

情報意味論(14)

強化学習

櫻井彰人
慶應義塾大学理工学部

まずMDPIについて

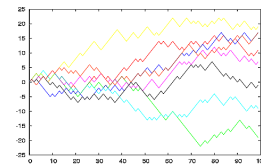
- 確率過程
- マルコフ性
- マルコフ鎖
- マルコフ決定過程
- 強化学習
- 強化学習の技法

確率過程

- 簡単に言えば: ランダムな時系列
- しばしば、インデックスのついた確率変数の集まりと考える
- 基本: 状態とその状態にいる確率(時刻でインデックスされている)の集合
- 離散確率過程を考える

確率過程の例

- 古典: ランダムウォーク
 - ある時刻 t_0 に状態 X_0 で開始する
 - 時刻 t_i にて、ステップ Z_i だけ動く。ただし $P(Z_i = -1) = p$ and $P(Z_i = 1) = 1 - p$
 - すなわち、時刻 t_i においては状態 $X_i = X_0 + Z_1 + \dots + Z_i$



http://en.wikipedia.org/wiki/Image:Random_Walk_example.png

マルコフ性

- "無記憶性" である
- 確率過程がマルコフ性を持つとは、状態が X_{n+1} となる(条件付)確率が現在の状態 X_n のみに依存して、過去の状態(系列)に依存しない場合である
 - $\text{Prob} \{ X_n = k \mid X_{n-1} = a, X_{n-2} = b, \dots \}$
 $= \text{Prob} \{ X_n = k \mid X_{n-1} = a \}$

次数が1の場合

マルコフ性の例

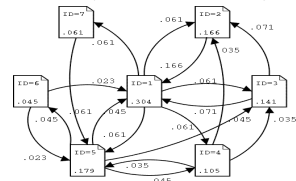
- 囲碁、将棋、チェス、オセロ、...:
 - 現在状態: 盤面(持ち駒があるならそれを含め)の現在の状態
 - 遷移する次の状態に関するすべての情報を持っている
 - 盤面自体であることに注意
 - 従って、マルコフ性があると考えてよい

人間が打つ場合には、同じ一局の中であっても、同じ定石・定跡を使わないことがある。

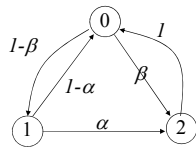
マルコフ鎖

- マルコフ性を持った離散時間確率過程
- 確率過程でなくてもマルコフ鎖とみなすことにより、有用な結果が得られることがある

$$PR(A) = \frac{1-d}{N} + d \left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots \right)$$



<http://en.wikipedia.org/wiki/PageRank>



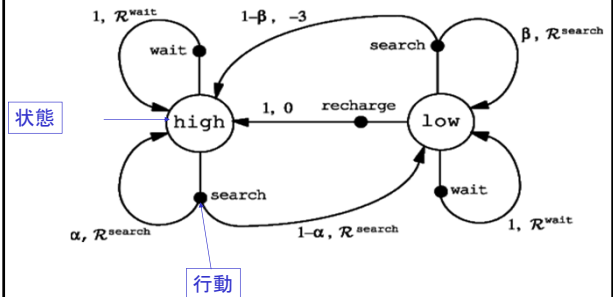
マルコフ決定過程 (MDP)

- 離散時間の決定過程
- マルコフ鎖の拡張
- 拡張部分:
 - 行動(action)の追加 (選択)
 - 報酬(rewards)の追加 (動機付け)
- (各状態に対して)行動が決まっていれば、MDPはマルコフ鎖と同じ

MDPs の記述

- 対 $(S, A, P(\cdot, \cdot), R(\cdot))$
 - S : 状態空間 (state space)
 - A : 行動空間 (action space)
 - $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$
 - $R(s) =$ 状態 s における即時報酬
- 目標は、累積報酬を最大化
- 有限 MDPs : 状態と行動が有限

