Introduction of Reinforcement Learning

KEISUKE FUKUTA

HARADA USHIKU LAB.

Outline

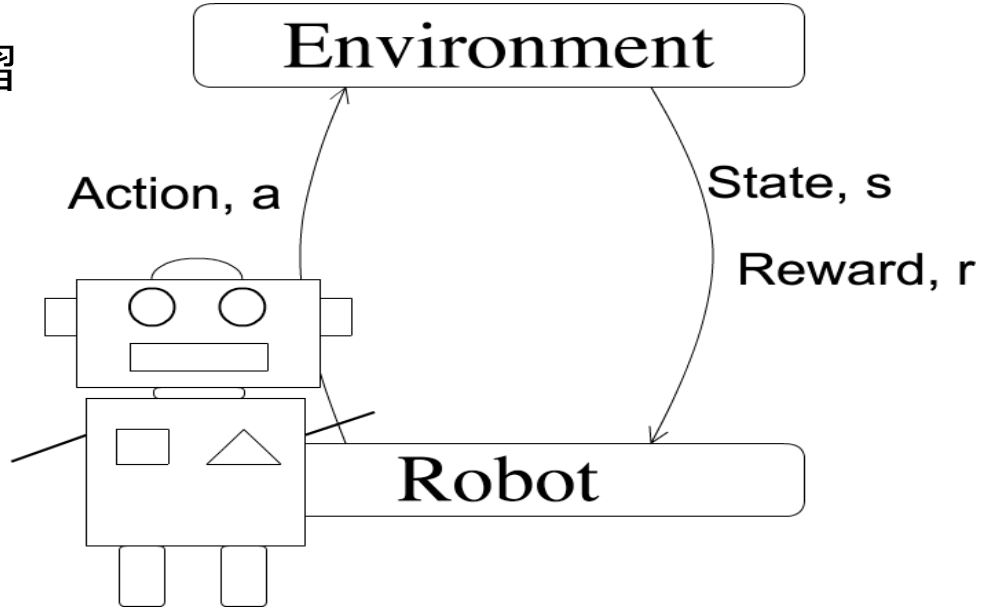
1. What is RL
2. Classical Algorithm in RL
   - TD-learning
   - Policy gradient method
3. Recent DRL application

