

自然言語処理 — 強化学習入門 —

<https://satoyoshiharu.github.io/nlp/>

導入

深層学習と強化学習

学習

静的な深層学習

教師なし学習

データ: 入力 x 。ゴール: x に隠れた構造を学習する。
例: これはあれに近いな

教師あり学習

データ: 入力 x 、ラベル y 。
ゴール: $x \rightarrow y$ を学習する。
例: これはリンゴだ

動的な強化学習

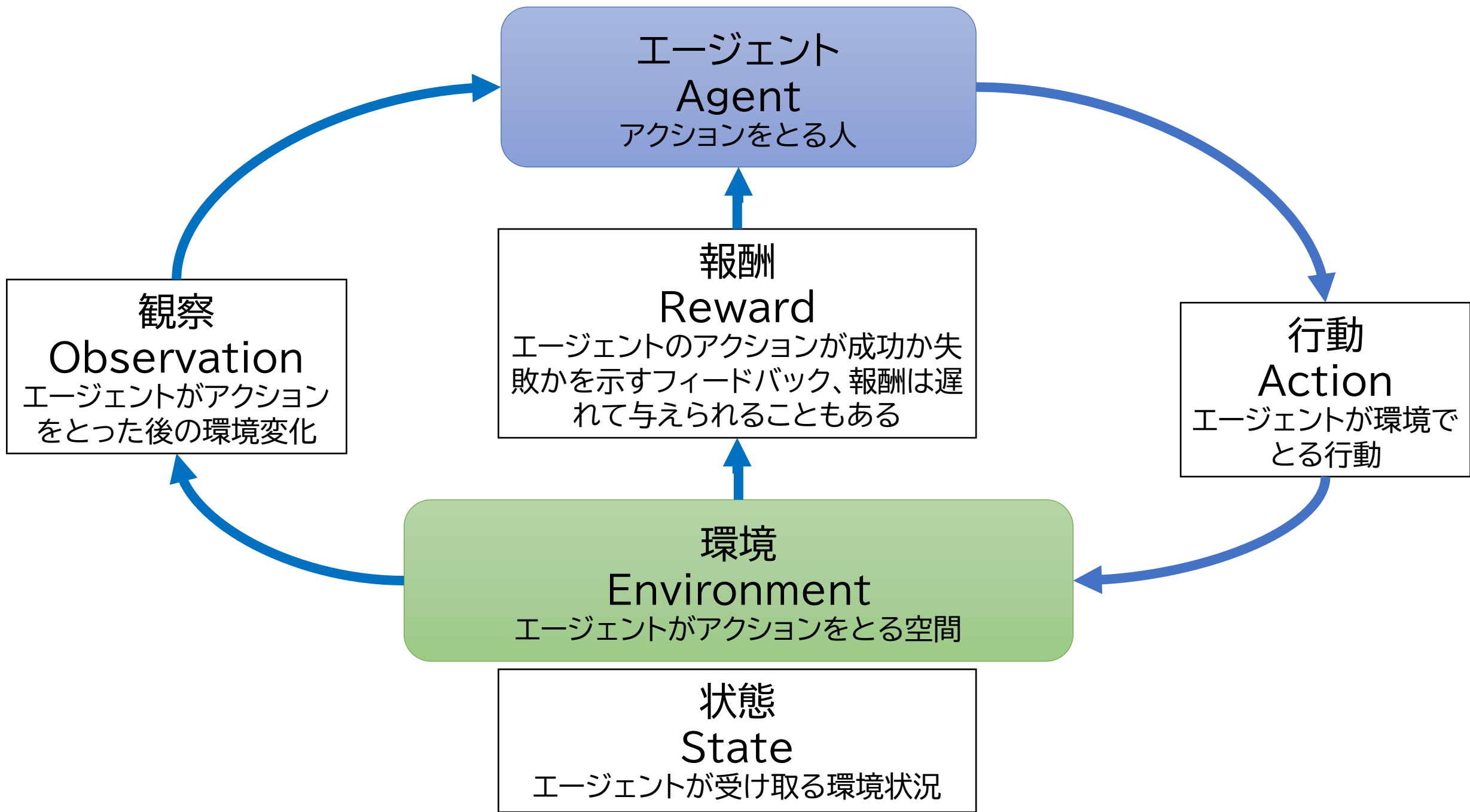
経験してみてもそのフィードバックを得る。少ない経験から学習する人に近い。

データ: 状態とアクション。
ゴール: 将来にわたっての報酬を最大化する。
例: これ食べれば生きられるから食べよう

強化学習はなぜ取りつきにくいのか？

- 国内のほとんどの解説は、銜学的に、最初から一般化した問題設定で説明を始め、長い数式が出てくるから。教育的説明ならば、単純な離散・確定的設定(\leftrightarrow 連続・確率的)から始めるべき。
- Sutton教科書の古典的な基礎アルゴリズム群の歴史をたどって、複雑なアルゴリズムを何個も体系的に把握することから始めようとするから。そうではなく、今、主流になっている方式から理解すればいい。なぜなら、最近の強化学習の急激な進歩に主に貢献したのは、古典的アルゴリズムではなく、膨大なデータを処理する技術のほうである([Andrej Karpathy blog](#))。

基礎



エピソード、終端状態

- 最終結果が出るまでの「状態、行動、報酬、…」系列、ないしその部分を、エピソードという

state(0) -> action(0) -> reward(1) -> state(1) ->
action(1) -> reward(2) -> ... -> state(n-1) -> action(n-1)
-> reward(n) -> state(n)

- 最終結果が出た状態 state(n) を終端状態という

報酬

- 累積報酬値

$$R_t = \sum r_i = r_t + r_{t+1} + \dots \quad (i=t..\infty)$$

- ディスカウントされた累積報酬値(将来の報酬を今よりより小さいとみなす)

$$R_t = \sum \gamma^i r_i \quad (0 < \gamma < 1)$$