MDPの遷移グラフ例



MDP の解 = 方策 π

- (過去に履歴とは独立して)現在の状態に基づきとるべき行動を与える
- (離散MDPの場合)状態を添え字とする2配列で表現
 - V : 価値関数 (value function), 現在の方策に従って行動した場合に得られる期待 (割引き) 累積報酬
 - π : 方策 (policy), つぎにとるべき行動

$$\pi(s) := \arg \max_a \sum_{s'} P_a(s, s') V(s')$$

アクションによって状態遷移確率が異なる

$$V(s) := R(s) + \gamma \sum_{s'} P_{\pi(s)}(s, s') V(s')$$

いろいろ方法はある

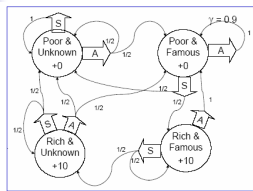
$$\pi(s) := \arg \max_a \sum_{s'} P_a(s, s') V(s') \quad 1$$

$$V(s) := R(s) + \gamma \sum_{s'} P_{\pi(s)}(s, s') V(s') \quad 2$$

Value Function

- Value Iteration
- Policy Iteration
- Modified Policy Iteration
- Prioritized Sweeping

例: Value Iteration



$$V(s) := R(s) + \gamma \max_a \sum_{s'} P_a(s, s') V(s')$$

k	V ^k (PU)	V ^k (PF)	V ^k (RU)	V ^k (RF)
1	0	0	10	10
2	0	4.5	14.5	19
3	2.03	8.55	18.55	24.18
4	4.76	11.79	19.26	29.23
5	7.45	15.30	20.81	31.82
6	10.23	17.67	22.72	33.68

興味がないのは、、、

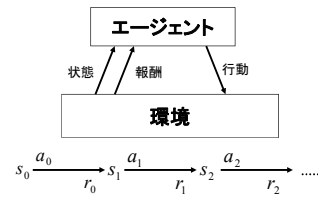
- もし遷移確率がわかっているなら、これは、単に計算の問題(とはいえ、それはそれで興味がある)、しかし、
- もし、遷移確率が未知だとしたら、どうしたらよいか?
- お手上げ?
- 強化学習が立ち向かう課題である

概要

- 強化学習とは
 - ある環境内で、状態を観測し行動する自律エージェントが、その目的を達するための最適方策を学習する方法
 - 学習は、行動の結果得られる報酬に基づく。報酬は、間接的で遅延を伴ってよい。受け取る報酬の累積を最大化する。
 - Q learning は、自分の行動が環境に与える影響に関する予備知識なしに、得られる遅延報酬から最適戦略を獲得する学習方法である。
 - 動的計画法に関連している

まずはイントロ

- エージェント
 - 環境(environment)の状態(state)を観測するセンサーを持ち、環境を変えることによる一連の行動(action)をとることができる



目標: (割引)累積報酬を最大化する行動の選択 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$
ただし $0 \leq \gamma < 1$ なぜ割引くか?

目標

- 各状態での行動に、その評価値(数値)を与える reward function を用いて定義することができる

エージェントの task

- 累積報酬(行動開始から終了まで)を最大化する制御方策(policy)を学習すること
- 制御方策 $\pi: S \rightarrow A$ (deterministic)
 - 関数近似の問題でもある $a = \pi(s)$

問題設定

- 行動の結果は、deterministic でも nondeterministic でもよい
- 行動が環境へ与える影響についての予備知識は、エージェントが持っているも持っていないもよい

特徴的な特徴を少々

- 遅延報酬 Delayed reward
 - 今貰う報酬は過去の行動の結果であるかもしれない。従って、訓練データは、一連の行動の結果である報酬列となる。($(s, \pi(s))$ ではない)
 - 貢献度分配問題 credit assignment problem
- 探索か利用か Exploration/Exploitation
 - エージェントがとる行動列に依存して、訓練データの分布が変わる
 - 探索とは未知の状態や行動をとる行動列(新しい情報を得ようとする行動列)をとること
 - 利用とは学習済みの知識のもと最大の報酬が得られる(累積報酬を最大化する)行動列を選ぶこと

- **部分観測状態 Partially observed states**
 - 状態の観測値一組が(実質的に異なる)複数の状態に対応すること(部分観測可能 partially observable). それまでの状態観測値列を用いて状態が絞ればよい(どれだけ記憶する?). 確率的に行動して平均化することを祈るか.
- **生涯学習 Life-long learning**
 - 複数のタスクを学習する場合に、もし、同じ(類似)環境にあるなら、過去に学習した方策や経験した報酬が、新しいタスクの学習に有向かもしれない
 - 環境は(自分の行動による以外に)刻々と変わるかもしれない. そのような環境変化に対応する.

