インテリジェントシステム

#8 強化学習
Reinforcement Learning

信州大学工学部電子情報システム工学科
丸山稔

強化学習：MDPと同じ問題設定⇒報酬期待値を最大化したい

MDP：
状態集合 $s \in \mathcal{S}$，行動集合 $a \in \mathcal{A}$, 状態遷移モデル $P(s'|s,a)$ ，報酬関数 $R(s,a,s')$

強化学習：
agentはMDPで定義された環境内にいる … 但し、**状態遷移モデル、報酬関数は未知**
⇒ 行動しながら学習していく必要がある　⇒ 適切な方策 π (s) を得たい

強化学習アルゴリズムの分類

・モデルに基づく強化学習（model-based RL）

・モデル無し強化学習（model-free RL）
　＊行動価値学習（action-utility learning）
　＊方策探索（policy search）

確率変数A

モデルP(A)が既知ならば

$$\mathbb{E}[A] = \sum_a P(a) \cdot a$$

モデルP(A)が未知…この分布に従って得られたサンプルを集める $\{a_1, a_2, \cdots, a_N\}$

モデルに基づく手法

$$\hat{P}(A = a) = \frac{N_a}{N}$$

$$\mathbb{E}[A] \approx \sum_a \hat{P}(a) \cdot a$$

モデル無しの手法

$$\mathbb{E}[A] \approx \frac{1}{N} \sum_i a_i$$

モデルに基づく強化学習（model-based reinforcement learning, MBRL）

agentは環境の状態遷移モデルを用いる

・初期状態ではモデルは未知の場合⇒**agentの行動の結果からモデルを学習していく**
・初期状態から既知の場合もある←ゲームのルールなど（囲碁、将棋etc）
良い手を打つ方策は未知だがルールは既知

多くの場合、価値関数U(s)を学習する

U(s)：状態sから開始して得られる報酬の和（割引率あり）

モデル無し強化学習（model-free reinforcement learning）

agentは状態遷移モデルを知らない＆学習もしない→どう行動したらよいかを決定するための
より直接的な表現を学習

＊行動価値学習（action-utility learning）
行動価値関数Q(s,a)を学習→ ex. Q-learning … agentはQ関数（Q(s,a)）を学習する
Q(s,a)：状態sで行動aをとった以降に得られる報酬の和
Q-関数が与えられれば、agentは状態sにおいてQ-値を最大にするaを選択すればよい

＊方策探索（policy search）
π(s)が対象→agentは　状態→行動を与える方策π(s)を学習する

受動的強化学習（Passive Reinforcement Learning）

まずは簡単な場合から…
環境は完全に観測可能で、状態空間や行動集合も小さな場合を考え、
さらに、エージェントは固定された方策 π (s)を持っているものとする

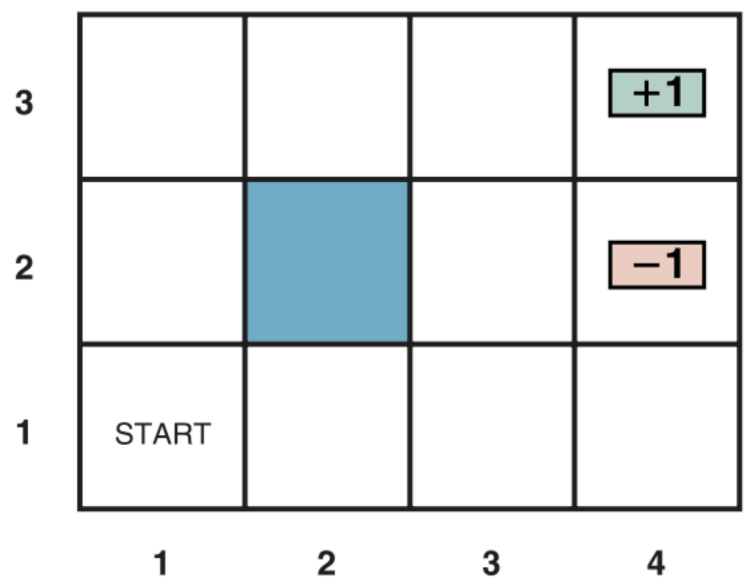
agent：価値関数 U^{π} を学習したい→ 受動的強化学習

方策評価（policy evaluation）と似ているが遷移確率 $P(s'|s,a)$ は未知
報酬関数（reward function）$R(s,a,s')$ も未知

$$U_i(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma U_i(s')]$$ … どうやって計算する？

