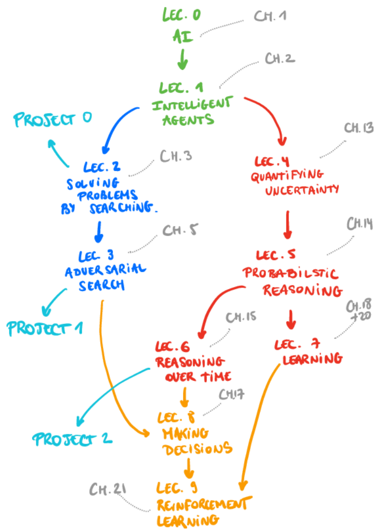
ปัญญาประดิษฐ์ (Artificial Intelligence)

บทบรรยายที่ 8: การตัดสินใจ (Making decisions)

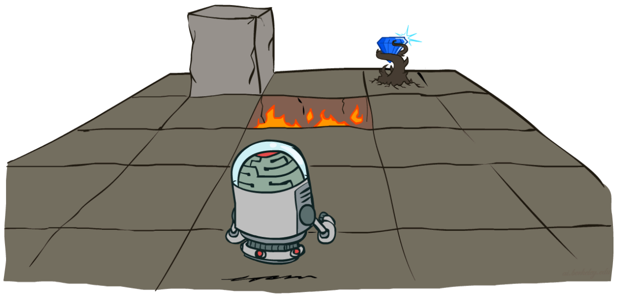
ผศ. ดร. อิทธิพล ฟองแก้ว
[ittipon@g.sut.ac.th]

LIÈGE université



LEC. 0
AI
CH. 1
LEC. 1
INTELLIGENT AGENTS
CH. 2
PROJECT 0
LEC. 2
SOLVING PROBLEMS BY SEARCHING.
CH. 3
LEC. 3
ADVERSARIAL SEARCH
CH. 5
PROJECT 1
LEC. 4
QUANTIFYING UNCERTAINTY
CH. 13
LEC. 5
PROBABILSTIC REASONING
CH. 14
LEC. 6
REASONING OVER TIME
CH. 15
LEC. 7
LEARNING
CH. 18 +20
PROJECT 2
LEC. 8
MAKING DECISIONS
CH. 17
LEC. 9
REINFORCEMENT LEARNING
CH. 21

เนื้อหาวันนี้



การให้เหตุผลภายใต้ความไม่แน่นอน และการ "ตัดสินใจ" (Reasoning under uncertainty and taking decisions):

- Markov decision processes
  - MDPs
  - สมการของ Bellman (Bellman equation)
  - Value iteration
  - Policy iteration
- Partially observable Markov decision processes (POMDPs)

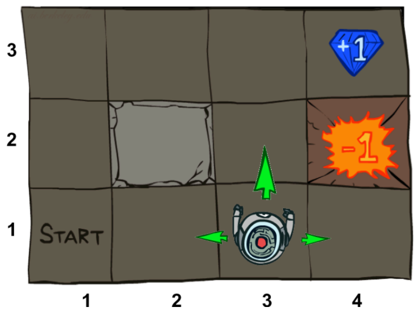
—
เครดิต: CS188, UC Berkeley.

โลกแบบกริด (Grid world)

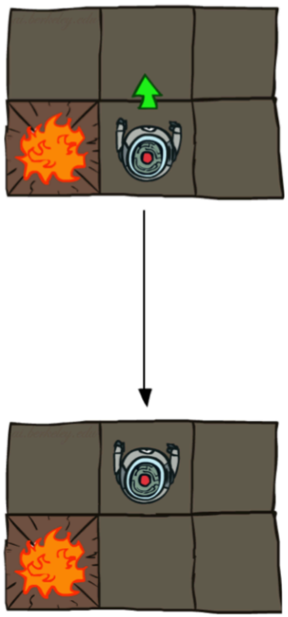
สมมติว่าเอเยนต์ของเราอาศัยอยู่ในสภาพแวดล้อมกริดขนาด 3×4.

- การเคลื่อนที่มีสัญญาณรบกวน: การกระทำไม่เป็นไปตามที่วางแผนเสมอไป
  - แต่ละการกระทำบรรลุผลตามที่ตั้งใจด้วยความน่าจะเป็น 0.8.
  - ที่เหลือ 0.2 การกระทำจะพาเอเยนต์ไปในทิศตั้งฉากกับทิศที่ตั้งใจ (ขวาหรือซ้ายของทิศที่ตั้งใจ)
  - หากมีผนังในทิศทางที่จะถูกพาไป เอเยนต์จะอยู่นิ่ง
- เอเยนต์ได้รับรางวัล (reward) ในแต่ละช่วงเวลา
  - รางวัลเล็กน้อยสำหรับการมีชีวิต (living reward) ในแต่ละก้าว (อาจเป็นลบ)
  - รางวัลใหญ่จะเกิดขึ้นเมื่อจบตอน (ดีหรือไม่ดี)

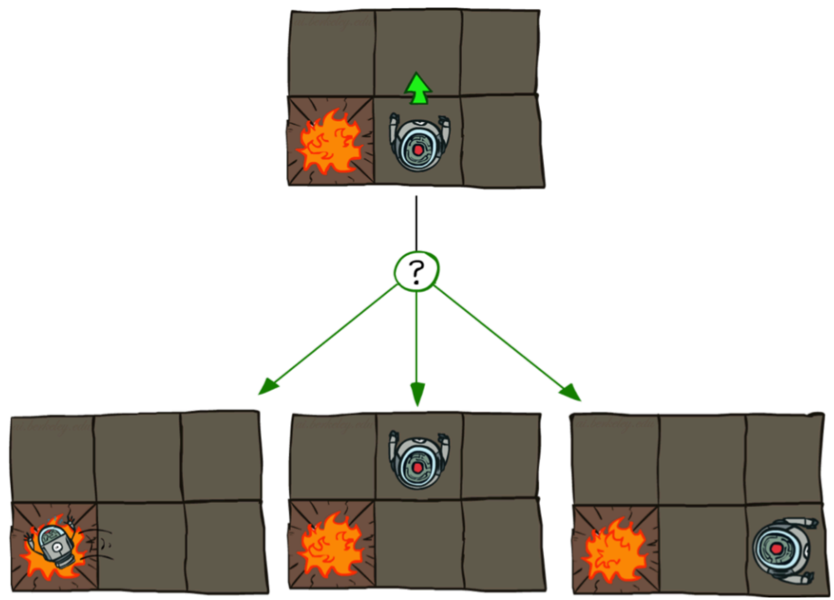
เป้าหมาย: ทำให้ผลรวมของรางวัลสูงสุด



การกระทำเชิงกำหนด (Deterministic actions)