マルコフ過程

- 強化学習は、以下のマルコフ過程を前提とする。
- 状態が、履歴から独立して、直前の状態だけで決まる場合、マルコフ的という。

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, \dots, S_t]$$

強化学習のプレイヤー

Policy/方
策

エージェント
の行動を決め
る

Value/価
値

ある状態ない
し行動がどれ
だけいいか

Model/モ
デル

エージェント
が環境を表現
する

行動価値 $Q(s,a)$, 状態価値 $V(s)$

- 行動価値 $Q(s,a)$: 状態 s でaction a をとったときに期待される
ディスカウント累積報酬値

$$\begin{aligned} Q(s_t, a_t) &= \text{期待値}[R_t \mid s_t, a_t] \\ &= \text{期待値}[\sum \gamma^i r_i] \quad (i=t..\infty, 0 < \gamma < 1) \end{aligned}$$

- 状態価値 $V(s)$: 状態 s で期待されるディスカウント累積報酬値
$$V(s) = \sum_a \pi(s,a) Q(s,a)$$

代表的なアルゴリズム

価値学習

- 状態の価値の漸化式的な関係(ベルマン方程式)を利用する
- アルゴリズム例: DQN
- ゴール: $Q(s,a)$ を見つける
- 最適ポリシー: $a = \operatorname{argmax} Q(s,a)$
- 応用例: Googleのレコメンド

ポリシー学習

- ポリシーを直接最適化する
- アルゴリズム例: PG
- ゴール: $\pi(s)$ を直接見つける
- 最適ポリシー: $\pi(s)$ は確率的で、確率に即して a を決める
- 応用例: AlphaGo

Actor-Critic

- 価値学習とポリシー学習のいいとこどりハイブリッド

ベルマン等式

- ある時刻の選択とそれ以降の決定問題の価値との関係(後ろ向き再帰)を示すことで、単純な部分問題に分割する

$$\text{状態価値 } V(s) = \max_a \{ \text{即時報酬 } r(s, a) + \gamma V(\text{新たな状態 } s') \}$$

