強化学習

公立小松大学

藤田 一寿

強化学習とは

- 数値化された報酬信号を最大にするために，何をすべきか（どのようにして状況に基づく動作選択を行うか）を学習する．（Sutton and Barto (三上，皆川訳) 強化学習）

- 答えは与えられていない．
  - 報酬という手掛かりがある．
- 試行錯誤で探す．
  - 環境に働きかけることで情報を得る．

強化学習の要素

エージェントは方策に従い行動し，報酬を受け取る．そして状態が変わる．

ハンターの例

どこに行けば獲物がたくさんとれるだろうか？

ハンター

ハンターは獲物をとりたい.
しかし，どこで獲物がよくとれるかわからない.
どこにいっても，とれることもあるし，とれないこともある.
ハンターは最も獲物をとれる確率が高いところを探したい.

ハンターの例

どこで獲物がよくとれるかわからないから色々なところで狩りをしよう.

ハンター

ハンターはどこで獲物がよくとれるかわからない.
全く手がかりがないので，ハンターはそれぞれの場所に行って狩りをしてみる.

ハンターの例

それぞれの場所で何度も狩りをしてみると，森が一番獲物を取ることができた．今後，森で狩りをしよう．

ハンター

ハンターはそれぞれの場所何度もに行って狩りをしてみる．
何度も狩りをしていると，獲物がとれる確率が高い場所がわかってくる．
獲物とれる確率が最も高い場所が分かれば，ハンターはそこにだけ狩りに行けばよい．

ハンターの例のまとめ

- ハンターは獲物がよくとれるかわからない.
- ハンターは手がかりがないので，それぞれの場所で狩りをする.
- それぞれの場所で何回か狩りをして，よく獲物がとれる場所を見つけられた.

ハンターの行動を強化学習と考えれば

- 狩場は環境とみなせる.
- ハンターが狩りをすることは行動とみなせる.
- 獲物は報酬とみなせる.
- 様々な場所で狩りをして獲物がとれるかどうか試すことを探索という.
- 試した結果を使うことを利用という.

強化学習の目的は何か（ハンターの例から）

- ハンターの目的は，最もよく獲物がとれる場所をなるべく早く探しだすこと．
  - 最も報酬（獲物）を得られる行動を決める最適な方策を探す．
- もしくは，最も効率良く，よく獲物がとれる場所を探し出すこと．
  - 行動の選択を通して得られる報酬の総和を最大化する．

探索と利用のトレードオフ

- ハンターはいろいろな場所で何度も狩りを行い，獲物がよくとれる場所を探す必要がある．
- ハンターとしては効率良く獲物がよくとれる場所を探したい．

- 探索する回数が少なければ少ないほどハンターは嬉しい．
- しかし，少ない探索回数から得た情報から良い狩場を見つけたとしても，その狩場より良い狩場があるかもしれない．

探索と利用のトレードオフ

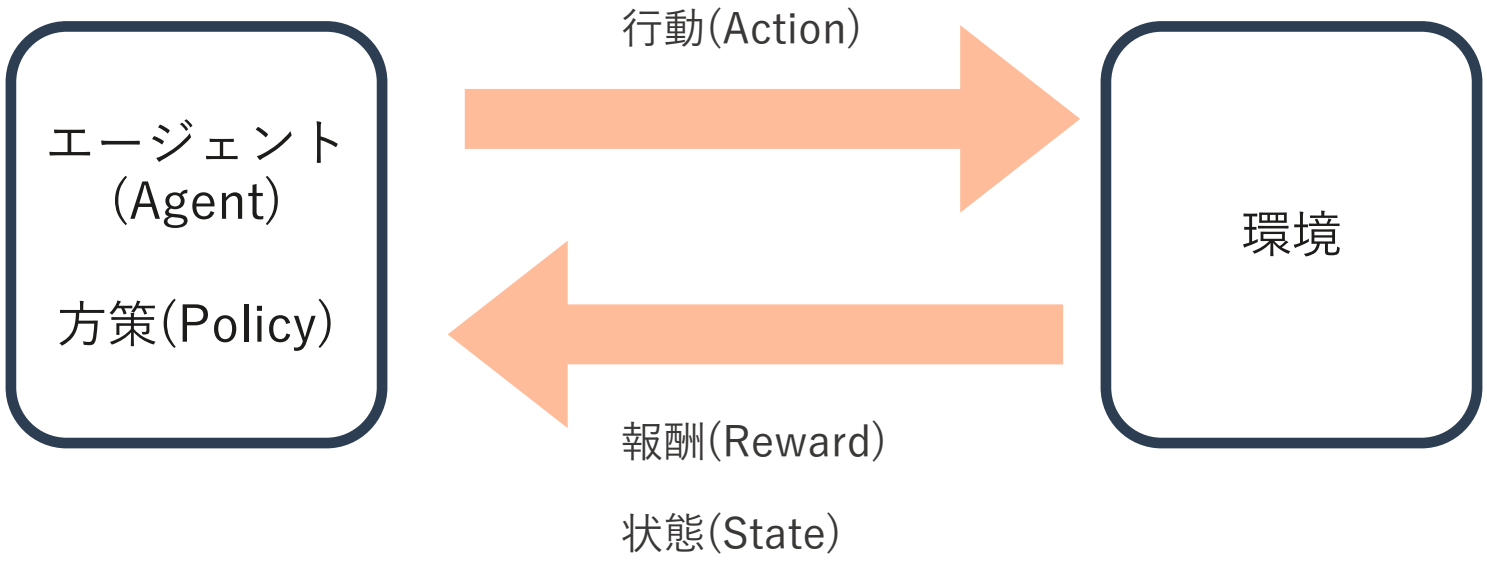
- 探索回数が多ければ多いほど正確に良い行動が見つかるが，良い行動をする回数は減り，探索を通して得られる報酬は少ないかもしれない.
- 一方，少ない探索回数から得られた情報を利用し良い行動を見つけても，もっと良い行動があるため得られる報酬は少ないかもしれない.
- これを探索と利用のトレードオフという.

不確かなときは楽観的に

- 探索しなければ正確に報酬を得られる確率を獲得できないが，正確に知るためには何回も探索しなければならない．（探索と利用のトレードオフ）

- 「不確かなときは楽観的に」という考え方で探索と利用のトレードオフを回避する．
- 「不確かなときは楽観的に」とは
  - エージェントが見積もっている報酬が得られる期待（報酬の期待値）が真の期待値より低いと探索されることが少なく修正されにくい．
  - 見積もりが真の期待値より高いと探索され，修正されることで見積もりが真の期待値に近づく．

強化学習の要素



エージェントは方策に基づき行動し，報酬を受け取る．
そして状態が変わる．

探索と知識利用のトレードオフ

- 今現在，最も良さそうな選択肢を選べば良い．
- しかし，他の選択肢を選ばないと他に良い選択肢があるかどうかわからない．
- 他の選択肢を調べてばかりでは，いつまでたっても良い選択肢を選べない．

- これまで得た知識を利用し最も報酬の高い行動をとろうとすると探索が減ってしまい，探索ばかりしていると得られる報酬は減ってしまう．

Multi armed bandit problem
多腕バンディット問題

Multi armed Bandit(MAB)問題

- 最も当たりを出す（最も期待値が高い）スロットマシンを探す問題
- 最もシンプルな強化学習の問題

当たる確率：0.3

当たる確率：0.7

当たる確率：0.1

当たる確率：0.8

どれを選べばよいのだろうか．
困った．

なぜmulti armed banditと呼ばれるのか？
Banditとは直訳すると盗賊の意味である．スロットマシンが人からお金を奪っていく様子を盗賊と言い表している．カジノでは1本の腕をスロットマシンがたくさん並んでいるが，それをたくさんの腕を持った盗賊(multi armed bandit)と比喩している．

MAB問題は何が問題なのか

- エージェントは事前にスロットマシンが当たる確率を知らない.
- エージェントが当たる確率を知るためには実際に試すしかない.
- 最も当たる確率が高いスロットマシンを早く見つけたい.
- 探す過程での報酬を高くしたい.
  - なるべく損したくない.