การกระทำเชิงสุ่ม (Stochastic actions)

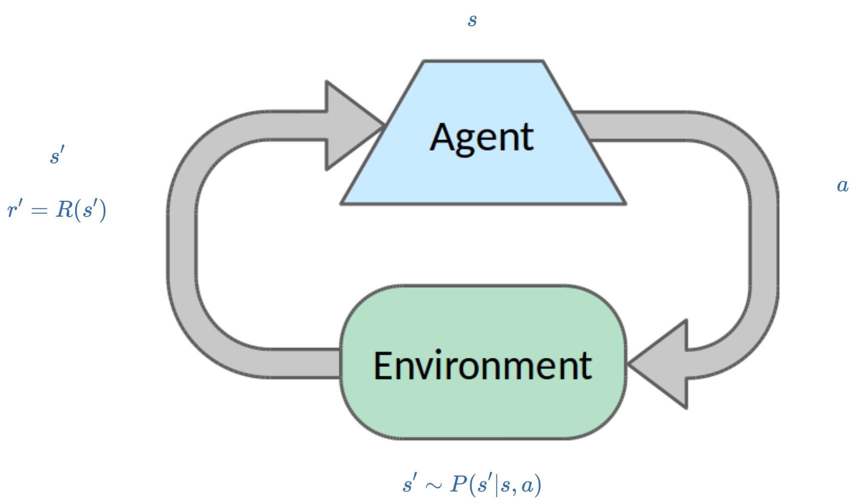


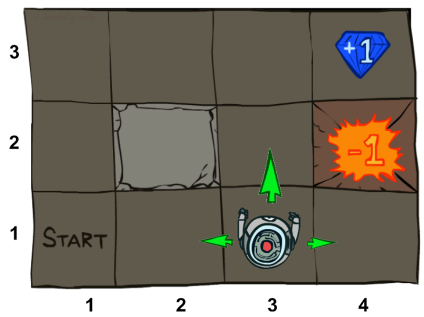
Markov decision processes

Markov decision processes

MDP หรือ Markov decision process คือพจน์ $(\mathcal{S}, \mathcal{A}, P, R)$ ดังนี้:

- $\mathcal{S}$ คือเซตของสถานะ s;
- $\mathcal{A}$ คือเซตของการกระทำ a;
- P คือแบบจำลองการเปลี่ยนผ่าน (stationary transition model) โดยที่ $P(s'|s,a)$ แทนความน่าจะเป็นที่จะไปถึงสถานะ s' เมื่อทำการกระทำ a ในสถานะ s;
- R คือฟังก์ชันรางวัลที่แมปไปยังรางวัลทันที (finite) $R(s)$ ที่ได้รับในสถานะ s.


s
Agent
s'
$r' = R(s')$
a
Environment
$s' \sim P(s'|s,a)$



ตัวอย่าง

- $\mathcal{S}$: ตำแหน่ง (i, j) บนกริด
- $\mathcal{A}$: $[\text{Up}, \text{Down}, \text{Right}, \text{Left}]$
- แบบจำลองการเปลี่ยนผ่าน: $P(s'|s, a)$
- รางวัล:

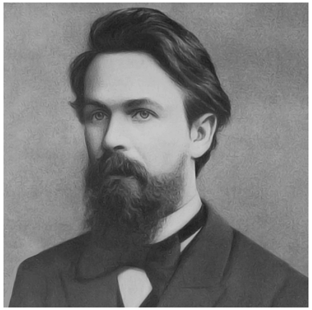
$$R(s) = \begin{cases} -0.3 & \text{สำหรับสถานะที่ไม่ใช่ปลายทาง (non-terminal states)} \\ \pm 1 & \text{สำหรับสถานะปลายทาง (terminal states)} \end{cases}$$

อะไรคือ "Markovian" ของ MDPs?

ให้สถานะปัจจุบันกำหนด อนาคตและอดีตจะเป็นอิสระต่อกัน:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0) = P(s_{t+1}|s_t, a_t)$$

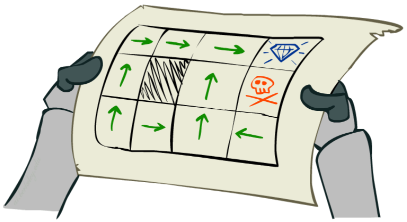
คล้ายกับปัญหาการค้นหา ที่ฟังก์ชันผู้สืบทอด (successor) ขึ้นอยู่แค่สถานะปัจจุบัน



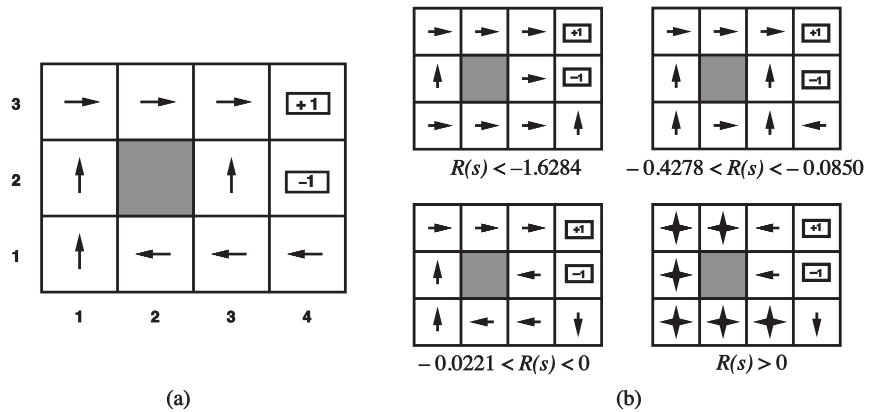
Andrey Markov

นโยบาย (Policies)