TD(時間的差分)学習

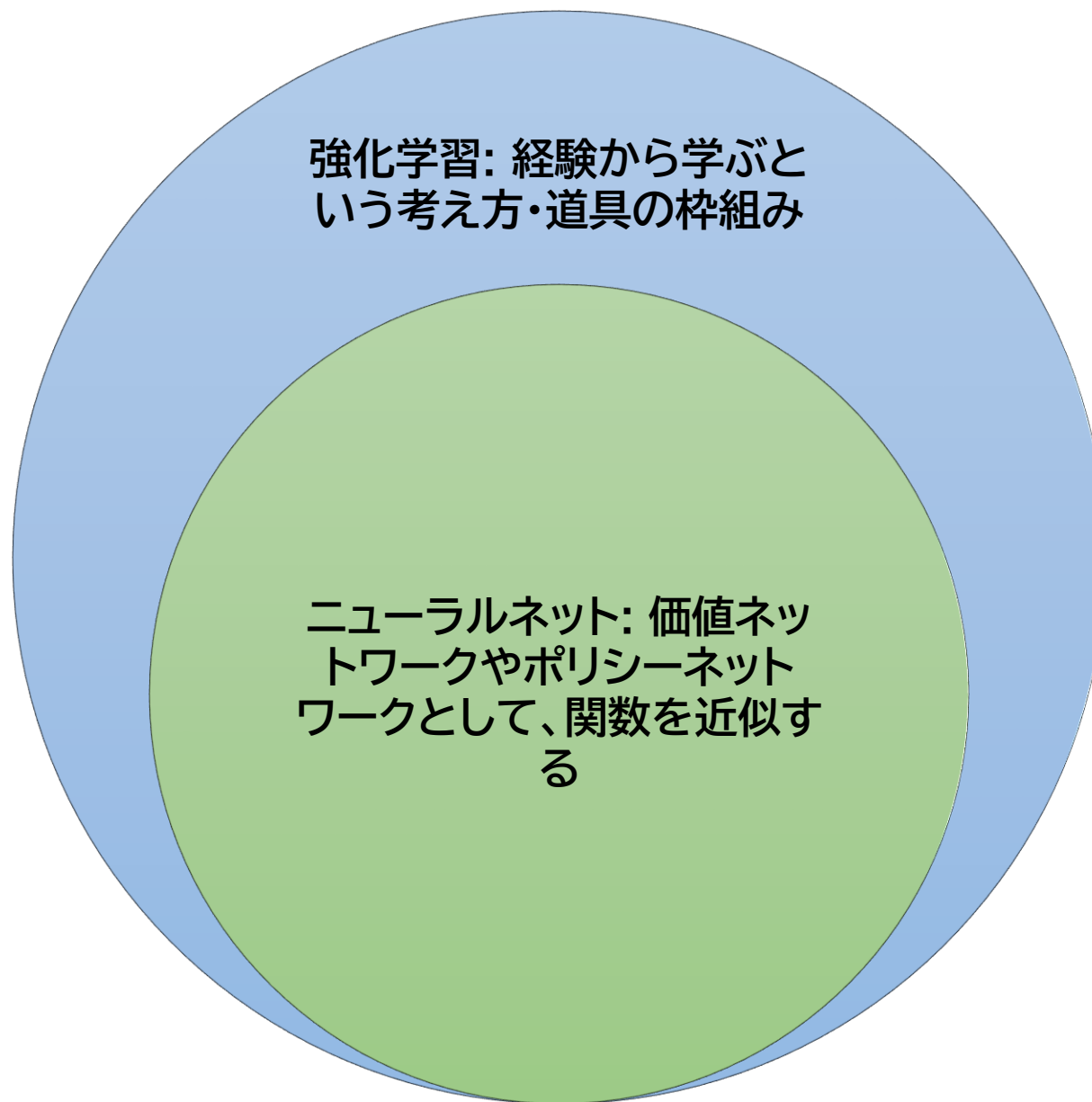
- ベルマン方程式を使い、次の状態 s' の価値と $r + \gamma V(s')$ との差を使って、学習する。

$$\begin{aligned} & \text{状態価値} V(s) \\ & \leftarrow \text{状態価値} V(s) \\ & + \alpha \{ \text{即時報酬} r + \gamma * \text{次の状態の価値} V(s') - \text{状態価値} V(s) \} \end{aligned}$$