タスクの記述

- **マルコフ決定過程 Markov Decision Process (MDP)**
 - 状態の有限集合 S , 行動の有限集合 A
 - 離散的な時間ステップ t ごとに,
 - エージェントは現在の状態 s_t を観測し、とるべき行動 a_t を選択する
 - 環境は報酬 $r_t = r(s_t, a_t)$ を与え、次の状態 $s_{t+1} = \delta(s_t, a_t)$ となる
 - マルコフ条件
 - $r(s_t, a_t)$ と $\delta(s_t, a_t)$ は現在の状態と行動にのみ依存
 - 関数 δ と r は non-deterministic であってよいし、エージェントにとって未知でよい
 - 今回は deterministic な場合のみ扱う.

- **エージェントのタスク**
 - 行動方策 $\pi : S \rightarrow A, \pi(s_t) = a_t$ を、累積報酬が最大化するように学習する
 - 任意の初期状態 s_t から開始し任意の行動方策 π に従った場合に得られる累積報酬は

$$V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$
 - $V^\pi(s)$: 割引付き(discounted)累積報酬(cumulative reward)
 - $0 \leq \gamma < 1$: 将来の報酬に対する割引率.
 - 累積報酬には別の定義がある(別の結果が得られる)

有限時間報酬 finite horizon reward 平均報酬 average reward

$$\sum_{i=0}^h r_{t+i} \qquad \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{i=0}^h r_{t+i}$$

- **最適方策 optimal policy**
 - 方策 π であって、任意の状態 $V^\pi(s)$ に対して次を最大化するもの

$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}} V^\pi(s), (\forall s)$$

$V^{\pi^*}(s) \rightarrow V^*(s)$

- **単純な grid-world 環境**
 - $\gamma = 0.9$
 - G: 目標状態, 吸収状態でもある
 - $0 + \gamma 100 + \gamma^2 0 + \gamma^3 0 + \dots = 90$
 - $0 + \gamma 0 + \gamma^2 100 + \gamma^3 0 + \dots = 81$

One optimal policy

Q learning

- 任意環境でエージェントが最適方策を学習できるようにするには、どうしたらよいだろうか?
 - 関数 $\pi^* : S \rightarrow A$ を直接学習するのは難しい
 - 訓練事例: 状態・行動対に対して得られた即時報酬の列
 - $i = 0, 1, 2, \dots$ について $r(s_i, a_i)$ ($\langle s_i, a_i \rangle$ ではない)
 - すなわち、状態・行動対上で定義される評価関数を学習する
 - エージェントが学習を試みるべき評価関数とは?
 - 状態 s における最適行動,

$$\pi^*(s) \equiv \operatorname{argmax}_a [r(s, a) + \gamma V^*(\delta(s, a))]$$
- 最適方策が学習できる. ただし、即時報酬関数 r と状態遷移関数 δ とに関する完全な知識がある場合には、という条件をつける
- しかし実際のところ、エージェントが r と δ の完全な知識を持つことは不可能

→ Q 関数を考えよう

Q 関数

Q 関数とは

- 状態 s から開始し最初の行動が a であるときの、割引き付き累積報酬の最大値

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$$

- 状態 s における最適行動 a , $\pi^*(s) = \arg \max_a Q(s, a)$
- もしエージェントが Q 関数を学習すると,
 - エージェントが関数 r と δ に関する知識を持っていなくとも、最適行動を選ぶことができる。
 - 行動を先読みした探索をしなくとも、最適行動が選択できる
 - Q 値を最大化する行動を選べば、最適方策となるからである

Q 学習のアルゴリズム

逐次近似

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$$

- Q と V^* の関係: $V^*(s) = \max_a Q(s, a')$
- Q の再帰的定義: $Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$
- \hat{Q} : Q に対する学習者の推定 (仮説).
 - 仮説 \hat{Q} は、普通、状態・行動対に関する大きな表 (Q 値表) を用いて表現される
 - 状態・行動対 $\langle s, a \rangle$ に関する値は $\hat{Q}(s, a)$

更新規則

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

- 結果として得られる新しい状態 s' と報酬 r とから学習する

Q 学習アルゴリズム (deterministic MDP に対して)

- 全 s, a について要素 $\hat{Q}(s, a)$ をゼロとする
- 現在の状態を観測する s
- 無限ループ
 - 行動 a を選び、それを実行する
 - 即時報酬を受領 r
 - 新しい状態を観測する s'
 - $\hat{Q}(s, a)$ に対して表を更新する