当たる確率：0.3

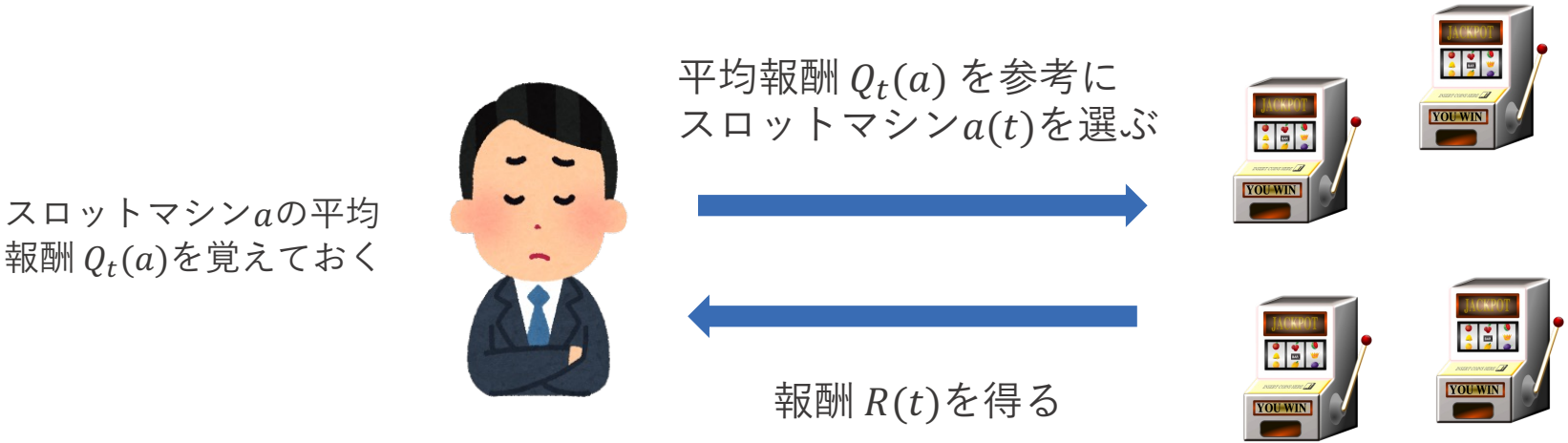
当たる確率：0.7

当たる確率：0.1

当たる確率：0.8

MABでの文字の定義

- $a(t)$：行動
- $R(t)$：報酬
- $Q_t(a)$：行動aをしたときの平均報酬
- MABでは行動はスロットマシンを選ぶことなので，状態は行動は一体となり，状態は考えない．



MAB問題を解く代表的なアルゴリズム

- greedyアルゴリズム
  - greedy（貪欲）に腕を選ぶ.
  - これまでの経験から最も報酬を得られる可能性が高い行動をする.
- ε -greedyアルゴリズム
  - これまでの経験から最も報酬を得られる可能性が高い行動をするが，たまに違う腕を選ぶ.
- UCB1アルゴリズム
  - これまでの経験から最も報酬を得られる可能性が高い行動をするが，あまり選んでいない腕も優先して選ぶ.

greedy algorithm

- まだn回選んだことがない腕がある場合，それを選ぶ．
- それ以外の場合，すべての腕に対しこれまでの報酬の平均$Q_t(a)$を計算する．
- $Q_t(a)$が最大の腕を選ぶ．

- 本当はあまり当たらない腕だが運良く当たりが多く出た場合，あまり当たらない腕ばかり選んでしまう可能性がある．その逆もありうる．

greedyアルゴリズムは間違った選択をするかもしれない

腕1
当たる確率：0.3

腕2
当たる確率：0.7

ステップ	報酬	
	腕1	腕2
1	1	1
2	1	0
3	1	
4	0	
5	0	
6	0	
7		1

たまたま当たりが続いた

腕1の情報が増える

腕1の方が当たりそうだ

たまたま当たったため，間違った選択をする．

腕1はハズレのようだ．腕2の方が良いかも

ε greedy algorithm

- まだ選んだことがない腕がある場合，その腕から一つ選ぶ.
- ε の確率ですべての腕からランダムに一つ選ぶ.
- 1- ε の確率で最も平均報酬$Q_t(a)$が高い腕を選択する.

- 十分試行を繰り返せば最も当たる腕を探せる.
- 一定の確率でランダムに行動を選択してしまうので，十分探索したあとでも最も当たる腕を選び続けることができない.

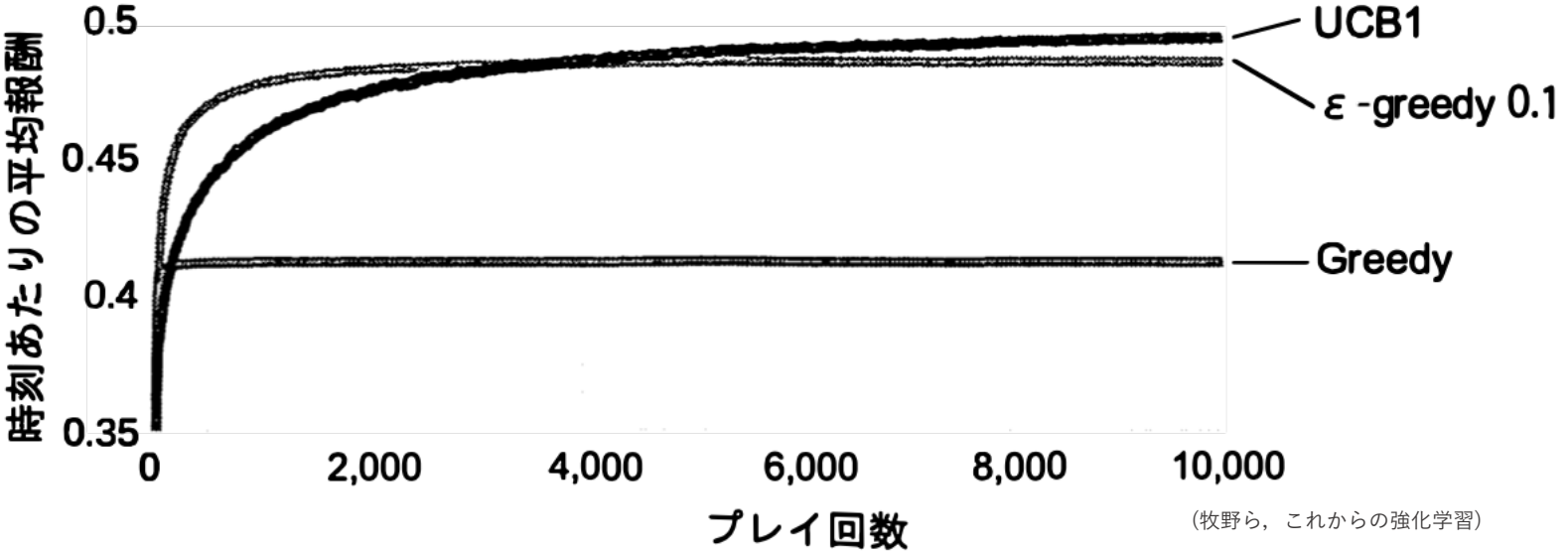
UCB1

- まだ選んだことがない腕がある場合，その腕から一つ選ぶ．
- $U = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$が高い腕を選択する．$t$はステップ数，$N_t(a)$は腕$a$を選んだ回数である．

- $Q_t(a)$が平均報酬が最も高い腕を選ばせようとし，$\sqrt{\frac{\ln t}{N_t(a)}}$があまり選ばれていない腕を選ばせようとする．
- $N_t(a)$が大きくなると$\sqrt{\frac{\ln t}{N_t(a)}}$は小さくなる（$\ln t$は対数なのであまり増加しない）．つまり，それぞれの腕がそれなりに選ばれれば，平均報酬が最も高い腕を選ぶことになる．

シミュレーション結果

- Greedyアルゴリズムはたまたま悪いスロットマシンを選びたまたま報酬が得られてしまったため，最適な行動ができず報酬が低い．
- UCB1も ε -greedyアルゴリズムも報酬が高いが，　ε -greedyアルゴリズムはランダムに行動するため悪い行動もせざるを得ず平均報酬はUCB1に負ける．