次の時刻の推定値、より正しい

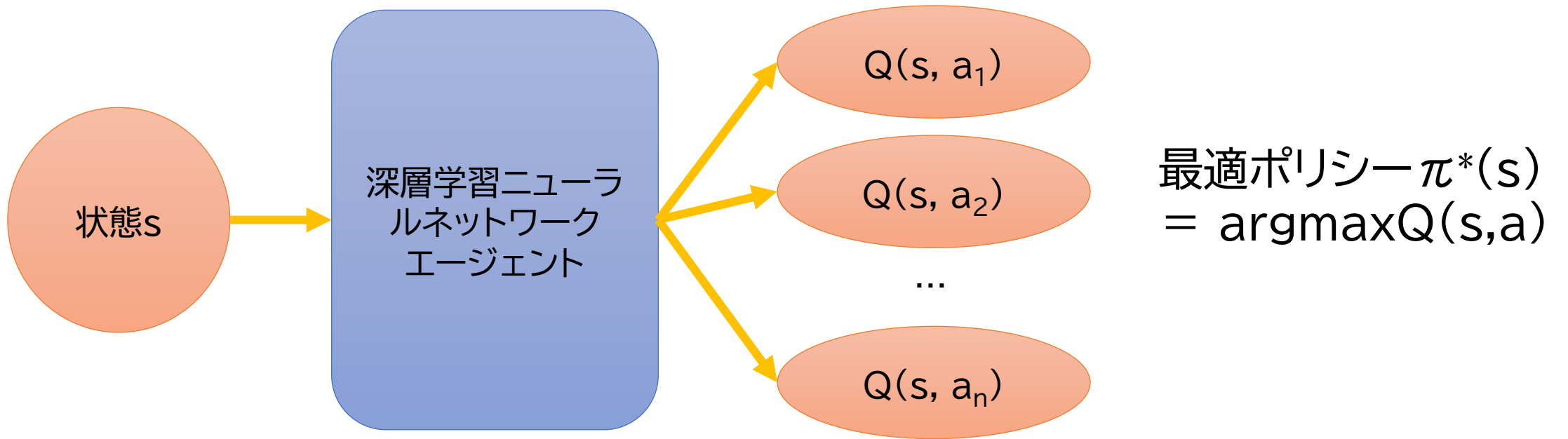
現在の推定値

強化学習は ニューラルネット による関数 近似を利用



DQN

- 課題設定: 入力 (s,a) から $Q(s,a)$ の関数を得たいところ、以下のように変更する。



DQN ネットワークのトレーニング

- 以下のQロスを使い、バックプロパゲーション学習させる

$$\text{Qロス} = || \text{target} - \text{prediction} ||^2$$

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

$$\text{prediction} = Q(s, a)$$

TD

- targetとpredictionで別々のQ表を持つのがDouble DQN。targetのQ表の更新を1000回に1回などと遅らせることで、トレーニングを安定させる。

DQN Atariゲーム

- CNN 2層、全結合 2層 だけで、ヒトより上手に球を返せるようになった

離散アクション空間

- 右左への移動など、有限個のアクションからなる空間

連続アクション空間

- 右へ3.5の力で、左へ1.2の力で、などと、実数値で表現できるアクションからなる空間。

確定的ポリシー

- アクション空間が、上下左右へのどれかへの移動、といった、有限個な世界におけるポリシー

確率的ポリシー

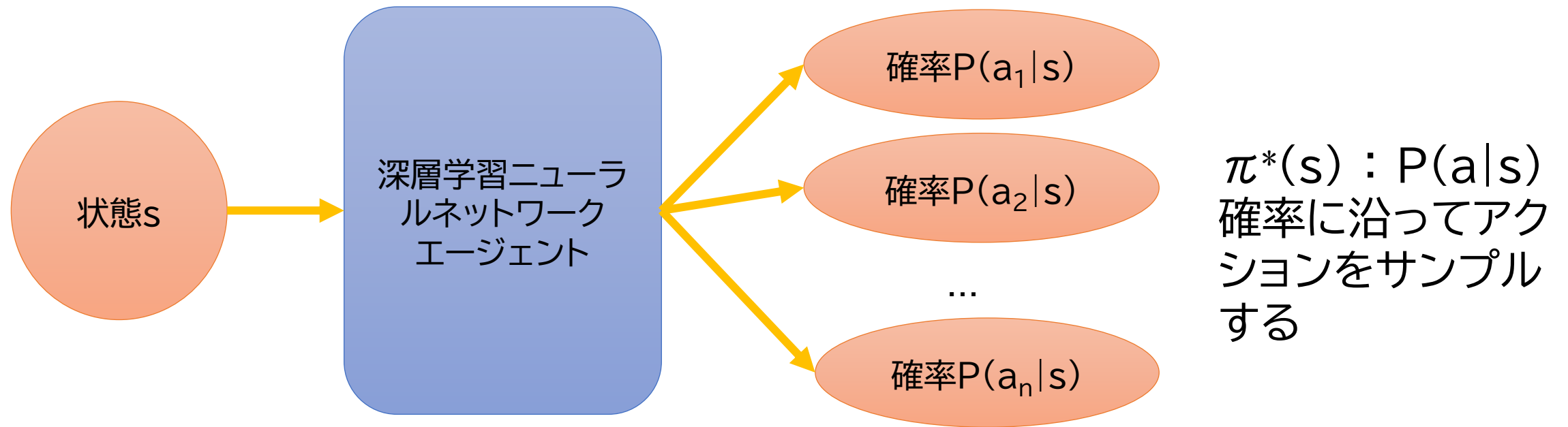
- 左へは確率0.8, 右へは確率0.05, などと、アクションを確率分布で持っておくポリシー。