受動的強化学習（Passive Reinforcement Learning）

agent：方策πに従って行動→試行（trials）の集合を得る
例えば…4x3 worldの場合
(1,1)からスタートして、πに従って行動
⇒ 終端状態(4,2)または(4,3)に到達するまでの状態と報酬の系列集合を得る



この情報に基づいて $U^\pi(s)$ を学習したい（sは非終端状態）

$$U^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})]$$

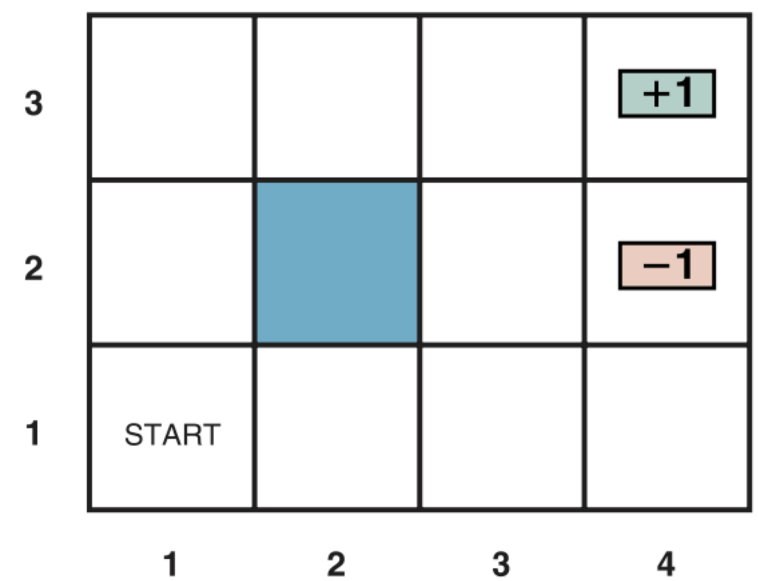
- 直接価値推定（direct utility estimation）
- 適応的動的計画法（adaptive dynamic programming）

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

直接的価値推定（direct utility estimation）

4x3 grid world



終端状態の報酬以外はr=-0.04, γ=1とする

この環境でエージェントが行動⇒ 状態と報酬の系列⇒各状態の価値のサンプルが得られる

例：

(1,1)→(1,2)→(1,3)→(1,2)→(1,3)→(2,3)→(3,3)→**(4,3)**：

・(1,1)から開始して得られた報酬総和は？

(-0.04) + (-0.04) + (-0.04) + (-0.04) + (-0.04) + (-0.04) + 1 = 0.76

・(1,2)から開始して得られた報酬総和は（２つのサンプルが得られる）

(-0.04) + (-0.04) + (-0.04) + (-0.04) + (-0.04) + 1 = 0.8

(-0.04) + (-0.04) + (-0.04) + 1 = 0.88

平均値により価値関数の推定が可能

受動的強化学習（Passive Reinforcement Learning）

直接的価値推定（direct utility estimation）

各状態の価値=その状態から開始して得られる報酬総和の期待値
各trial … 出現/訪れた状態に関するサンプルを与える

4x3 worldの例：３つのtrialsが得られたとする（終端状態の報酬以外はr=-0.04, γ=1とする）

①(1,1)→(1,2)→(1,3)→(1,2)→(1,3)→(2,3)→(3,3)→**(4,3)**：

②(1,1)→(1,2)→(1,3)→(2,3)→(3,3)→(3,2)→(3,3)→**(4,3)**：

③(1,1)→(1,2)→(1,3)→(2,3)→(3,3)→(3,2)→**(4,2)**：

(1,1)の出現回数３：それ以降の報酬総和は0.76, 0.76, -1.2
(1,2)の出現回数４：それぞれの報酬総和は0.8, 0.88, 0.8, -1.16