- ในปัญหาการค้นหาแบบกำหนดค่า (deterministic single-agent search) เป้าหมายของเราคือหาแผนที่เหมาะสมที่สุด หรือชุดของการกระทำตั้งแต่เริ่มจนจบ
- สำหรับ MDPs เราต้องการหา policy ที่เหมาะสมที่สุด $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$
    - นโยบาย π คือการแมปจากสถานะไปยังการกระทำ
    - นโยบายที่เหมาะสมที่สุดคืออันที่ทำให้คาดหมายอรรถประโยชน์ (expected utility) สูงสุด เช่น ผลรวมของรางวัลที่คาดหมาย
    - นโยบายแบบชัดแจ้ง (explicit policy) นิยามเอเยนต์เชิงปฏิกิริยา (reflex agent)
- Expectiminimax ไม่ได้คำนวณนโยบายทั้งชุด แต่ให้เพียงการกระทำที่สถานะเดียว



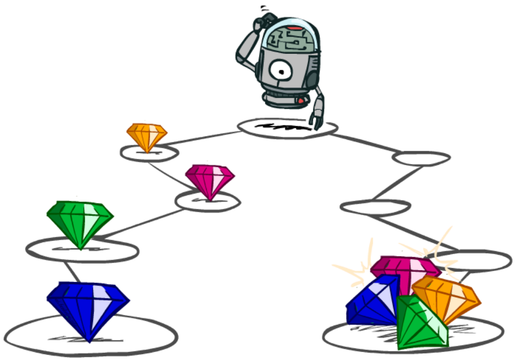
นโยบายที่เหมาะสม เมื่อ $R(s) = -0.3$ สำหรับสถานะที่ไม่ใช่ปลายทางทั้งหมด s.



(a) นโยบายที่เหมาะสม เมื่อ $R(s) = -0.04$ สำหรับสถานะที่ไม่ใช่ปลายทางทั้งหมด s. (b) นโยบายที่เหมาะสมสำหรับช่วงของ $R(s)$ สี่ค่า

ขึ้นกับ $R(s)$ ดุลยภาพระหว่างความเสี่ยงและรางวัลจะเปลี่ยน จากกล้าเสี่ยงไปจนถึงระมัดระวังมาก

อรรถประโยชน์ตามเวลา (Utilities over time)



เอเยนต์ควรมีความชอบ (preferences) เหนือลำดับสถานะหรือรางวัลอย่างไร?

- มากหรือน้อย? $[2, 3, 4]$ หรือ $[1, 2, 2]$?
- ตอนนี้หรือต่อไป? $[1, 0, 0]$ หรือ $[0, 0, 1]$?

เครดิต: CS188, UC Berkeley.

ทฤษฎีบท

ถ้าเราสมมติความชอบที่เป็น stationary เหนือลำดับของรางวัล คือ

$$[r_0, r_1, r_2, ...] \succ [r_0, r_1', r_2', ...] \Rightarrow [r_1, r_2, ...] \succ [r_1', r_2', ...],$$

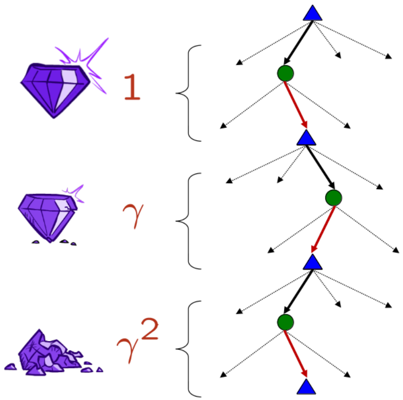
จะมีเพียงสองวิธีที่สอดคล้องกันในการกำหนดอรรถประโยชน์ให้กับลำดับ:

อรรถประโยชน์แบบบวกต่อกัน (Additive utility): $V([r_0, r_1, r_2, ...]) = r_0 + r_1 + r_2 + ...$

อรรถประโยชน์แบบมีส่วนลด (Discounted utility): ($0 < \gamma < 1$) $V([r_0, r_1, r_2, ...]) = r_0 + \gamma r_1 + \gamma^2 r_2 + ...$

การลดค่า (Discounting)

- ทุกครั้งที่เราเปลี่ยนไปสถานะถัดไป เราคูณส่วนลดหนึ่งครั้ง
- ทำไมต้องลด?
  - รางวัลที่ได้เร็วกว่าโดยมากมีอรรถประโยชน์สูงกว่ารางวัลที่ได้ช้า
  - ช่วยให้อัลกอริทึมของเราลู่เข้า (converge)



ตัวอย่าง: ส่วนลด $\gamma = 0.5$

- $V([1,2,3]) = 1 + 0.5 \times 2 + 0.25 \times 3$
- $V([1,2,3]) < V([3,2,1])$

ลำดับอนันต์ (Infinite sequences)

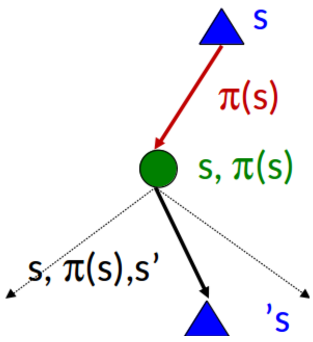
ถ้าเอเยนต์มีชีวิตตลอดไป? เราจะได้รางวัลเป็นอนันต์หรือไม่? การเปรียบเทียบลำดับรางวัลที่มีอรรถประโยชน์เป็น $+\infty$ เป็นปัญหา

วิธีแก้:

- ขอบเขตเวลา (Finite horizon): (คล้าย depth-limited search)
  - จบตอนหลังจากจำนวนก้าวคงที่ T.
  - ทำให้ได้นโยบายที่ไม่ stationary (π ขึ้นกับเวลาที่เหลือ)
- ส่วนลด (เมื่อ $0 < \gamma < 1$ และรางวัลมีขอบเขตโดย $\pm R_{\max}$):

$$V([r_0, r_1, ..., r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq \frac{R_{\max}}{1-\gamma}$$

  γ เล็กลง ทำให้ขอบเขตเวลาสั้นลง
- สถานะดูดซับ (Absorbing state): รับประกันว่า สำหรับทุกนโยบาย จะไปถึงสถานะปลายทางในที่สุด



การประเมินนโยบาย (Policy evaluation)

อรรถประโยชน์คาดหมายที่ได้จากการทำตาม π โดยเริ่มจาก s ให้โดย

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]\Bigg|_{s_0=s}$$

โดยความคาดหมายคือเหนือการแจกแจงความน่าจะเป็นของลำดับสถานะซึ่งกำหนดโดย s และ π.

เครดิต: CS188, UC Berkeley.

นโยบายที่เหมาะสม (Optimal policies)

ในบรรดานโยบายทั้งหมดที่เอเยนต์สามารถทำได้ นโยบายที่เหมาะสมที่สุดคือ π_s ที่ทำให้คาดหมายอรรถประโยชน์สูงสุด:

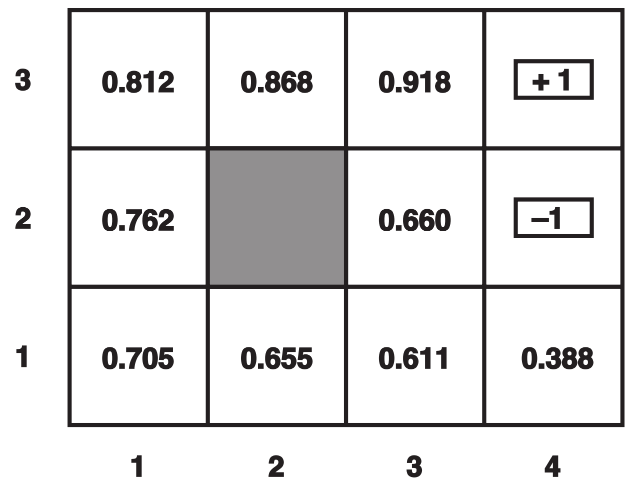
$$\pi_s = \arg\max_\pi V^\pi(s)$$

เนื่องจากมีการลดค่า อรรถประโยชน์ นโยบายที่เหมาะสมจะ "ไม่ขึ้นกับ" สถานะเริ่มต้น s (ดูภายหลัง) ดังนั้นเราจึงเขียนเพียง π^*

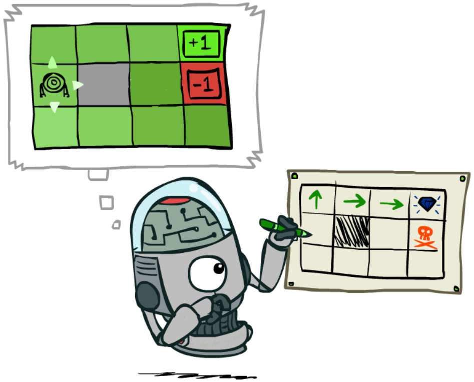
ค่าอรรถประโยชน์ของสถานะ (Values of states)

อรรถประโยชน์ หรือค่า $V(s)$ ของสถานะ นิยามเป็น $V^{\pi^*}(s)$

- คือ รางวัล (แบบลดค่า) ที่คาดหวัง หากเอเยนต์ทำตามนโยบายที่เหมาะสมโดยเริ่มจาก s
- สังเกตว่า $R(s)$ และ $V(s)$ เป็นปริมาณที่ต่างกันมาก:
  - $R(s)$ คือรางวัลระยะสั้นสำหรับการไปถึง s
  - $V(s)$ คือรางวัลรวมระยะยาวจาก s เป็นต้นไป



อรรถประโยชน์ของสถานะใน Grid World คำนวณด้วย $\gamma = 1$ และ $R(s) = -0.04$ สำหรับสถานะที่ไม่ใช่ปลายทาง



การสกัดนโยบาย (Policy extraction)

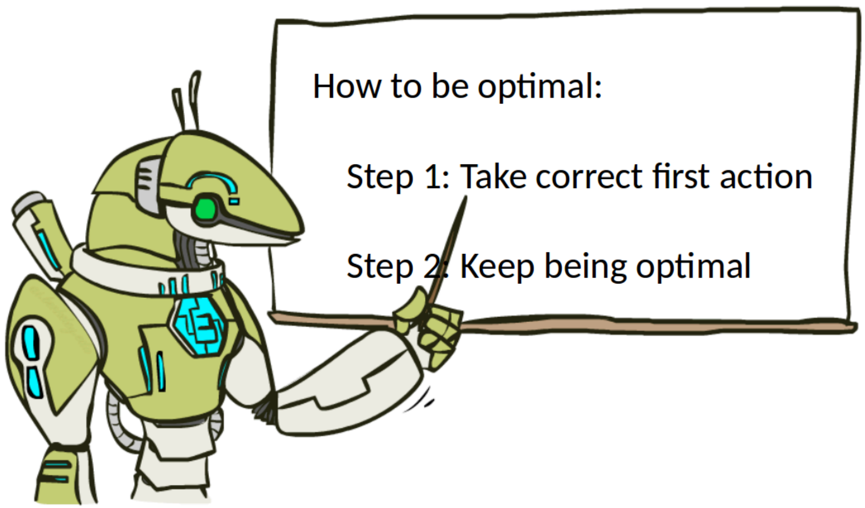
ใช้หลักการ Maximum Expected Utility การกระทำที่เหมาะสมจะทำให้คาดหมายอรรถประโยชน์ของสถานะถัดไปสูงสุด ดังนั้น

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)V(s').$$

ดังนั้น เราสามารถสกัดนโยบายที่เหมาะสมได้ หากเราประมาณอรรถประโยชน์ของสถานะได้

—
เครดิต: CS188, UC Berkeley.