- 確定的であると、最適化するうちに、ある状態 s ではいつもあるアクション a をとることになり、最適でないところに収束してしまう可能性がある。そのため、報酬を高めること (exploit) だけでなく、一定の確率でランダムにアクションを選び未知を試す (explore) ことを混ぜる。
- ある確率で未知のアクションを試すことが仕組みの中に含まれているので、exploit-exploreの仕組みはいらない。

DQNの欠点

- 離散で小さなアクション空間しか対応できない。
- DQNのポリシーは、決める確定的なもの。

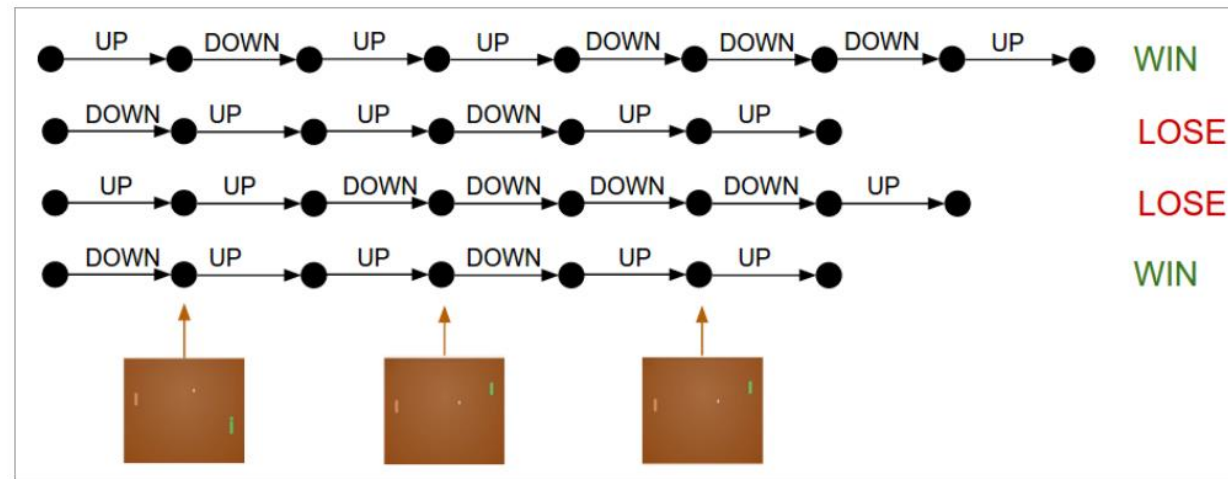
PG (REINFORCE)



$\sum P(a_i|s)=1$ 、
アクション空間は連続値でもよく、その場合 $\int P=1$

PGのトレーニング

- 初期化後、以下を繰り返す
- ポリシーを使い、終了状態まで実行する。
- 全(状態、アクション、報酬)を記録。
- ポリシー更新: 失敗したエピソードの全アクションの確率を低くし、成功したエピソードの全アクションの確率を高くする



<http://karpathy.github.io/2016/05/31/rl/>

PGのトレーニング

- 報酬が大きいほど確率を大きくし、逆は小さくするために、ロス関数は、アクション確率の対数尤度 $\log P(a_t | s_t)$ に報酬 R_t をかける。