➡ （状態, その後の報酬）のペアが得られる⇒各状態について平均値を更新していく

問題点：

・価値はBellman方程式を満たす必要があるが、この条件を満たす保証はない
・収束までに時間がかかる

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

適応的動的計画法（adaptive dynamic programming, ADP）

ADP ： 価値関数に関する制約（Bellman方程式…方策は固定）を用いる

$$U_i(s) = \sum_{s'} P(s'|s, \pi(a))[R(s, \pi(s), s') + \gamma U_i(s')]$$

線形方程式を解く or 方策反復（policy iteration）で解く

↓

状態遷移モデルや報酬関数が未知← サンプルから学習

$P(s'|s,a)$ ← 状態-行動(s,a)の後、得られた状態s'…得られた回数を記憶しておく

表 $N_{s'|s,a}$

$$P(s'|s,a) \leftarrow \frac{N_{s'|s,a}[s,a][s']}{\sum_t N_{s'|s,a}[s,a][t]}$$

↓ 簡略版Bellman-updateを繰り返す

$$U(s) \leftarrow \sum_{s'} P(s'|s, \pi(a))[R(s, \pi(s), s') + \gamma U(s')]$$

受動的強化学習（Passive Reinforcement Learning）

適応的動的計画法（adaptive dynamic programming）

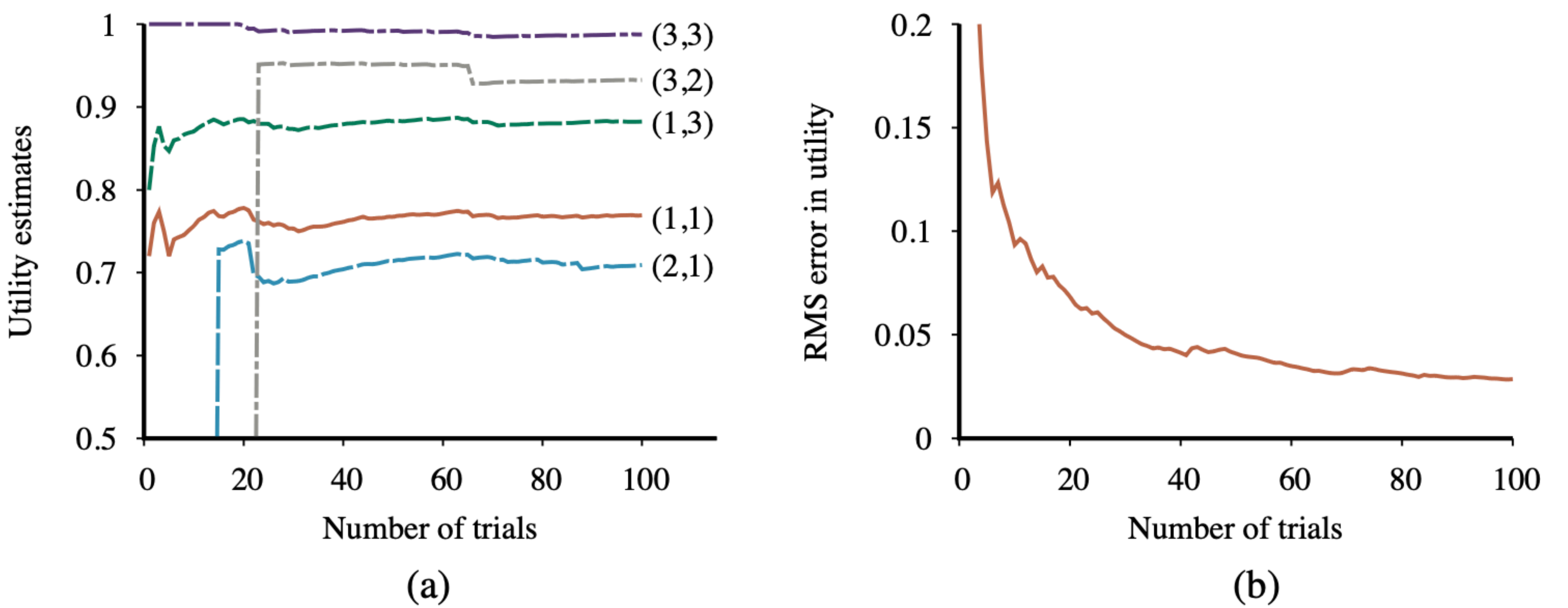
function PASSIVE-ADP-LEARNER($percept$) **returns** an action
  inputs: $percept$, a percept indicating the current state s' and reward signal r
  persistent: π, a fixed policy
    mdp, an MDP with model P, rewards R, actions A, discount γ
    U, a table of utilities for states, initially empty
    $N_{s'|s,a}$, a table of outcome count vectors indexed by state and action, initially zero
    s, a, the previous state and action, initially null