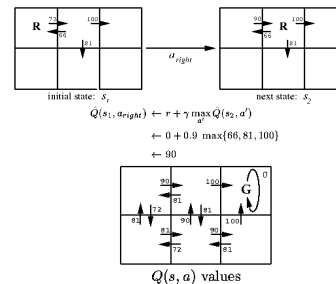
$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

deterministic Markov 決定過程
 r は上限あり
 どの状態・行動対も、無限回数、訪問される

例図

- 学習は、複数のエピソード(ゴールに到達すると1エピソード)から構成されていると仮定



- 個々のエピソードでは、エージェントはランダムに選択された初期状態から開始し、ゴールにいたるまで行動実行を繰り返す
- ゴール状態に到達すると、当該エピソードは終了し、エージェントは、次のエピソードに備え、新しい、ランダムに選ばれた初期状態に運ばれる。

Q 学習アルゴリズムの一般的な性質

- \hat{Q} 値は、学習期間中、減少しない。
 $(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$
- どの \hat{Q} 値も区間 $[0, \text{真の } Q]$ にある。
 $(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$

- Deterministic MDP.
- \hat{Q} 値の初期値はゼロ。
- 報酬は非負。

収束

- \hat{Q} は真の Q に収束する!

- 仮定
 - 系は deterministic MDP.
 - 即時報酬は上限あり。 $(\forall s, a) |r(s, a)| \leq c$
 - エージェントは、全ての状態・行動対を無限回訪問することになるよう、行動選択を続ける。

キーとなるアイデア

- 誤差最大の、表の要素 $\hat{Q}(s, a)$ の誤差は ϵ 因子 だけ、表の更新に伴い、減少する。

$\hat{Q}_n(s, a)$ は $Q(s, a)$ に $n \rightarrow \infty$ のとき収束する。 for all s, a .

略証.

- \hat{Q}_n を n 回更新後の表とする。また、 Δ_n を \hat{Q}_n 中の最大誤差とする; すなわち

$$\Delta_n \equiv \max_{s,a} |\hat{Q}_n(s,a) - Q(s,a)|$$
- $n+1$ 回目に更新される、表中のどの要素 $\hat{Q}_n(s,a)$ に対しても、更新後の推定値 $\hat{Q}_{n+1}(s,a)$ の誤差は

$$\begin{aligned} |\hat{Q}_{n+1}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s',a')) - (r + \gamma \max_{a'} Q(s',a'))| \\ &\leq \gamma \max_{s',a'} |\hat{Q}_n(s',a') - Q(s',a')| \\ &= \gamma \Delta_n \end{aligned}$$

$|\hat{Q}_{n+1}(s,a) - Q(s,a)| \leq \gamma^n \Delta_0, \Delta_n \rightarrow 0 \text{ as } n \rightarrow \infty$

実験方法

- エージェントはどうやって行動を選択すべきか?
 - 行動 a は $\hat{Q}(s,a)$ を最大化するように選ぶ (知識の利用)
 - リスク: 学習の初期段階で発見した高 \hat{Q} 値の行動にコミットしすぎる可能性がある
 - より高い値を持つ他の行動への探索が行われない可能性あり。
- 確率的アプローチをとろう

$$P(a_i | s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}} \quad k > 0$$
 - より高い \hat{Q} 値をもつ行動は、より高い確率をもつ
 - 大 $k \rightarrow$ (経験の) 利用, 小 $k \rightarrow$ 探索
 - k は繰り返し回数に応じて変更する (探索 \rightarrow 利用)

更新列

- 収束速度の改善方法 (学習の効率化)
 - 同一のエピソードを、時間的に逆順に、学習する
 - 繰り返し回数は少なくすむ、しかし必要メモリ量は大きい
 - 過去の状態・行動対を、即時報酬とともに記憶する
 - もし $\hat{Q}(s,a)$ が次の状態 $s' = \delta(s,a)$ の $\hat{Q}(s',a)$ によって決まり、かつ、それ以降の学習が $\hat{Q}(s',a)$ を変えるら、当該遷移 $\langle s, a \rangle$ を選び続けると $\hat{Q}(s,a)$ の値が変わりうる
- もし $r(s,a)$ と $\delta(s,a)$ が既知なら、もっと効率的な方法が可能である

報酬と行動が nondeterministic な時

- Nondeterministic な場合
 - 報酬関数 $r(s,a)$ と遷移関数 $\delta(s,a)$: 確率的
- Nondeterministic な Markov 決定過程
 - $r(s,a)$ と $\delta(s,a)$ の確率分布は現在の状態と行動にのみ依存する
 - 割引き付き累積報酬の期待値

$$V^\pi(s_t) \equiv E \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \right]$$
 - deterministic な場合も含む

