$

D: Distance between 2 vectors

Remember upper bounded by $\gamma\|V(s_i') - \hat{V}(s_i')\|$

$D(1 + \gamma^1 + \gamma^2 + \dots) = \frac{D}{1-\gamma}$ Exponential Convergence

Geometric Series — from first step

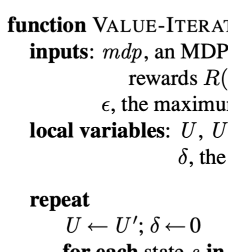
Every step is contraction, whole thing converges, contraction at rate of γ^i

$\|B(V) - B(V')\|_\infty \le \gamma^i \|V - V'\|_\infty$

AI's Perspective

AI thinks very differently than us, they think in this multi parallel mdp vector space, totally in a different space

This is the AI's perspective, different from how we think of search like a tree and the space is there having things worked out in the right way just because of its mathematical structure



$\mathbb{R}^n$

Absorbing vector field

$V^* = B_{max}(V^*) \leftarrow \pi^*$ read off action to take from value, you know which action to take

$V^\pi = B^\pi(V^\pi)$

global optimum state is found!!!

Bellman must satisfy all state, all state must be maximized

One optimum, multiple policy may exist

Very different topological property, not talking about convexity

Look at local → because of structure of MDP, whole thing work

MDP Algorithm

```
function VALUE-ITERATION(mdp, ε) returns a utility function
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a),
            rewards R(s), discount γ
          ε, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
                   δ, the maximum change in the utility of any state in an iteration

  repeat
      U ← U'; δ ← 0
      for each state s in S do
          U'[s] ← R(s) + γ max_{a ∈ A(s)} Σ_{s'} P(s' | s, a) U[s']   "Bellman Update"
          if |U'[s] − U[s]| > δ then δ ← |U'[s] − U[s]|
  until δ < ε(1 − γ)/γ
  return U
```

Algorithm 17.4, AIMA Ch17