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)V(s')$$



สมการของ Bellman (The Bellman equation)

อรรถประโยชน์ของสถานะเท่ากับรางวัลทันทีของสถานะนั้น บวกกับอรรถประโยชน์ที่ถูกลดค่าคาดหมายของสถานะถัดไป โดยสมมติว่าเอเยนต์เลือกการกระทำที่เหมาะสม:

$$V(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V(s').$$

- สมการเหล่านี้เรียกว่า Bellman equations เป็นระบบสมการไม่เชิงเส้น $n = |\mathcal{S}|$ สมการกับตัวแปรเท่ากัน
- อรรถประโยชน์ของสถานะ ซึ่งนิยามเป็นอรรถประโยชน์คาดหมายของลำดับสถานะถัดไป คือคำตอบของชุดสมการ Bellman

ตัวอย่าง

$$
\begin{aligned}
V(1,1) = -0.04 + \gamma \max[&0.8V(1,2) + 0.1V(2,1) + 0.1V(1,1), \\
&0.9V(1,1) + 0.1V(1,2), \\
&0.9V(1,1) + 0.1V(2,1), \\
&0.8V(2,1) + 0.1V(1,2) + 0.1V(1,1)]
\end{aligned}
$$

Value iteration

เนื่องจากตัวดำเนินการ $\max$ ทำให้สมการ Bellman เป็นไม่เชิงเส้น การแก้ระบบสมการโดยตรงจึงเป็นปัญหา

อัลกอริทึม value iteration ให้กระบวนการเวียนกลับสู่จุดตรึง (fixed-point iteration) เพื่อคำนวณค่าอรรถประโยชน์ของสถานะ $V(s)$:

- ให้ $V_i(s)$ เป็นค่าประมาณอรรถประโยชน์ของ s ที่ขั้นตอนที่ i
- Bellman update คือการอัปเดตค่าประมาณทั้งหมดพร้อมกันให้ "สอดคล้องในเชิงเฉพาะที่" กับสมการ Bellman:

$$V_{i+1}(s) := R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

- ทำซ้ำจนลู่เข้า

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
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
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) \, U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

เกณฑ์หยุดอิงกับข้อเท็จจริงว่า ถ้าการอัปเดตมีค่าน้อย แปลว่าความคลาดเคลื่อนก็เล็กด้วย คือถ้า

$$||V_{i+1} - V_i|| < \epsilon(1-\gamma)/\gamma$$

แล้วจะมี

$$||V_{i+1} - V|| < \epsilon$$

(ตัวอย่างโค้ดทีละขั้น)

การลู่เข้า (Convergence)

ให้ V_i และ V_{i+1} เป็นค่าประมาณต่อเนื่องของค่าแท้จริง V.

ทฤษฎีบท. สำหรับค่าประมาณสองตัว V_i และ V_i' ใดๆ,

$$||V_{i+1} - V'_{i+1}||_\infty \leq \gamma ||V_i - V_i'||_\infty.$$

- กล่าวคือ Bellman update เป็นการหด (contraction) ด้วยแฟกเตอร์ γ บนปริภูมิเวกเตอร์อรรถประโยชน์
- ดังนั้นค่าประมาณใดๆ จะต้องเข้าใกล้กันเรื่อยๆ และโดยเฉพาะจะเข้าใกล้ V จริง

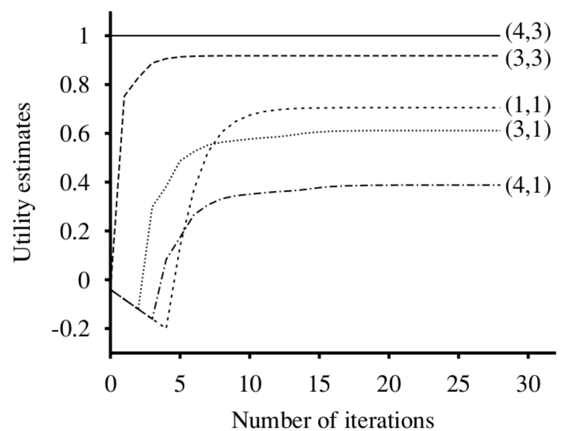
$\Rightarrow$ Value iteration ลู่เข้าสู่คำตอบเอกลักษณ์ของสมการ Bellman เสมอเมื่อ $\gamma < 1$

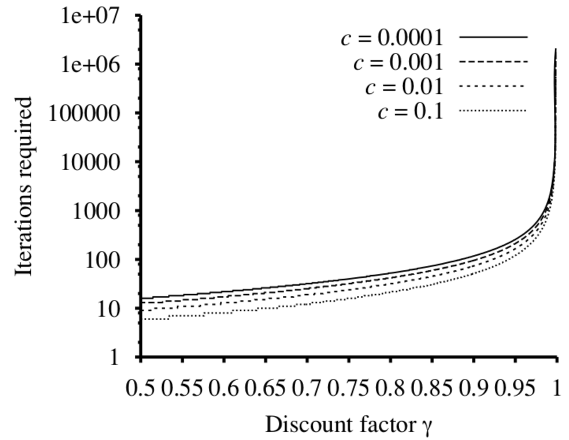
สมรรถนะ (Performance)

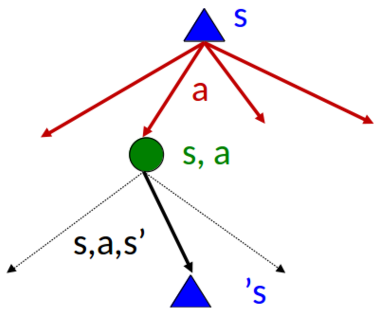
เพราะว่า $||V_{i+1} - V||_\infty \le \gamma ||V_i - V||_\infty$ ความคลาดเคลื่อนลดลงอย่างน้อยแฟกเตอร์ γ ในแต่ละรอบ