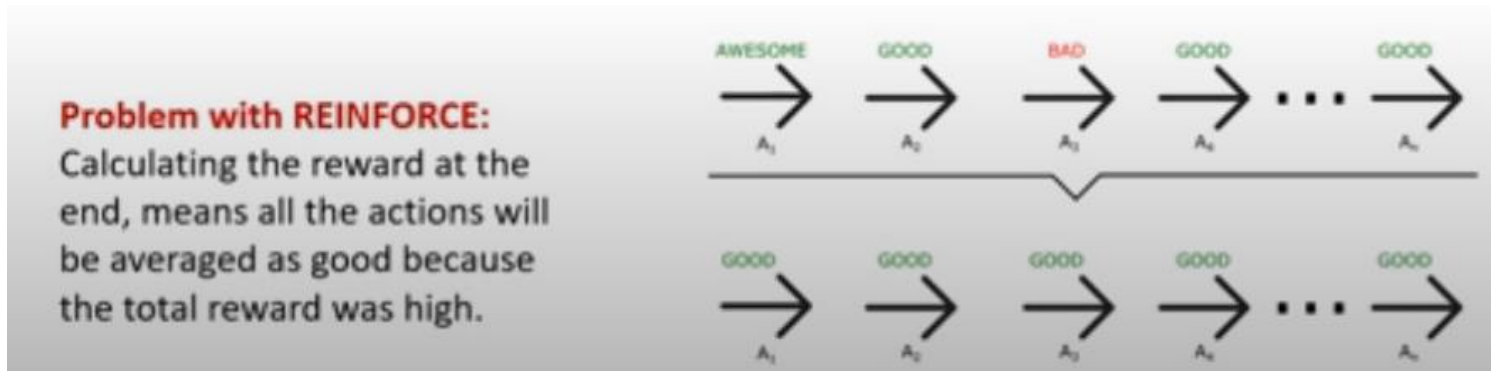
$$\text{ロス} = - \log P(a_t | s_t) R_t$$

- 勾配降下法

$$\begin{aligned} w' &= w - \nabla(\text{ロス}) \\ &= w + R_t \nabla \log P(a_t | s_t) \quad \because \text{ポリシー勾配定理} \end{aligned}$$

PGの欠点

- どのアクションがどれくらい結果に貢献したかがない。



[MIT 6.S091: Introduction to Deep Reinforcement Learning \(Deep RL\)](#)

Advantage Actor-Critic (A2C)

- 二つのニューラルネットワークを持つ
 - Actor: ポリシー学習
 - $w \leftarrow w + R \nabla \log P(a|s)$ で、確率勾配に重みを与えるRの代わりに $Q(s,a)$ を使うことで、各アクションのポリシー確率に対し、最終結果にどのくらい貢献したかの重みづけを与える
 - Critic: Q学習

実際

PGトレーニングの実際

- 自動運転

- ポリシーを起動し終了状態まで実行することは難しい(失敗は自動車の破壊)。
- シミュレーターを開発し、まずその上でトレーニングする。その後、実際のエンジンに転移学習させ、実際の環境出動かす。

RLの進歩

- ゲーム
 - AlphaGo (2016): 人のエキスパートの記録で教師あり学習。その後、エンジン自身で対戦させて強化学習。
 - AlphaZero (2018): 強化学習のみ。碁以外のゲームも。
 - MuZero (2020): ゲームの規則も学習。環境を観測、次の状態を木探索、ポリシーをプラン、アクション、これを繰り返す。

参考資料

入門的資料

- ★★★[YouTube] [A friendly introduction to deep reinforcement learning, Q-networks and policy gradients](#), by Serrano.Academy 英語
 - 内容が入門的。ベルマン方程式、価値(Q)ネットワーク、ポリシーネットワーク、ポリシー・グラディエントに関して、数式なしで、アニメーションで説明していて、基本的な概念が直観的にわかりやすい。説明は、ほぼ MIT 6.S191 (2021) の流れに同じ。
- ★★★★★[YouTube] [MIT 6.S191 \(2021\): Reinforcement Learning](#), by Alexander Amin 英語
 - Q価値学習とポリシー学習とその違いの解説
 - 数式少なく、基本的なところを説明できている。日本語のものにはない優れモノ。
- ★[YouTube] [MIT 6.S091: Introduction to Deep Reinforcement Learning \(Deep RL\)](#), by Lex Fridman, 2019 英語
 - 前半から末尾まで、広い視野からの実践的な解説。
 - 後半、DQN、PG、Actor-Criticなどのアルゴリズムの説明および関連の説明が、簡潔で明快。数式少なめで、ここまで説明できるのはすごい。