  if s' is new **then** $U[s'] \leftarrow 0$
  if s is not null **then**
    increment $N_{s'|s,a}[s, a][s']$
    $R[s, a, s'] \leftarrow r$
    add a to $A[s]$
    $\mathbf{P}(\cdot \mid s, a) \leftarrow$ NORMALIZE($N_{s'|s,a}[s, a]$)
    $U \leftarrow$ POLICYEVALUATION(π, U, mdp)
    $s, a \leftarrow s', \pi[s']$
    return a

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

適応的動的計画法（adaptive dynamic programming）



S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

時間的差分学習（Temporal Difference Learning、TD学習）

次の状態遷移列を観測
②(1,1)→(1,2)→(1,3)→(2,3)→(3,3)→(3,2)→(3,3)→**(4,3)**：

②の実行で状態遷移(1,3) → (2,3) （報酬 r = -0.04）が発生

この状態遷移が常に起こるとすると…即ち $P(s'|s,\pi(s))=1$ とすると

$U^\pi(1,3)=0.88 \qquad U^\pi(2,3)=0.96$

$U^\pi(1,3)=-0.04+U^\pi(2,3)=0.92$

現在の推定値（0.88）を少し増加すべき

$$\begin{cases} sample = R(s,\pi(s),s') + \gamma U^\pi(s') \\ \text{現在の推定値} = U^\pi(s) \end{cases}$$

平均化（running average） $U^\pi(s) \leftarrow (1-\alpha) * U^\pi(s) + \alpha * sample$

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha[R(s,\pi(s),s') + \gamma U^\pi(s') - U^\pi(s)]$$

動的強化学習（Active Reinforcement Learning）

Temporal Difference Learning

移動平均（running average）

平均値の計算

$$x_1, x_2, \cdots, x_n \quad \rightarrow \text{平均値} \quad \mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(\sum_{i=1}^{n-1} x_i + x_n) = \frac{1}{n}((n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} x_i + x_n)$$

$$= \frac{n-1}{n}\mu_{n-1} + \frac{1}{n}x_n = (1-\frac{1}{n})\mu_{n-1} + \frac{1}{n}x_n$$

定数αを用いると　$\mu_n = (1-\alpha)\mu_{n-1} + \alpha x_n$

データが到着する度に　$\mu_n = (1-\alpha)\mu_{n-1} + \alpha x_n$　で　μ_n を更新

$$\begin{aligned}\mu_n &= (1-\alpha)\mu_{n-1} + \alpha x_n \\ &= (1-\alpha)\{(1-\alpha)\mu_{n-2} + \alpha x_{n-1}\} + \alpha x_n = (1-\alpha)^2\mu_{n-2} + \alpha(1-\alpha)x_{n-1} + \alpha x_n \\ &= (1-\alpha)^3\mu_{n-3} + \alpha(1-\alpha)^2 x_{n-2} + \alpha(1-\alpha)x_{n-1} + \alpha x_n \\ &\cdots\end{aligned}$$

古いデータに小さい重みを与えて（重み付き）平均値を計算していることになる

受動的強化学習（Passive Reinforcement Learning）

Temporal Difference Learning

状態遷移 $s \to s'$ が行動 $\pi(s)$ によって発生したとき $U^\pi(s)$ を以下のように更新する

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha[R(s, \pi(s), s') + \gamma U^\pi(s') - U^\pi(s)]$$