What is Reinforcement Learning

試行錯誤を通じて未知の環境に適応する学習制御の枠組

状態入力に対する正しい行動出力を示す教師は存在しないかわりに
報酬というスカラーを受け取る

最適な戦略を報酬から間接的に学習

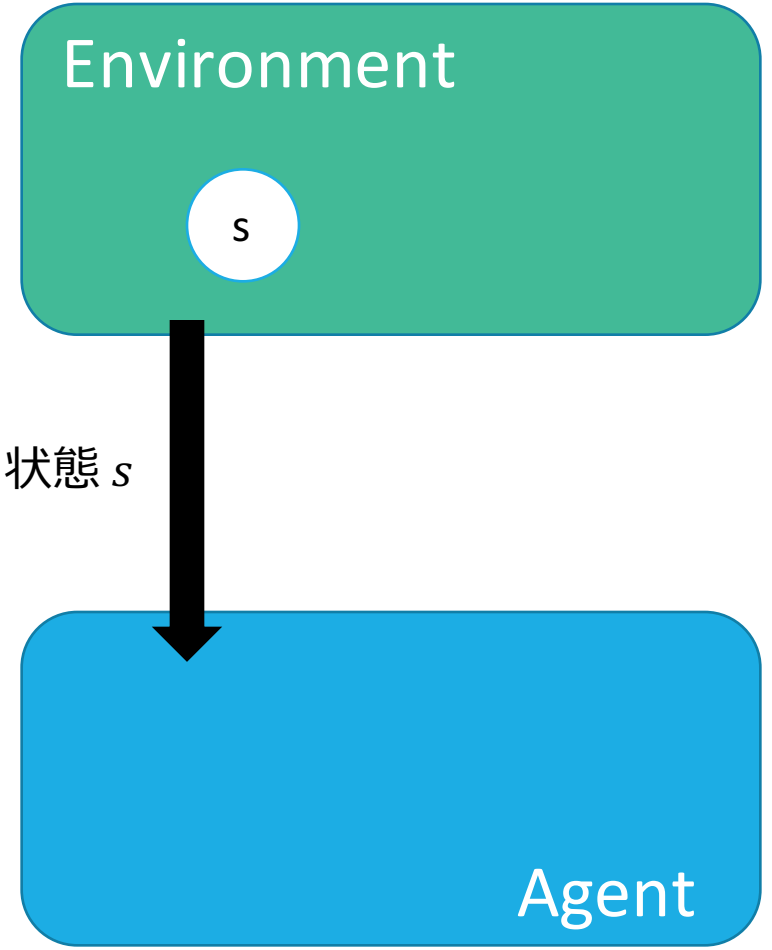


Demo

- https://www.youtube.com/watch?v=jXHnBouhAdU

What is RL

- 各Time Step $t = 0, 1, \ldots$ で
  1. Agentは状態 $s_t \in S$ を観測

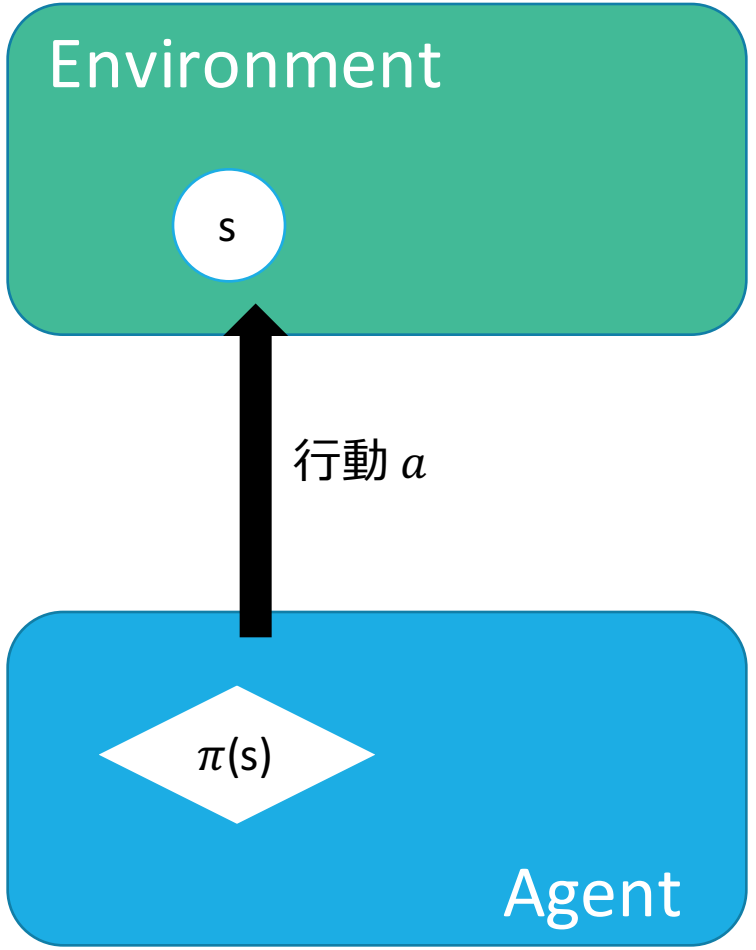


What is RL

- 各Time Step $t = 0, 1, \ldots$ で
  1. Agentは状態 $s_t \in S$ を観測
  2. 状態 s_tに応じて行動 $a_t \in A$ を選択