（牧野ら，これからの強化学習）

式をいじって色々考察してみる

平均報酬

- 行動が一つしかない場合，nステップ目の行動に使う平均報酬は次のように書ける．
- $Q_n = \frac{R_1+R_2+\cdots+R_{n-1}}{n-1}$
- $n+1$ステップ目では

$$Q_n = \frac{R_1 + R_2 + \cdots + R_n}{n} = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) = \frac{1}{n}\left(R_n + \frac{n-1}{n-1}\sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}(R_n + (n-1)Q_n) = \frac{1}{n}(R_n + nQ_n - Q_n) = Q_n + \frac{1}{n}(R_n - Q_n)$$

- 平均報酬を報酬の推定値だとすると

  新しい推定値 ← 古い推定値 + ステップサイズ(得られた報酬 − 古い推定値)

- と見なせる．

単純なバンディットアルゴリズム

- 先の式を使うと，多腕バンディット問題を解く単純なアルゴリズムは次のようになる．

- 腕はk本ある．それぞれの腕に対し平均報酬$Q(a)$と選んだ回数$N(a)$がある．行動をAとする．
- まず，$a = 1$からkまでについて次のように初期化する．
  - $Q(a) \leftarrow 0, N(a) \leftarrow 0$
- 次の処理をループさせる．
  - $1 - \varepsilon$の確率で$Q(a)$が最大の行動Aを取る．
  - εの確率でランダムに行動を選ぶ．選ばれた行動はAとする．
  - 実際に行動し報酬Rを得る．
  - $N(A) \leftarrow N(A) + 1$
  - $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big(R - Q(A)\big)$

非定常な問題のときの報酬の期待値

- 先の平均報酬の式から，報酬の期待値が次のように求まるとする．
- $Q_{n+1} = Q_n + \alpha(R_n - Q_n)$
- αはステップサイズパラメタで 0 から 1 の値を取る．
- Q_{n+1}を式変形すると次のようになる．

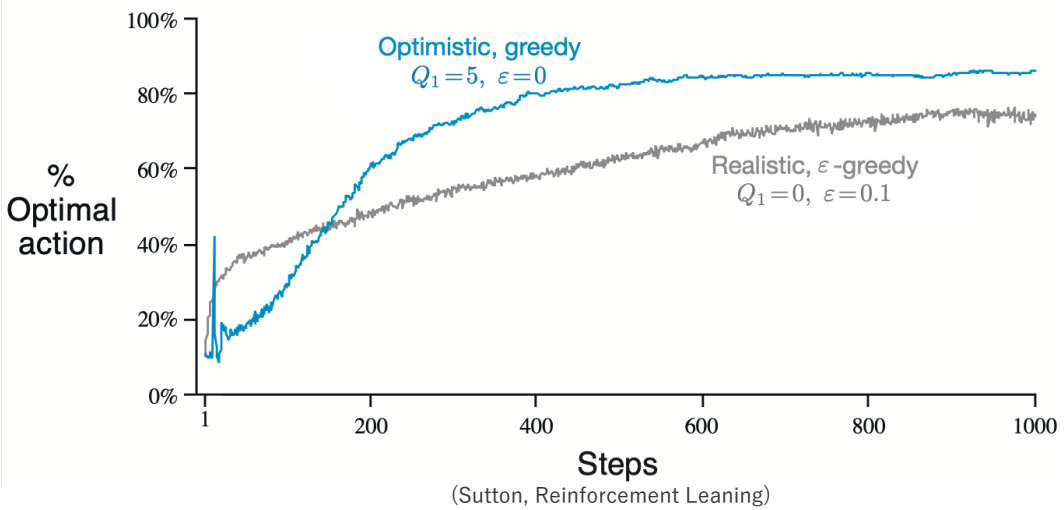
$$
\begin{aligned}
&Q_{n+1} = Q_n + \alpha(R_n - Q_n) = \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)(\alpha R_{n-1} + (1-\alpha)Q_{n-1}) = \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \cdots + (1-\alpha)^{n-1}\alpha R_{n-1} + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}
$$

非定常な問題のときの報酬の期待値

- $Q_{n+1} = Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$
- この式は，過去に得た報酬の情報ほど使われない．
  - 過去に得た報酬 R_iは $(1-\alpha)^{n-i}$がかけてある．
  - $(1-\alpha)$は1より小さいため， iが小さければ小さいほど（過去ほど）$(1-\alpha)^{n-i}$が小さくなる．
  - つまり，より古い R_iはあまり使われなくなる．
- 最近の情報をより使うため，状況が変化しても対応可能となる．
  - 過去の間違った行動を忘れるとも解釈できる．

楽観的な行動

- n+1ステップでの報酬の期待値は$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$と書ける.
- Q_1はエージェントがはじめに想定している期待値と見なせる.
- つまり，この値が大きければ楽観的で低ければ悲観的に思っているといえる.



(Sutton, Reinforcement Leaning)

$Q_1 = 5$と見積もったgreedyアルゴリズムは，はじめに楽観的に期待値を見積もることで素早く良い行動を探し当てている．Greedyな行動をしているにも関わらずである.

MAB問題の応用事例

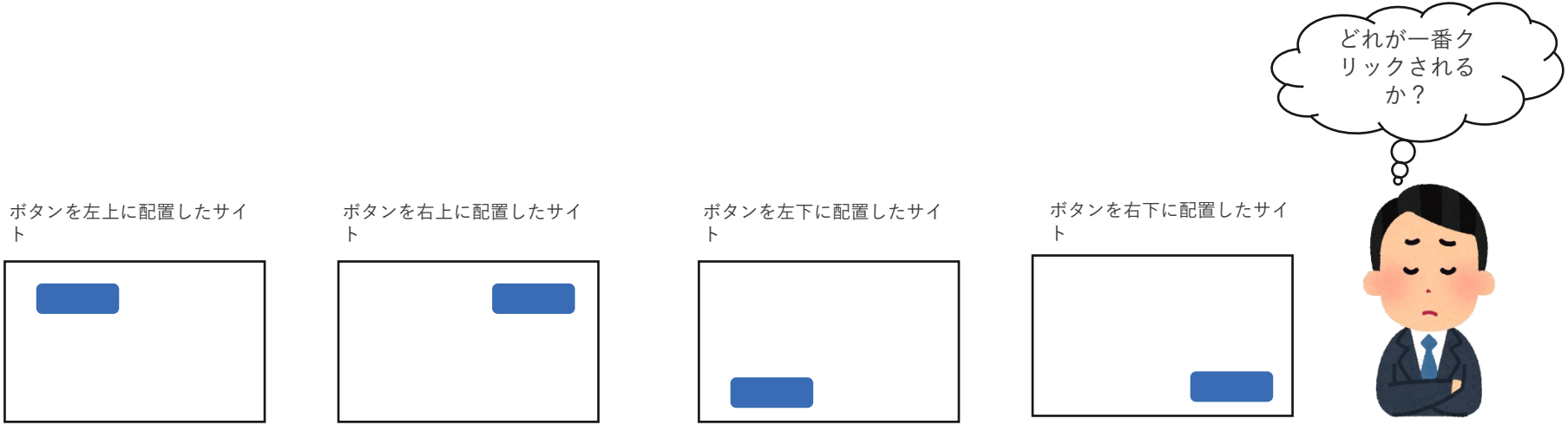
MAB問題の応用

- 配置問題
- モンテカルロ木探索
- 深層強化学習
  - AlphaZero

配置問題

配置問題

- ウェブサイトのどこに，ボタンをおけば最もクリックされるのか？
- スロットマシンは，それぞれのボタン配置に対応する．
- 報酬は，クリックに対応する．
- MABアルゴリズムを用いそれぞれのボタン配置を試しながら，最適な配置を探す．



モンテカルロ木探索

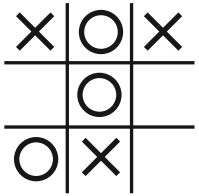
モンテカルロ木探索とは

- ゲーム木を探索するためアルゴリズム.
- 囲碁AIのアルゴリズムとして有名.
- 多腕バンディットが活用されている.