➡ **TD（temporal difference）方程式**

連続した時刻間での価値の違いに基づく

TD項： $R(s, \pi(s), s') + \gamma U^\pi(s') - U^\pi(s)$ ➡ これが減少するように更新

理想的には平衡状態→価値予測値が（局所的に）正しければ成立

$$\begin{cases} R(s, \pi(s), s') + \gamma U^\pi(s') - U^\pi(s) > 0 & \Rightarrow \text{平衡するためには } U^\pi(s) \text{ 増大させる} \\ R(s, \pi(s), s') + \gamma U^\pi(s') - U^\pi(s) < 0 & \Rightarrow \text{平衡するためには } U^\pi(s) \text{ 減少させる} \end{cases}$$

➡ 平衡状態に近づける…但しここでは実際に観測された２状態s,s’しか考慮に入れていない

本来の平衡条件には全てのs’が関与

TD項の計算に遷移モデルは必要ない

→ 環境自身が隣接する状態との関係を観測例の形で提示する

受動的強化学習（Passive Reinforcement Learning）

Temporal Difference Learning

function PASSIVE-TD-LEARNER(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r
  persistent: π, a fixed policy
    s, the previous state, initially null
    U, a table of utilities for states, initially empty
    N_s, a table of frequencies for states, initially zero

  if s' is new **then** $U[s'] \leftarrow 0$
  if s is not null **then**
    increment $N_s[s]$
    $U[s] \leftarrow U[s] + \alpha(N_s[s]) \times (r + \gamma\, U[s'] - U[s])$
  $s \leftarrow s'$
  return $\pi[s']$