入門的資料、サンプルコード付き

- ★作りながら学ぶ強化学習 - 初歩からPyTorchによる深層強化学習まで、by 小川雄太郎
 - 第4回～第8回
 - GridWorldのサンプルコードをいじりながら、深層学習以前のSARSA、Q表学習まで。
 - Windows+Chrome+TryJupyter でなく、Google Colabのほうが楽。
 - 第9回～解説は優れているが、gymを使うサンプルコードが古く、動かない。
 - 同作者による、PyTorch Tutorial Sampleコードの改造版 PyTorch「強化学習(DQN)チュートリアル」サンプルコード も動かなくなっている。

入門的資料、サンプルコード付き、Policy Gradient

- ★★★Deep Reinforcement Learning: Pong from Pixels, by Andrej Karpathy (Tesla)
 - 英語
 - ポリシー・グラディエントに関し、わかりやすいと評判のブログ解説
 - 教師あり学習の変種という説明。
 - 方策勾配定理に関する最もわかりやすい直感的な説明。
 - YouTube版 [Deep RL Bootcamp Lecture 4B Policy Gradients Revisited](#) でコード解説
 - [pg-pong.py](#) by Andrej Karpathy のPython3版 -> https://colab.research.google.com/drive/1BTiGjP_FD0PdYfazpn61nWPC0YJuZPOE?usp=sharing
 - ROMエラーは、以下で回避
 - `%pip uninstall atari-py`
 - `%pip install atari-py==0.2.5`

入門的資料、サンプルコード付き、Deep Q-Learning

- [\[Weekly RL with code\] DQNでCartPole問題を解く](#)
 - “CartPole-v1” で2022.12.07時点で動いている
 - コード（ほぼGymの中）
<https://colab.research.google.com/gist/AGIRobots/1358b17a6169746842002783086f3282/-weekly-rl-with-code-dqn-cartpole.ipynb>
- ★[【入門】Q学習の解説とpythonでの実装 ～シンプルな迷路問題を例に～](#)
 - コード https://github.com/tocom242242/QLearning_in_GridWorld
- 以下は、動作未確認
 - https://github.com/icoxfog417/chainer_pong

入門的資料、サンプルコード付き、古典的 アルゴリズム

- ★★Sutton教科書記載の古典的なアルゴリズムを見ていくのだが、本人がもやもやしたことを納得いくように図込み、コード込みで説明しなおしたもの。とても理解しやすい。
 - [今さら聞けない強化学習\(1\):状態価値関数とBellman方程式](#)
 - [今さら聞けない強化学習\(2\):状態価値関数の実装](#)
 - [今さら聞けない強化学習\(3\):行動価値関数とBellman方程式](#)
 - [今さら聞けない強化学習\(4\):行動価値関数の実装](#)
 - [今さら聞けない強化学習\(5\):状態価値関数近似と方策評価](#)
 - [今さら聞けない強化学習\(6\):反復法による最適方策](#)
 - [今さら聞けない強化学習\(7\):モンテカルロ法で価値推定](#)
 - [今さら聞けない強化学習\(8\):モンテカルロ法でOpenAI GymのCartpoleを学習](#)
 - [今さら聞けない強化学習\(9\):TD法の導出](#) <= これ秀逸
 - [今さら聞けない強化学習\(10\): SarsaとQ学習の違い](#)
- ★[YouTube] [強化学習の概要](#) に始まる10個の動画、Tasuki Oike
 - 古典的アルゴリズム(表形式)とDQNを、簡単な具体例とコード例と一緒に解説している。Sutton系のアルゴリズムに関し、とてもいい教材。

研究者向き解説資料

- Sutton系
 - [Tutorial: Deep Reinforcement Learning](#), by David Silver, Google DeepMind
 - 英語の66ページスライド
 - 実際のシステム例の説明は高度、ある程度なじんだあとでないと難しい。それ以外、前半から半ばまでは、入門者向きで、コンパクトにわかりやすくまとめられている。
 - [Reinforcement Learning: An Introduction](#) , by Sutton & Barto
 - 英語、フリーで読める
 - この分野の古典、標準教科書
 - [Introduction to Reinforcement Learning with David Silver](#)
 - 英語
 - Suttonさんの標準教科書相当のトピックをDavid Silverさんが再構築した講義録
- [Slideshare] [ゼロから始める深層強化学習](#), by Preferred Networkの人
- [深層強化学習アルゴリズムまとめ](#)