- モンテカルロ木探索（Monte Carlo Tree Search: MCTS）

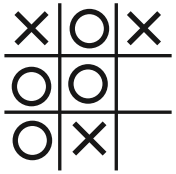
マルバツゲームを解く

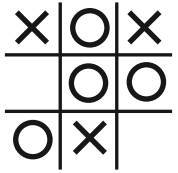
現在の盤面

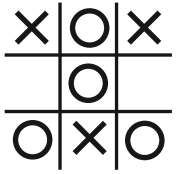


どの手を選べば良いかな？

選べる盤面



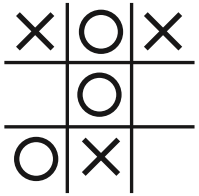




選べる盤面を見ても勝てるかどうか分からない.

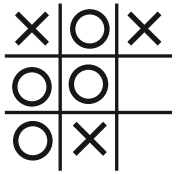
マルバツゲームを解く

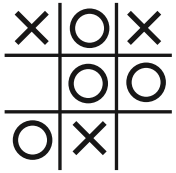
現在の盤面

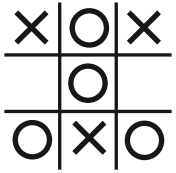


どの手を選べば良いかな？

選べる盤面

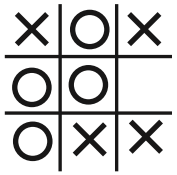




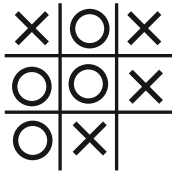


選べる盤面を見ても勝てるかどうか分からない．

1手先を考えてみるか．

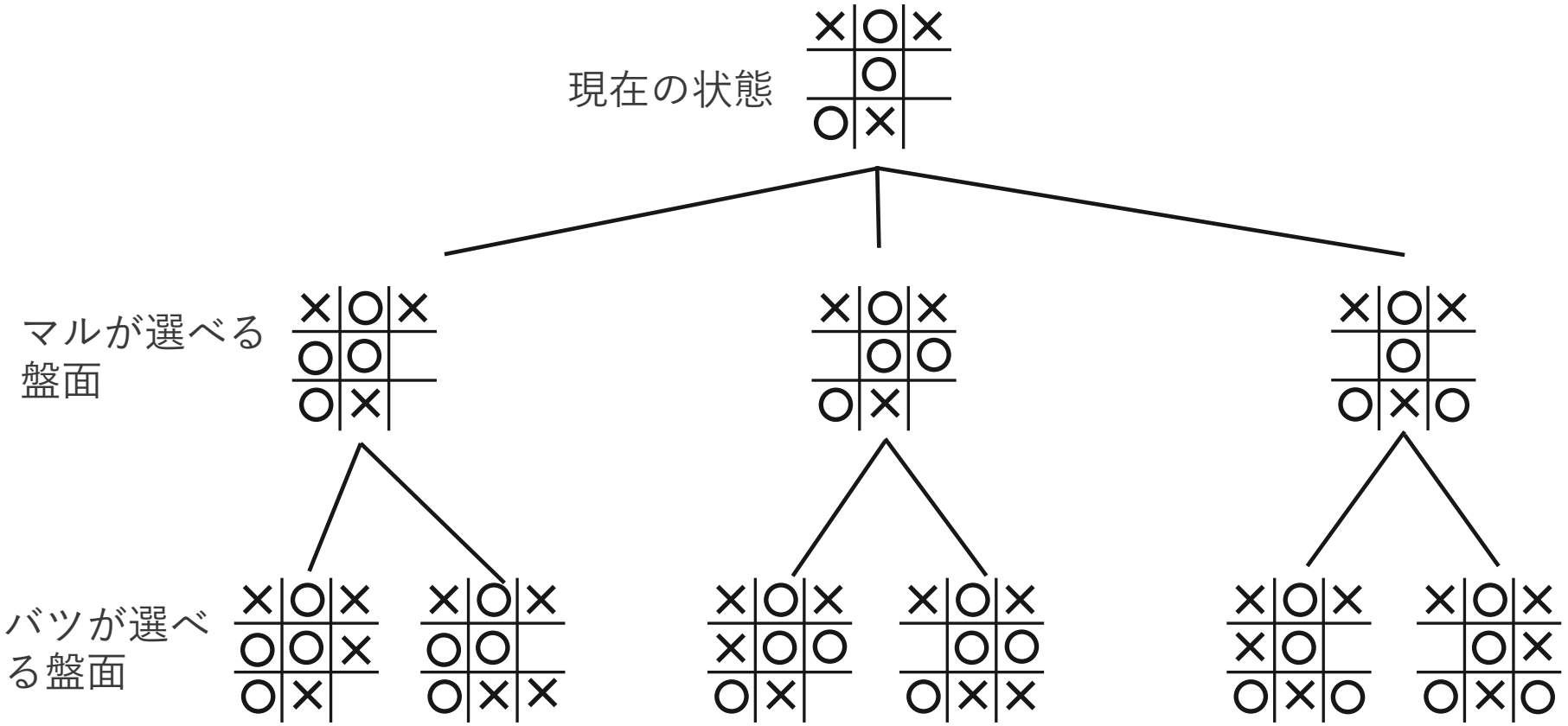


勝ち



1手読んでみると，1番上の盤面に勝ち目があることが分かる．

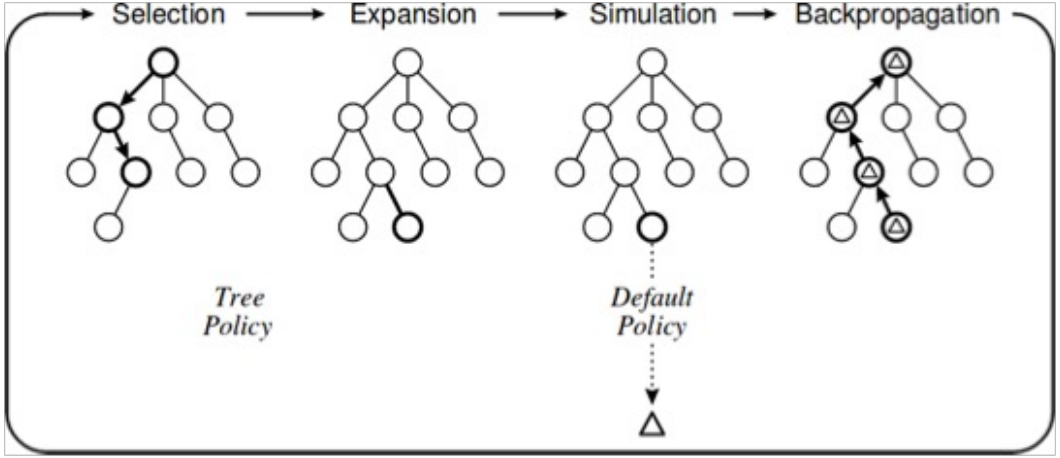
マルバツゲームのゲーム木の例



起こりうる盤面の一覧．起こりうる盤面を木構造で書いたものをゲーム木という．

モンテカルロ木探索のアルゴリズム

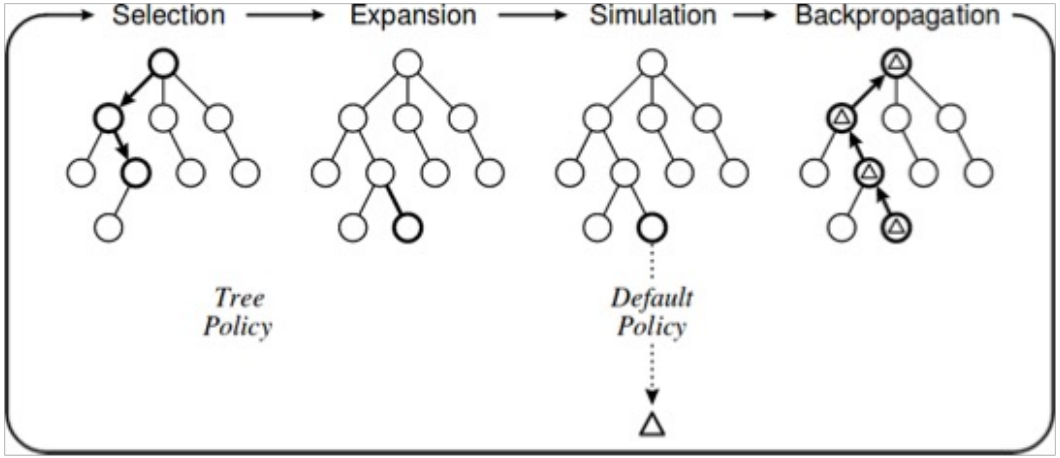
- モンテカルロ木探索は先程のゲーム木を探索するアルゴリズムである.
- モンテカルロ木探索は次の4つの処理で構成される.
- Selection:
- Expansion
- Simulation
- Backpropagation