ดังนั้น value iteration ลู่เข้าเร็วแบบเอ็กซ์โปเนนเชียล:

- คลาดเคลื่อนเริ่มต้นสูงสุด $||V_0 - V||_\infty \le 2R_{\max}/(1-\gamma)$
- เพื่อให้คลาดเคลื่อนไม่เกิน ϵ หลัง N รอบ ต้องมี $\gamma^N 2R_{\max}/(1-\gamma) \le \epsilon$







ปัญหาของ value iteration

Value iteration ทำ Bellman updates ซ้ำๆ:

$$V_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

- ปัญหา 1: ช้า – $O(|\mathcal{S}|^2|\mathcal{A}|)$ ต่อรอบ
- ปัญหา 2: ค่า $\max$ ที่แต่ละสถานะ เปลี่ยนไม่บ่อย
- ปัญหา 3: นโยบาย π_i ที่สกัดจาก V_i อาจเหมาะสมแล้ว แม้ V_i ยังไม่แม่นยำ!

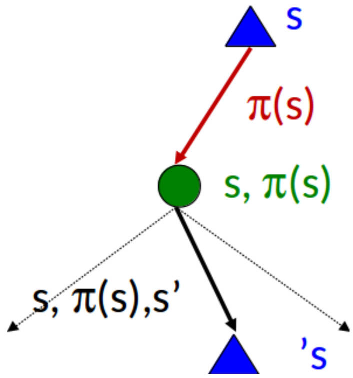
Policy iteration

อัลกอริทึม policy iteration คำนวณนโยบายโดยตรง (แทนที่จะคำนวณค่าอรรถประโยชน์ของสถานะ) โดยสลับสองขั้นตอนนี้:

- การประเมินนโยบาย: เมื่อกำหนด π_i คำนวณ $V_i = V^{\pi_i}$ คืออรรถประโยชน์ของแต่ละสถานะหากทำตาม π_i
- การปรับปรุงนโยบาย: คำนวณนโยบายใหม่ π_{i+1} ด้วยการมองไปข้างหน้า 1 ก้าว บนพื้นฐาน V_i:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} P(s'|s,a) V_i(s')$$

อัลกอริทึมนี้ยังคงให้คำตอบเหมาะสม และอาจลู่เข้าเร็วกว่าอย่างมากในบางกรณี



การประเมินนโยบาย (Policy evaluation)

ที่รอบที่ i เราได้เวอร์ชันง่ายของสมการ Bellman ที่เชื่อมค่าอรรถประโยชน์ของ s กับของเพื่อนบ้าน:

$$V_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s))V_i(s')$$

สมการเหล่านี้ตอนนี้เป็น เชิงเส้น เพราะไม่มีตัวดำเนินการ $\max$ แล้ว

- สำหรับ n สถานะ เรามี n สมการ n ตัวไม่ทราบค่า
- แก้ได้พอดีใน $O(n^3)$ ด้วยพีชคณิตเชิงเส้นมาตรฐาน (สังเกตว่าเราแทนที่ a ด้วย $\pi_i(s)$)

บางครั้ง $O(n^3)$ ก็แพงเกินไป โชคดีที่ไม่จำเป็นต้องประเมินนโยบายอย่างแม่นยำเสมอไป วิธีประมาณก็เพียงพอ

วิธีหนึ่งคือรันสมการ Bellman แบบง่าย k รอบ:

$$V_{i+1}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) V_i(s')$$

อัลกอริทึมผสมนี้เรียกว่า modified policy iteration

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

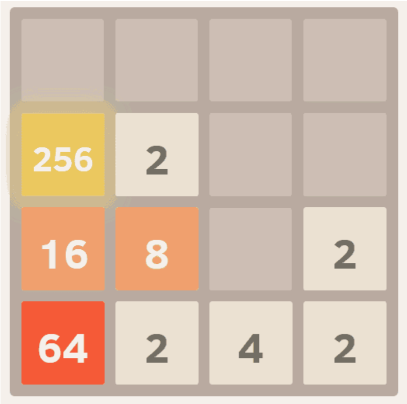
  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

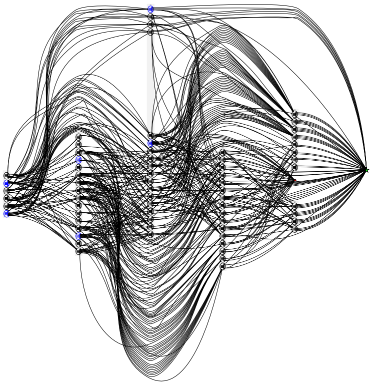
(ตัวอย่างโค้ดทีละขั้น)

ตัวอย่างสรุป: 2048

เกม 2048 เป็น Markov decision process!

- $\mathcal{S}$: คอนฟิกูเรชันทั้งหมดของกระดาน (มหาศาล!)
- $\mathcal{A}$: ปัดซ้าย ขวา ขึ้น ลง
- $P(s'|s,a)$: เข้ารหัสพลวัตของเกม
  - รวมช่องตัวเลขที่เท่ากัน
  - วางตัวเลขแบบสุ่มบนกระดาน
- $R(s) = 1$ ถ้า s เป็นสถานะชนะ มิฉะนั้น 0





แบบจำลองการเปลี่ยนผ่านสำหรับกระดาน 2×2 และสถานะชนะที่ 8

เครดิต: jdlm.info, The Mathematics of 2048.

การเล่นที่เหมาะสมสำหรับกริด 3×3 และสถานะชนะที่ 1024.