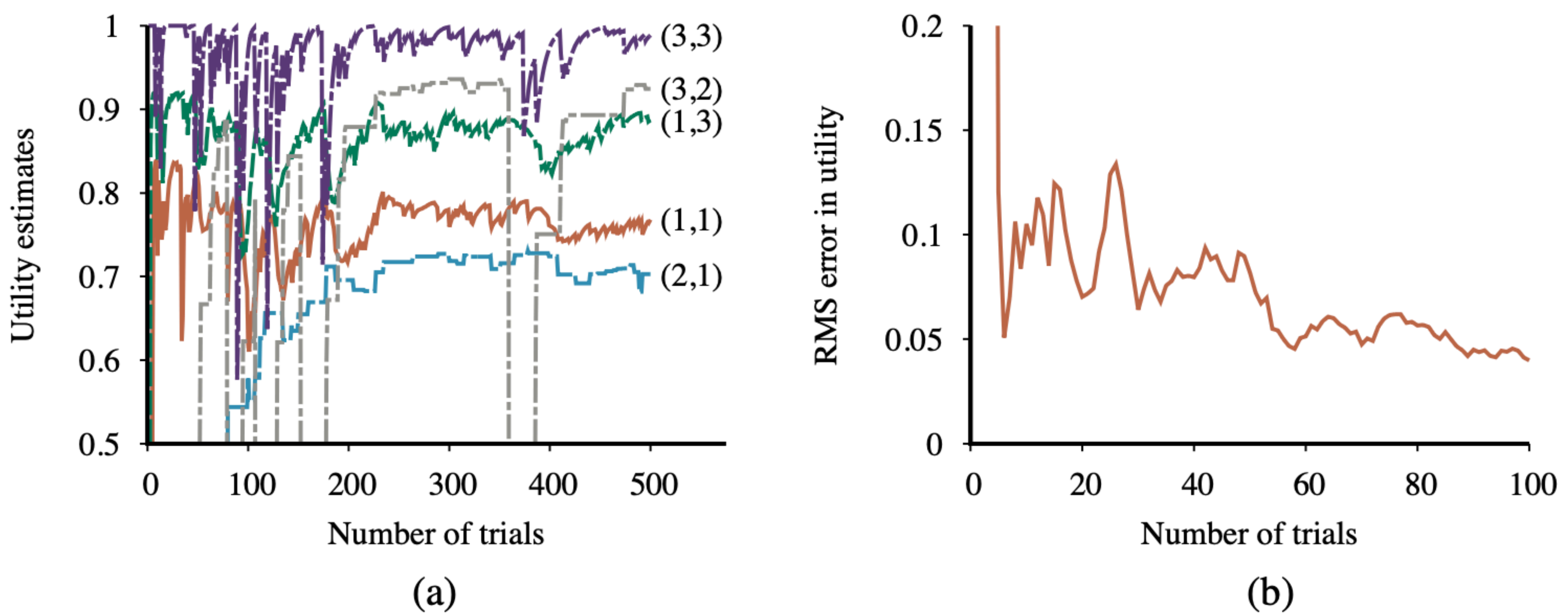
α：学習率→状態sを訪れた回数が増大すると値が減少するように設定

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

Temporal Difference Learning

$$\alpha(n) = \frac{60}{59 + n}$$



ADPと比較すると…

TDはADPほど収束が高速でもなく変動も大きいが、観測毎の計算コストは少ない

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020

http://aima.cs.berkeley.edu/figures.pdf

受動的強化学習（Passive Reinforcement Learning）

Temporal Difference Learning

TD法のまとめ

$$sample = R(s, \pi(s), s') + \gamma U^{\pi}(s')$$

$$\begin{aligned} U^{\pi}(s) \leftarrow & (1-\alpha)U^{\pi}(s) + \alpha \cdot sample \\ &= U^{\pi}(s) + \alpha\underbrace{(sample - U^{\pi}(s))}_{\text{TD-エラー}} \end{aligned}$$

経験データ⇒sample⇒ $U^{\pi}(s)$ の値を $U^{\pi}(s')$ と整合するように修正していく

モデルなし学習（方策評価、移動平均）によるBellman updateとみなせる
… 但し…
　得られた価値関数の使用（行動選択や方策改善）のためには遷移確率モデルが必要

例：１ステップ先の予測

$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma U(s')]$$

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

動的強化学習（Active Reinforcement Learning）

Temporal Difference Q-Learning

Q-関数（行動価値関数）の定義を思い出してみると…

$Q^*(s,a)$：状態sで行動aを行い、その後最適な行動をとったときの累積報酬期待値

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

⇒ Q関数が得られれば（遷移確率モデルがなくても）最適行動（方策）が得られる

Q関数の場合のBellman方程式

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')]$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Q関数の場合の近似Bellman update

$$\begin{aligned} Q(s,a) \leftarrow & (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \\ &= Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \end{aligned}$$

➡ 遷移確率モデル無しに、学習したQ(s,a)から方策が得られる

…但しQ(s,a)はU(s)より $|\mathcal{A}|$ 倍大きい！

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

動的強化学習（Active Reinforcement Learning）

Temporal Difference Q-Learning

Q-学習（Q-learning）まとめ

サンプル：(s, a, s', r) を得る

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a') = r + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \cdot sample$$

α は時間経過とともに適切に減少させていく必要あり

$$\sum_t \alpha(t) = \infty, \quad \sum_t \alpha(t)^2 < \infty$$

e.x. $\alpha(t) = \frac{1}{t}, \quad \alpha(t) = \frac{K}{K + t}$ など…

S.Russell & P.Norvig, Artificial Intelligence 4th Ed., Pearson, 2020
http://aima.cs.berkeley.edu/figures.pdf

探索と活用（Exploration vs Exploitation）

ε 貪欲探索⇒活用に特化した貪欲方策にランダム性を導入したもの

探索が必要になるのは？... 各(s,a)の評価が正しくないとき
推定誤りの種類：過大評価＆過小評価

状態sにおける行動aが過大評価⇒aは最適でないのに選択される⇒(s,a)の事例が増大
⇒評価の修正につながる
状態sにおける最適行動a*を過小評価⇒a*以外を選択⇒(s,a*)に関する経験が増加しない

↓

原則：不確実なときは楽観的に行動する i.e. 推定が不確実⇒優先的に行動

探索関数（exploration function）
u: 推定値、n: （状態、行動）対の訪問回数（出現回数） → 例： $f(u,n) = u + \frac{K}{\sqrt{n}}$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N_{s',a'})]$$

経験した回数が増加すると共に（推定の確実性向上）通常のupdateになる
回数が少ないときは過大評価