(Browne et al., 2012)

モンテカルロ木探索のアルゴリズム

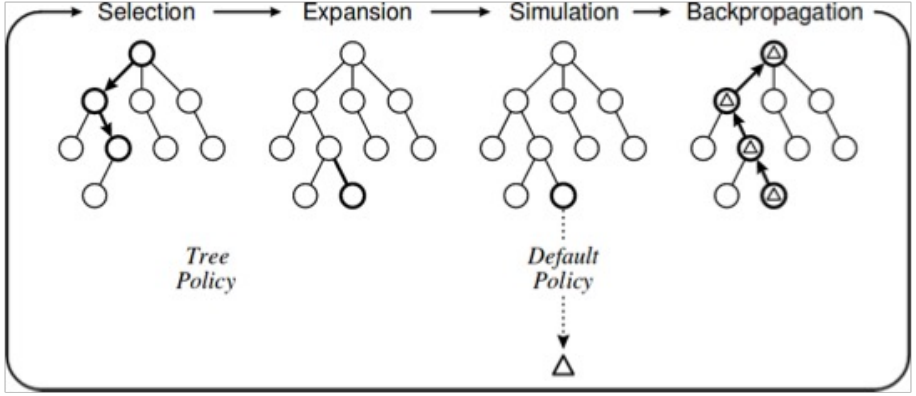
- Selection:
  - ランダム，greedy， ε -greedy，UCB1などを用い手（子ノード）を選ぶ(バンディット問題).
  - 葉ノードまで続ける.
- Expansion
  - N回葉ノードを訪れたら，その葉ノードを展開する.



(Browne et al., 2012)

モンテカルロ木探索のアルゴリズム

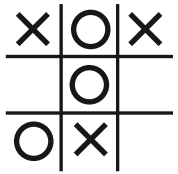
- Simulation
  - 葉ノードから終局までランダムに手を選ぶ.
- Backpropagation
  - 勝敗の結果を親ノードに伝える．それをルートノードまで繰り返す.
  - selectをルートノードからやり直す.
- N回，Selection，Expansion，Simulation，Backupを行った後，ルートノードの子ノードのうち最も選んだ回数が多かった子ノードの手を打つ.



(Browne et al., 2012)

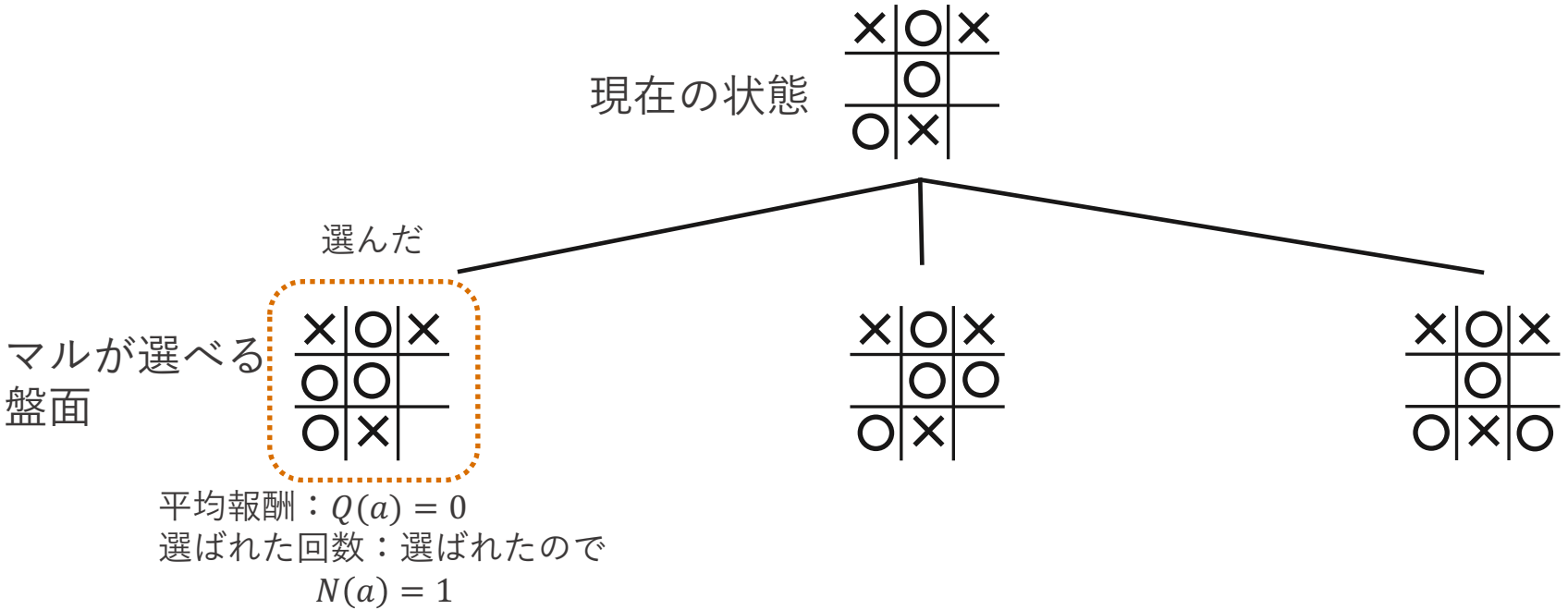
モンテカルロ木探索の例

現在の状態



起こりうる盤面を列挙し，木に追加する．

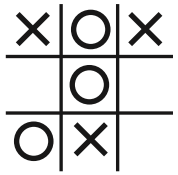
モンテカルロ木探索の例

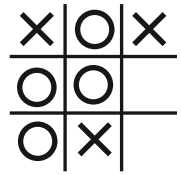


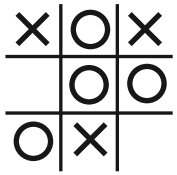
Selection: MABアルゴリズムで盤面を選ぶ．各盤面は平均報酬と選ばれた数を保存している．
Expantion: 選んだ盤面がN回選ばれていれば盤面を追加する．
今回は1度目なので追加しない．