- Q -値を再定義: Q の期待値をとる

$$\begin{aligned} Q(s,a) &\equiv E[r(s,a) + \gamma V^*(\delta(s,a))] \\ &= E[r(s,a)] + \gamma E[V^*(\delta(s,a))] \\ &= E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) V^*(s') \quad V^*(s) = \max_a Q(s,a) \end{aligned}$$
- 更新規則

$$\hat{Q}_n(s,a) \leftarrow (1 - \alpha_n) \hat{Q}_{n-1}(s,a) + \alpha_n \left[r + \max_{a'} \hat{Q}_{n-1}(s',a') \right]$$

ただし $\alpha_n = \frac{1}{1 + \text{visits}_n(s,a)}$

 - $\text{visits}_n(s,a)$: (この第 n 回目の更新を含め) これまでにこの状態・行動対を訪問した回数

- 現在の \hat{Q} 値の重みが時間とともに減少
- \hat{Q} 値の更新が、deterministic な場合に比べ、よりゆっくりとなる
- 学習時に、ある速度で α を小さくしていくと、正しい Q 関数に収束する
 - nondeterministic MDP に対する Q 学習の収束
 - 報酬は有界, 初期値は任意の有限値,

もし $0 \leq \alpha_n < 1$ かつ $\sum_{i=1}^{\infty} \alpha_{n(i,s,a)} = \infty, \sum_{i=1}^{\infty} [\alpha_{n(i,s,a)}]^2 < \infty$ ならば $\hat{Q}_n(s,a)$ は $Q(s,a)$ に $n \rightarrow \infty$ のとき確率1で収束, for all s, a .

Temporal difference learning

■ Q-学習

- 学習は、隣接状態の Q 値推定値の差を、繰り返し減少させることにより、行われる
- より一般的な *temporal difference algorithms* クラスの特別な形
 - エージェントが異なった時刻に得た Q 値推定値の差を、減少させることにより、学習が進行する
- Q 学習の学習規則：

$$Q^{(1)}(s_t, a_t) = r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

$$Q^{(n)}(s_t, a_t) = r_t + \gamma r_{t+1} + \dots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

■ TD(λ) by Sutton(1988)

$$Q^\lambda(s_t, a_t) = r_t + \gamma[(1-\lambda) \max_a \hat{Q}(s_t, a_t) + \lambda Q^\lambda(s_{t+1}, a_{t+1})] \quad 0 \leq \lambda \leq 1$$

■ もし $\lambda=0$,

- \hat{Q} 値の推測値の1-ステップ分の違いしか考えないことに相当

λ が増大するにつれ, アルゴリズムは, より離れた先読みとの差をより強めるようになる

■ もし $\lambda=1$,

- ただ観測された r_{t+1} 値のみを考えることになる. 現在の \hat{Q} の推測値からの貢献は考えない

TD(λ) を考えた動機

- ある状況では, より離れた先読みを考えた方が, 学習が進む

例からの一般化

■ Q 学習もつともきつい制約は

- ▶ 目標関数は、明示的に、表で表現され、個々の入力値(状態・行動対)に対して、それぞれに記述しないといけないこと
- 一種の丸暗記であり、未知の状態・行動対に対する Q 値の推定を、過去見た値から一般化しようということは、一切、試みられていない

■ 実用的なシステム

- 関数近似(例えば、ニューラルネット+BP)を Q 学習規則とを結合する. すなわち、状態・行動対の表をニューラルネットに置き換えて学習する. それぞれ $\hat{Q}(s, a)$ の更新を訓練事例とする:
 - 状態と行動を何らかの方法でコード化し、ネットワークの入力とし、出力の目的値として、Q学習の更新規則で得られる \hat{Q} を用いる

動的計画法との関係

■ 強化学習 (Q 学習) は MDP をとく動的計画法 dynamic programming に深く関係している.

- DP: $r(s, a)$ と $\delta(s, a)$ に関する完全な知識が必要
 - 計算の手間がより少ない方法で求解できる

■ ベルマン方程式 Bellman's equation

- 多くの動的計画法の基礎

$$(\forall s \in S) V^*(s) = E[r(s, \pi(s)) + \gamma V^*(\delta(s, \pi(s)))]$$

- 最適方策 π^* はこのベルマン方程式を満たし、ベルマン方程式を満たす任意の方策 π は最適方策である.