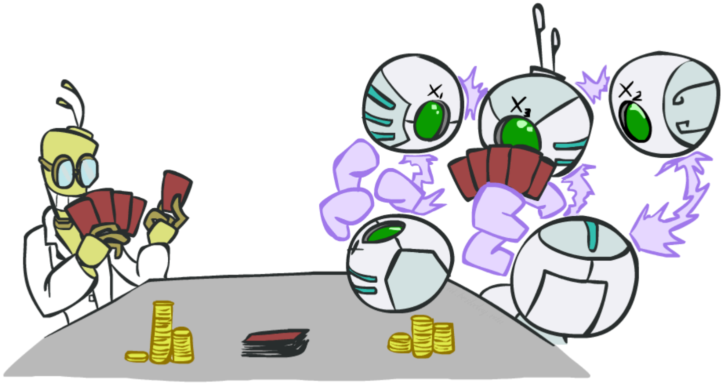
ดู jdlm.info: The Mathematics of 2048.

Partially observable Markov decision processes

POMDPs

ถ้าสภาพแวดล้อม "สังเกตได้บางส่วน" (partially observable) ล่ะ?

- เอเยนต์ไม่รู้ว่าตนเองอยู่ในสถานะ s ไหน
  - เพราะฉะนั้น ประเมิน $R(s)$ ของสถานะที่ไม่รู้ไม่ได้
  - และการพูดถึงนโยบาย $\pi(s)$ ก็ไม่มีความหมาย
- แทนที่จะเป็นเช่นนั้น เอเยนต์รับรู้สัญญาณ e ผ่านแบบจำลองเซ็นเซอร์ $P(e|s)$ เพื่อใช้ให้เหตุผลเกี่ยวกับสถานะที่ไม่รู้ s

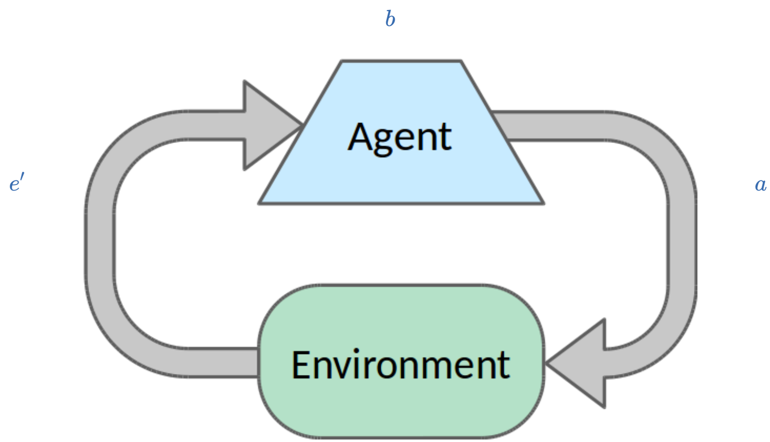


—
เครดิต: CS188, UC Berkeley.

เราจะสมมติว่าเอเยนต์คงไว้ซึ่งสถานะความเชื่อ (belief state) b.

- b แทนการแจกแจงความน่าจะเป็น $\mathbf{P}(S)$ ของความเชื่อของเอเยนต์ ณ ตอนนี้เหนือสถานะของมัน
- $b(s)$ แทนความน่าจะเป็น $P(S = s)$ ภายใต้ belief ปัจจุบัน
- belief state b ถูกอัปเดตเมื่อมีหลักฐาน e ใหม่เข้ามา

นี่คือการกรอง (Filtering)!



$$s' \sim P(s'|s, a)$$

$$e' \sim P(e'|s')$$

Belief MDP

ทฤษฎีบท (Åström, 1965). การกระทำที่เหมาะสมขึ้นกับ belief state ปัจจุบันของเอเยนต์เท่านั้น

- นโยบายที่เหมาะสมสามารถบรรยายได้ด้วยแมป $\pi^*(b)$ จาก belief ไปยังการกระทำ
- ไม่ขึ้นกับสถานะจริงที่เอเยนต์อยู่

กล่าวคือ POMDPs สามารถลดรูปเป็น MDP บนปริภูมิ belief-state ได้ โดยกำหนดแบบจำลองการเปลี่ยนผ่าน $P(b'|b,a)$ และฟังก์ชันรางวัล ρ บน belief states

ถ้า b เป็น belief ก่อนหน้า และเอเยนต์ทำการกระทำ a และรับรู้ e แล้ว belief ใหม่เหนือ S' คือ

$$b' = \alpha \mathbf{P}(e|S') \sum_s \mathbf{P}(S'|s,a) b(s) = \alpha\, \text{forward}(b,a,e).$$

ดังนั้น

$$\begin{aligned}
P(b'|b,a) &= \sum_e P(b',e|b,a) \\
&= \sum_e P(b'|b,a,e) P(e|b,a) \\
&= \sum_e P(b'|b,a,e) \sum_{s'} P(e|b,a,s') P(s'|b,a) \\
&= \sum_e P(b'|b,a,e) \sum_{s'} P(e|s') \sum_s P(s'|s,a) b(s)
\end{aligned}$$

โดยที่ $P(b'|b,a,e) = 1$ ถ้า $b' = \text{forward}(b,a,e)$ และเป็น 0 มิฉะนั้น

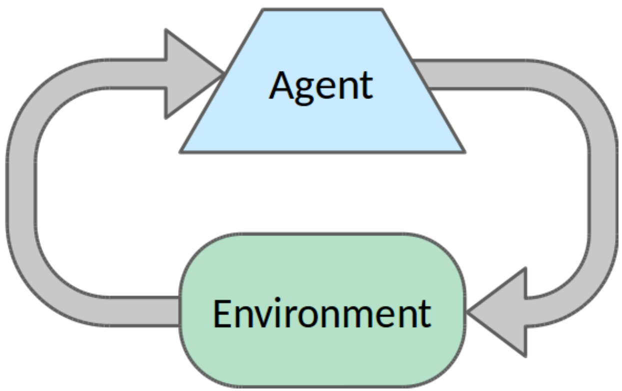
เรายังกำหนดฟังก์ชันรางวัลสำหรับ belief state ได้เป็นรางวัลคาดหมายเหนือสถานะจริงที่เอเยนต์อาจอยู่:

$$\rho(b) = \sum_s b(s)R(s)$$

b

b'