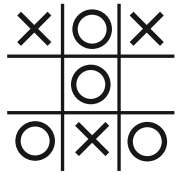
モンテカルロ木探索の例

現在の状態



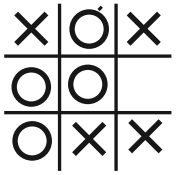






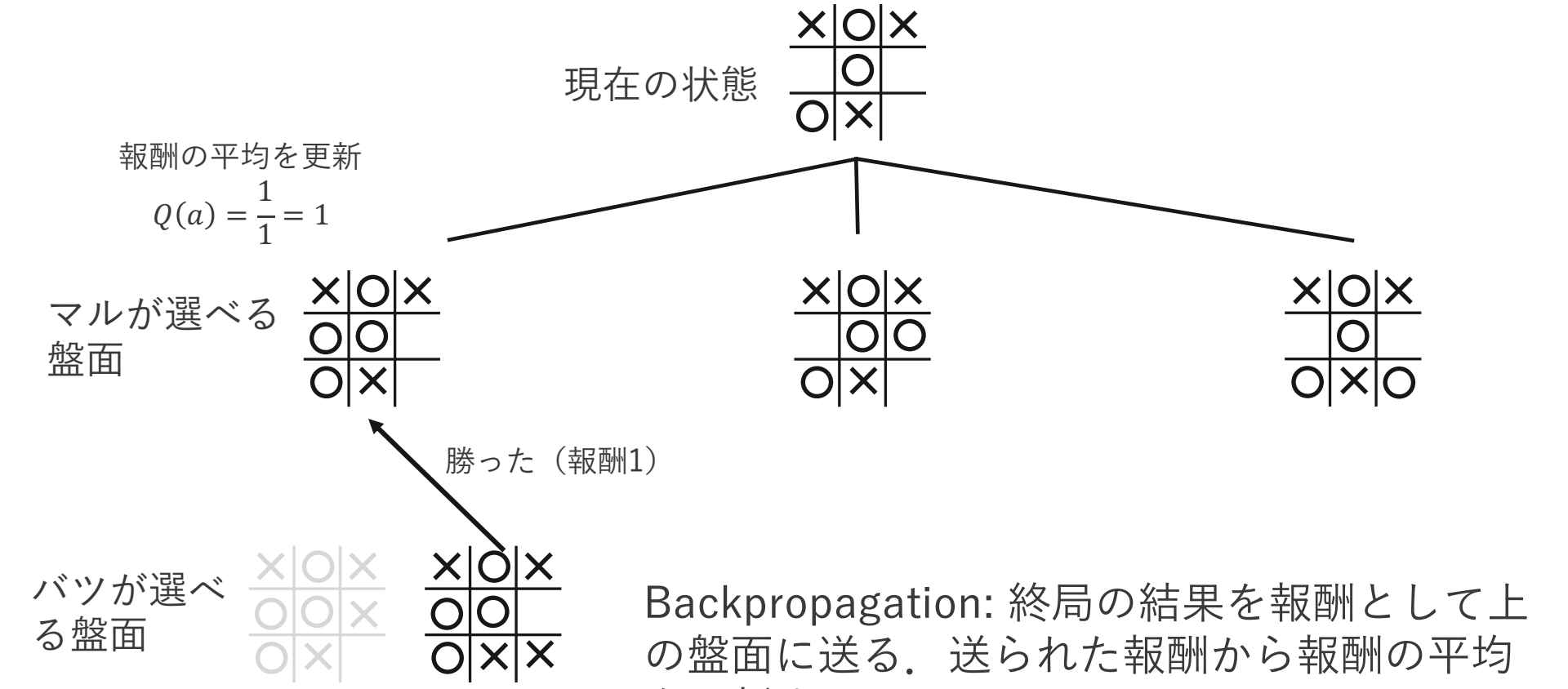
マルが選べる盤面

バツが選べる盤面



Simulation: 起こりうるランダム盤面をランダムに終局まで選び続ける.

モンテカルロ木探索の例



Backpropagation: 終局の結果を報酬として上の盤面に送る．送られた報酬から報酬の平均を更新する．

AlphaZero

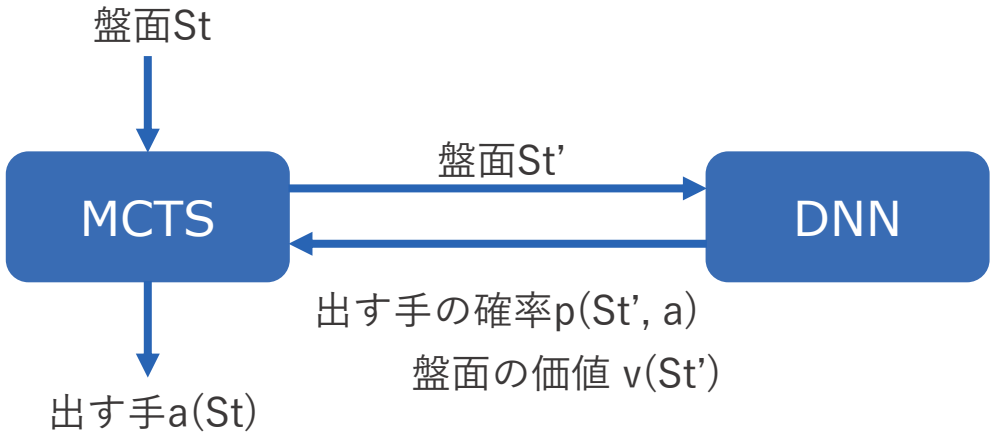
モンテカルロ木探索の欠点

- モンテカルロ木探索は探索回数が多ければ多いほど正確な結果を導ける.
- しかし，手数が多いゲームでは，計算時間の制限から探索しきれないため正確な結果を出せないかもしれない.
  - ゲーム木が幅広く深い場合は，探索空間が広く探索回数を増やさないと良い行動を導けない.
- 計算時間が制限されているため探索回数は限られている．その条件下でより正確な結果を導く必要がある.

- この問題を深層ニューラルネットワーク（DNN）で解決したのがAlphaZeroである.

AlphaZeroの概要

- 深層ニューラルネットワークとモンテカルロ木探索(MCTS)で構成構成される.
- 深層ニューラルネットワークは盤面から，出す手の確率と勝敗の結果を推定する.
- モンテカルロ木探索で出す手を決定する.

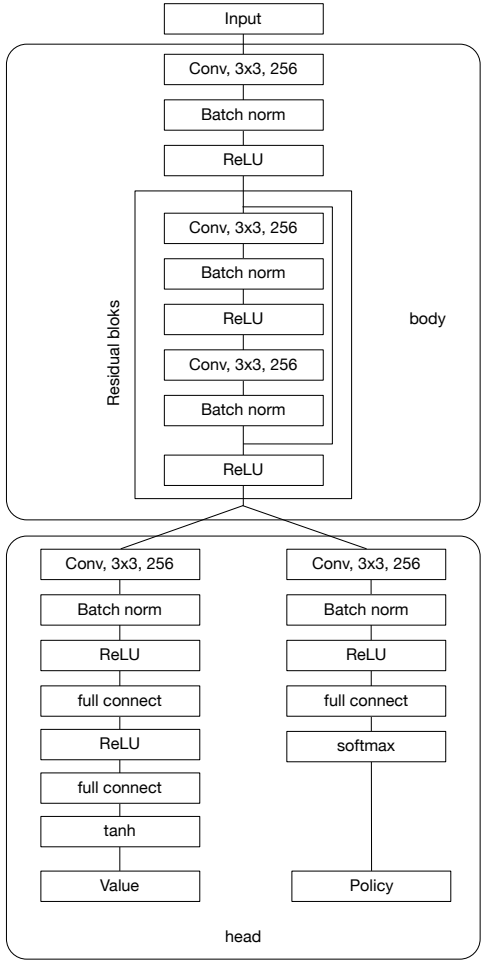


Deep neural networkの学習

- 学習データは，盤面と最終的な勝ち負け，その盤面での出す手の確率を用いる．
- 出す手の確率はモンテカルロ木探索における盤面を選んだ回数を用いる．
- 学習データはAlphaZero同士の対戦（自己対戦 ：self-play）の結果のみ用いる．
- 事前に学習データを用意する必要はない．

AlphaZeroのニューラルネットワーク

- 価値と方策を出力するネットワーク
- Policy headとvalue headからなるヘッド部とボディー部で構成される.
- Policy headの出力が出す手の確率
- Value headの出力が推定した結果
  - player1が勝てば１，負ければ-1
- ボディー部はn個のresidual blocksで構成される.
- 入力はk手分の盤面+現在のプレーヤ．盤面はプレーヤごとに用意する．それぞれ0，1で表される.

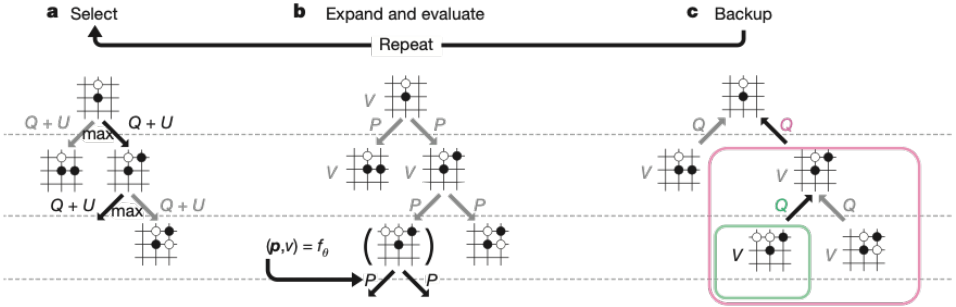


AlphaZeroのモンテカルロ木探索

- Select:
  - Q(s,a)+U(s,a)が最も大きい子ノードを選ぶ．
  - 子ノードが葉ノードでなければ，上に戻る．そうでなければ次へ．
- Expand and evaluate
  - 葉ノードを展開し，次へ．
- Backup
  - (推定した)結果を親ノードに 伝える．それをルートノードまで繰り返す．
  - selectをルートノードからやり直す．
- N回上記の処理を行った後，ルートノードの子ノードのうち最も選んだ回数が多かった子ノードの手を打つ．

$$Q(s,a) = \frac{W(s,a)}{N(s,a)}$$

$$U(s,a) = C_{\mathrm{puct}} p(s,a) \sqrt{N(s)} / N(s,a)$$



(Silver et al., 2017)

モンテカルロ木探索との違い

- AlphaZeroはモンテカルロ木探索の一種である.
- モンテカルロ木探索との違いは，Simulationにおいて終局までゲーム木を探索しない点である.
- AlphaZeroは終局までゲーム木を探索しない．報酬は深層ニューラルネットワークで計算する.
- モンテカルロ木探索の場合，盤面から得られる報酬は終局での勝ち負けで算出したが，AlphaZeroでは深層ニューラルネットワークから算出された勝率を報酬とする.

一般的な強化学習

MABと一般的な強化学習

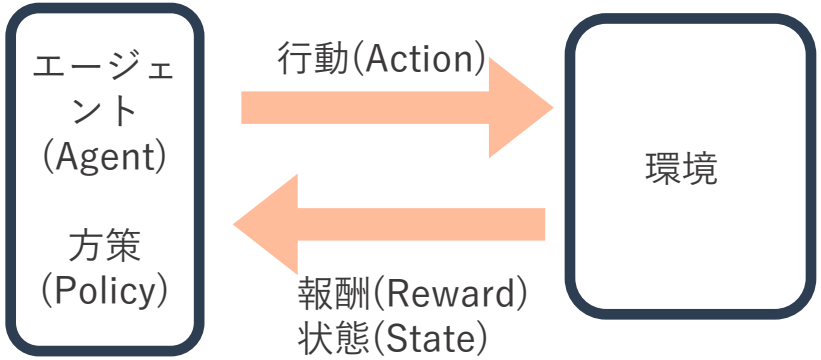
- MABでは行動はスロットマシンを選ぶことなので，状態は行動は一体となり，状態は考えない．
- 一般的には，行動と状態は同じではない．

MAB

平均報酬 $Q_t(a)$ を参考に
スロットマシン$a(t)$を選ぶ

報酬 $R(t)$を得る

一般的な強化学習



強化学習の要素

- 状態がN種類あるとすると状態集合は$S = \{s_1, s_2, \dots, s_N\}$と表せる．
  - 時刻tのとき状態はS_tとし，Sの要素のうちのいずれかである．
- 状態sのとき，取りうる行動がM種類あるとすると行動集合$A(s) = \{a_1, a_2, \dots, a_M\}$と表せる．
  - 時刻tにおける行動A_tは$A(S_t)$の要素のうちいずれかの値をとる
- 時刻tにおいて状態S_tで行動$A(S_t)$をし，状態がS_{t+1}に変わったとき，環境から受け取る報酬をR_{t+1}とする．

$A(s)$
$=$ {池に行く，草原に行く，森に行く，帰る}

$S = \{池, 草原, 森, 家\}$

全ては確率で決まる

- 時刻tのとき状態がs'，報酬がrであるとすると，これらは次の確率で決まる．
- $p(s', r \mid s, a) = \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$
- これは以前の状態がs'，その時とった行動がaであるという条件のもとでの s'， rの確率である．

- 状態sで行動aを行ったとき，状態s' が起こる確率は，先の確率を報酬rで周辺化すれば良い．
- $p(s' \mid s, a) = \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathrm{R}} p(s', r \mid s, a)$

報酬の期待値

- 報酬の期待値（期待報酬）は状態と行動で決まるため，sとaの変数である．
- $r(s,a) = E[\,R_t \mid S_{t-1} = s, A_{t-1} = a\,] = \sum_{r\in \mathrm{R}} r\, p(\,r \mid s,a\,) = \sum_{r\in \mathrm{R}} r \sum_{s'\in S} p(\,s', r \mid s,a\,)$

  周辺化
- また，状態sのとき行動aとったとき，状態がs'になり報酬rをもらうため，報酬はs, a, s'の3変数の関数で表現できる．
- $r(s,a,s') = E[\,R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\,] = \sum_{r\in \mathrm{R}} r\, p(\,r \mid s,a,s'\,) = \sum_{r\in \mathrm{R}} r \dfrac{p(\,r, s' \mid s,a\,)}{p\left(\,s' \mid s,a\,\right)}$

  ベイズ定理

未来の報酬

- 単純に未来に得られるすべての報酬の期待値，期待報酬は次のように書ける．
- $G_t = R_{t+1} + R_{t+1} + R_{t+1} + \cdots + R_T$
- Tは最終時間ステップである．
- これは単純に同じ重みで未来のすべての報酬を足している．

割引報酬

- 未来になればなるほど予測は外れると考えると，遠い未来の報酬と近い未来の報酬が同じ重みで足されるのではなく，遠い未来の報酬は少なめに足したほうが良いだろう．
- そこで，未来の報酬を割り引いた割引報酬を使う．
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1}$
- γを割引率といい，0から1の間の値を取る．

価値関数

- エージェントはそもそも何が目的だろうか？
- 未来の報酬を最大化することが最大の目的だろう．
- つまり，未来の報酬を最大化する状態にしたいとエージェントは考える．
- 状態sで得られる割引報酬の期待値を価値関数といい次のように書く．
- $v_\pi(s) = E_\pi[\, G_t \mid S_t = s \,] = E_\pi[\, \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \,]$
- πは方策で，エージェントの行動のルールを表す．つまり，この価値関数は方策 πをとるエージェントが状態sになったとき得られる割引報酬の期待値である．

価値関数

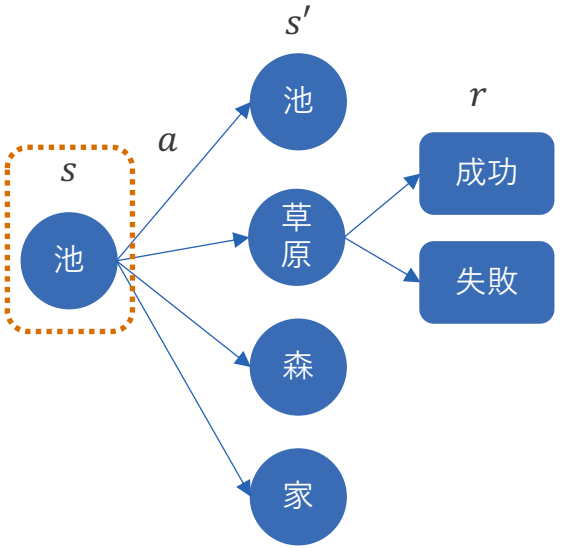
$$v_\pi(s) = E_\pi[\, G_t \mid S_t = s \,] = E_\pi[\, R_t + \gamma G_{t+1} \mid S_t = s \,]$$
$$= E_\pi[\, R_t \mid S_t = s \,] + \gamma E_\pi[\, G_{t+1} \mid S_t = s \,]$$

第1項は

$$E_\pi[\, R_t \mid S_t = s \,] = \sum_{s',r,a} p(r, s' \mid s, a)\pi(a \mid s)\, r$$

と書ける.

- 状態sのときの価値関数は，その状態のときに将来得られる報酬の期待値である.
- 状態sから報酬を得るには行動aをしなければならない．状態sのとき得られる報酬の期待値を計算するには，状態sのときに行えるすべての行動を考慮しなければならない.
- 状態sから
- 状態sのとき行動aをする確率は$\pi(a \mid s)$である．πは行動を決めるのでエージェントの方策とみなせる.