$\rho(b')$



a

$b' \sim P(b'|b, a)$

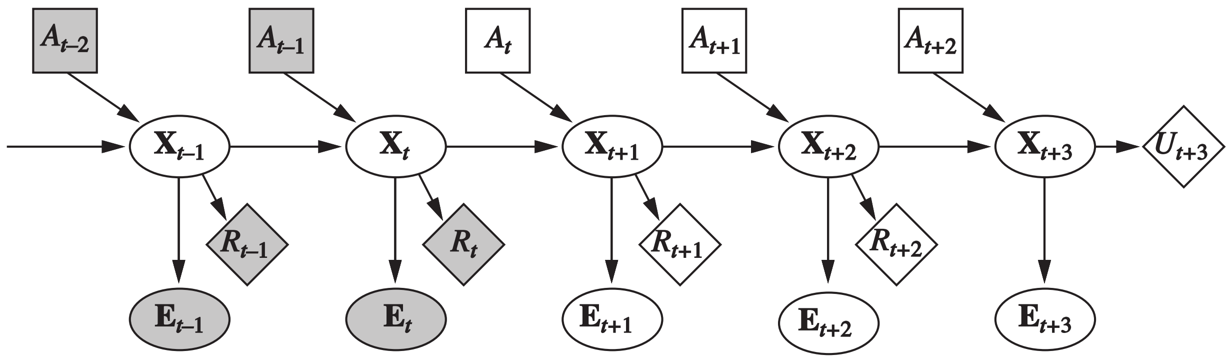
แม้ว่าเราจะลด POMDPs เป็น MDPs ได้ แต่ Belief MDP ที่ได้มีปริภูมิสถานะเป็น ต่อเนื่อง (และโดยมากมิติมาก)

- อัลกอริทึมก่อนหน้าทั้งหมดใช้ตรงๆ ไม่ได้
- ที่จริง การแก้ POMDPs เป็นปัญหายากที่ยังไม่มีอัลกอริทึม exact ที่มีประสิทธิภาพ
- แต่ธรรมชาติก็เป็น POMDP

เอเยนต์แบบออนไลน์ (Online agents)

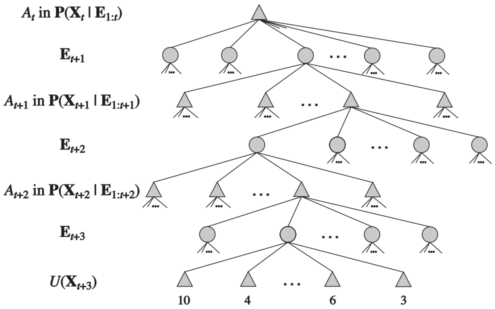
แม้จะยากที่จะหา π^* โดยตรง เอเยนต์เชิงทฤษฎีการตัดสินใจ (decision-theoretic agent) สำหรับ POMDPs สร้างได้ดังนี้:

- แบบจำลองการเปลี่ยนผ่านและเซ็นเซอร์แทนด้วย dynamic Bayesian network
- ขยายเครือข่ายด้วยโหนดการตัดสินใจ (A) และอรรถประโยชน์/รางวัล (R, U) ให้เป็น dynamic decision network
- ใช้อัลกอริทึม filtering รวมหลักฐานและการกระทำใหม่เพื่ออัปเดต belief state
- ตัดสินใจโดยจำลองไปข้างหน้าสำหรับลำดับการกระทำที่เป็นไปได้และเลือกอันที่ดีที่สุด (โดยประมาณ) คล้าย Expectiminimax แบบตัดความลึก



ที่เวลา t เอเยนต์ต้องตัดสินใจว่าจะทำอะไร

- โหนดทึบหมายถึงตัวแปรที่รู้ค่าแล้ว
- คลี่เครือข่ายออกสำหรับขอบเขตเวลาจำกัด
- รวมโหนดรางวัลของ $\mathbf{X}_{t+1}$ และ $\mathbf{X}_{t+2}$ แต่ใช้ (ค่าประมาณ) อรรถประโยชน์ของ $\mathbf{X}_{t+3}$



ส่วนหนึ่งของคำตอบแบบมองไปข้างหน้าของเครือข่ายการตัดสินใจก่อนหน้า:

- โหนดสามเหลี่ยมแต่ละอันคือ belief state ที่เอเยนต์ตัดสินใจ
  - belief state ที่แต่ละโหนดคำนวณได้โดยใช้การกรอง (filtering) กับลำดับของการรับรู้และการกระทำที่นำไปถึงโหนดนั้น
- โหนดวงกลมคือการสุ่มเลือกโดยสภาพแวดล้อม

การตัดสินใจสกัดได้จากต้นไม้ค้นหาโดยแบ็คอัปอรรถประโยชน์ (ประมาณ) จากใบ ขณะที่โหนดโอกาสใช้ค่าเฉลี่ย และโหนดตัดสินใจใช้ค่าสูงสุด

สรุป (Summary)

- ปัญหาการตัดสินใจตามลำดับในสภาพแวดล้อมที่ไม่แน่นอน (MDPs) นิยามด้วยแบบจำลองการเปลี่ยนผ่านและฟังก์ชันรางวัล
- อรรถประโยชน์ของลำดับสถานะคือผลรวมของรางวัลทั้งหมดในลำดับ อาจมีการลดค่าตามเวลา
    - คำตอบของ MDP คือ นโยบายที่กำหนดการตัดสินใจสำหรับทุกสถานะที่เอเยนต์อาจเจอ
    - นโยบายที่เหมาะสมทำให้อรรถประโยชน์ของลำดับสถานะที่พบเมื่อทำตามนโยบายสูงสุด
- Value iteration และ Policy iteration ใช้แก้ MDP ได้ทั้งคู่
- POMDPs ยากกว่า MDPs มาก อย่างไรก็ดี เราสร้างเอเยนต์เชิงทฤษฎีการตัดสินใจสำหรับสภาพแวดล้อมเหล่านี้ได้

จบการนำเสนอ