価値関数

$$E_\pi[\, G_{t+1} \mid S_t = s \,]$$

$$= \sum_{s',r,a} p(r, s' \mid s, a)\pi(a \mid s)\, E_\pi[G_{t+1} \mid S_{t+1} = s']$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) E_\pi[G_{t+1} \mid S_{t+1} = s']$$

$v_\pi(s) = E_\pi[\, G_t \mid S_t = s \,]$だから

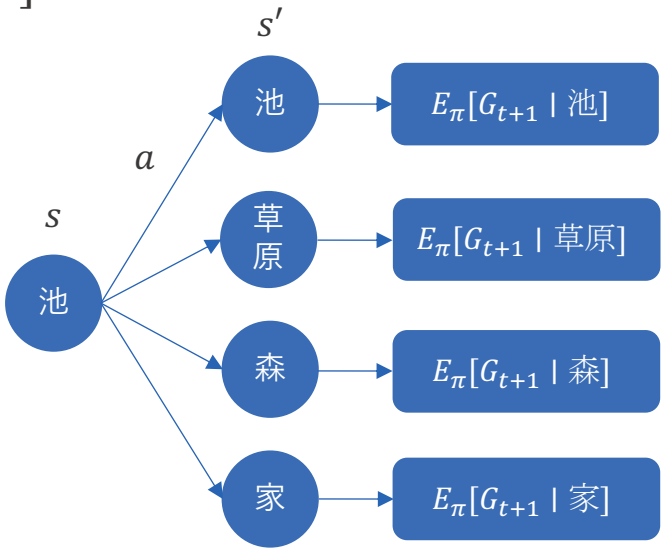
$$E_\pi[\, G_{t+1} \mid S_t = s \,]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) v_\pi(s')$$

よって

$$v_\pi(s) = E_\pi[\, G_t \mid S_t = s \,] = E_\pi[\, R_t + \gamma G_{t+1} \mid S_t = s \,]$$

$$= E_\pi[\, R_t + \gamma v_\pi(s') \mid S_t = s \,]$$



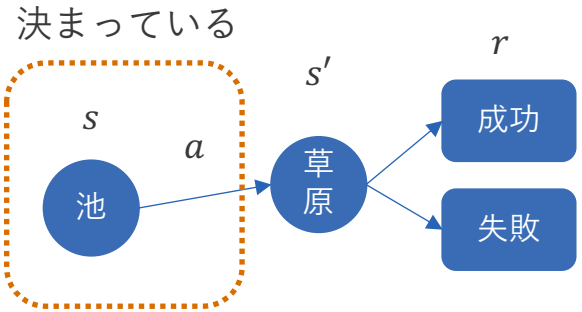
行動価値に対するベルマン方程式

$$v_\pi(s) = E_\pi[\, R_t + \gamma v_\pi(s') \mid S_t = s \,]$$
$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)(r + \gamma v_\pi(s'))$$

これを行動価値に対するベルマン方程式という.

行動価値関数

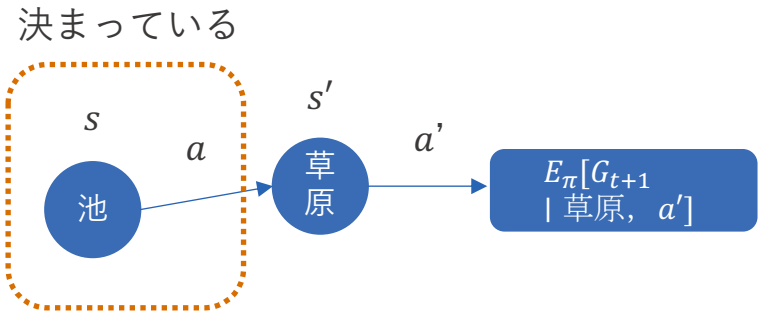
- 状態sになるためには行動しなければならない．そう考えれば，価値観数は行動aにも依存しているだろう．
- 行動aを考慮した価値関数を行動価値関数といい，次のように書く．
- $q_\pi(s,a) = E_\pi[\, G_t \mid S_t = s, A_t = a \,]$



行動価値関数

$$q_\pi(s,a) = E_\pi[\, G_t \mid S_t = s, A_t = a \,] = E_\pi[\, R_t + \gamma G_{t+1} \mid S_t = s, A_t = a \,] =$$
$$= E_\pi[\, R_t \mid S_t = s, A_t = a \,] + \gamma E_\pi[\, G_{t+1} \mid S_t = s, A_t = a \,]$$
$$= \sum_{s\prime,r} p(s', r \mid s, a)r + \gamma \sum_{s\prime,r} p(s', r \mid s, a) E_\pi[G_{t+1} \mid S_t = s', A_t = a']$$
$$= \sum_{s\prime,r} p(s', r \mid s, a)r + \gamma \sum_{s\prime,r} p(s', r \mid s, a) q_\pi(s', a')$$
$$= E_\pi[\, R_t + \gamma q(s', a') \mid S_t = s, A_t = a \,]$$

これを行動価値に対するベルマン方程式という.



エージェントはどう行動すればよいか

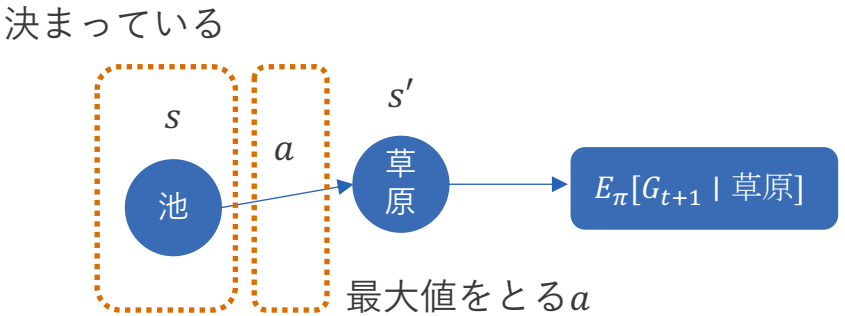
- エージェントの目的は報酬の最大化である.
- 我々が知りたいのは，エージェントが目的を達するためにはどのような行動（方策）をとればよいかである.
- 最も良い方策π_*とすると，方策π_*をとったときの価値関数は
- $v_{\pi_*}(s) = \max_{\pi} v_{\pi}(s) = \max_{\pi} E_{\pi}[\, G_t \mid S_t = s \,]$
- と書ける．これを最適状態価値関数と呼ぶ.
- 方策π_*をとったときの行動価値関数は
- $q_{\pi_*}(s,a) = \max_{\pi} q_{\pi}(s,a) = \max_{\pi} E_{\pi}[\, G_t \mid S_t = s, A_t = a \,]$
- と書ける．これを最適行動価値関数と呼ぶ.

価値に対するベルマン最適方程式

- 最適な方策π_*をとったときの価値関数は

$$
\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) = \max_{a \in A(s)} E_{\pi_*}[\, G_t \mid S_t = s, A_t = a \,] \\
&= \max_{a \in A(s)} E_{\pi_*}[\, R_t + \gamma G_{t+1} \mid S_t = s, A_t = a \,] \\
&= \max_{a \in A(s)} E_{\pi_*}\left[\, R_t + \gamma v_{\pi_*}(S_{t+1}) \mid S_t = s, A_t = a \,\right] \\
&= \max_{a \in A(s)} \sum_{s', r} p(\, s', r \mid s, a \,)\left(r + \gamma v_{\pi_*}(S_{t+1}) \right)
\end{aligned}
$$

- これを価値に対するベルマン最適方程式という.

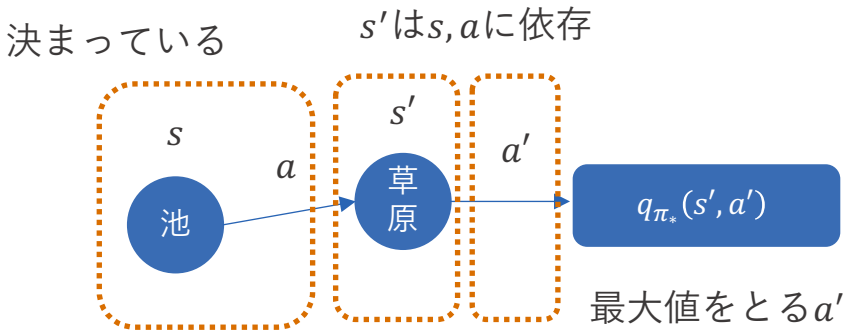


行動価値に対するベルマン最適方程式

- 最適な方策π_*をとったときの価値関数は

$$q_*(s,a) = E\left[R_t + \gamma \max_{a'} q_{\pi_*}(S_t, a') \,\middle|\, S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left(r + \gamma \max_{a'} q_{\pi_*}(s', a')\right)$$

- これを，行動価値に対するベルマン最適方程式という．



エージェントはどうこうどうすればよいか

- 最適価値関数v_*が分かっていれば，v_*が最大の行動を取れば良い.
- しかし，実際は分からない.
- エージェントは過去の試行錯誤からえた情報から最適価値関数もしくは最適行動価値関数を得なければならない.