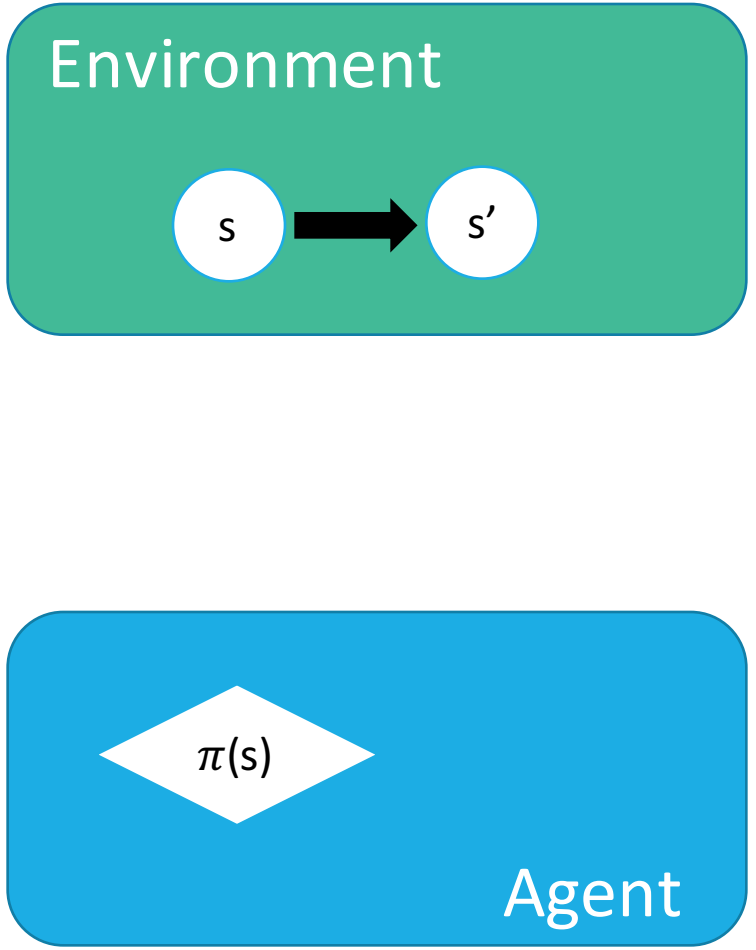
方策 $\pi(s)$

Agentの行動規則
状態から行動への写像 $\pi: S \rightarrow A$



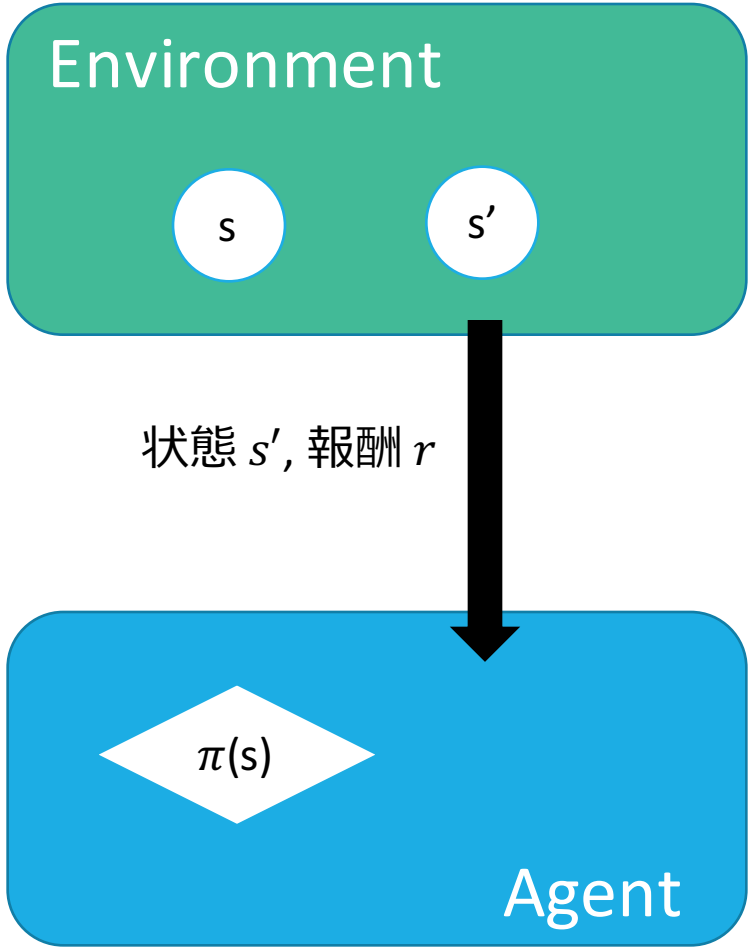
What is RL

- 各Time Step $t = 0, 1, \ldots$ で
  1. Agentは状態 $s_t \in S$ を観測
  2. $\pi(s_t)$ を利用して行動 $a_t \in A$ を選択
  3. 環境は $s_t \rightarrow s_{t+1}$ に遷移する



What is RL

- 各Time Step $t = 0, 1, \ldots$ で
  1. Agentは状態 $s_t \in S$ を観測
  2. $\pi(s_t)$ を利用して行動 $a_t \in A$ を選択
  3. 環境は $s_t \rightarrow s_{t+1}$ に遷移する
  4. Agentは環境から 報酬$r_t \in R$ と s_{t+1} を観測

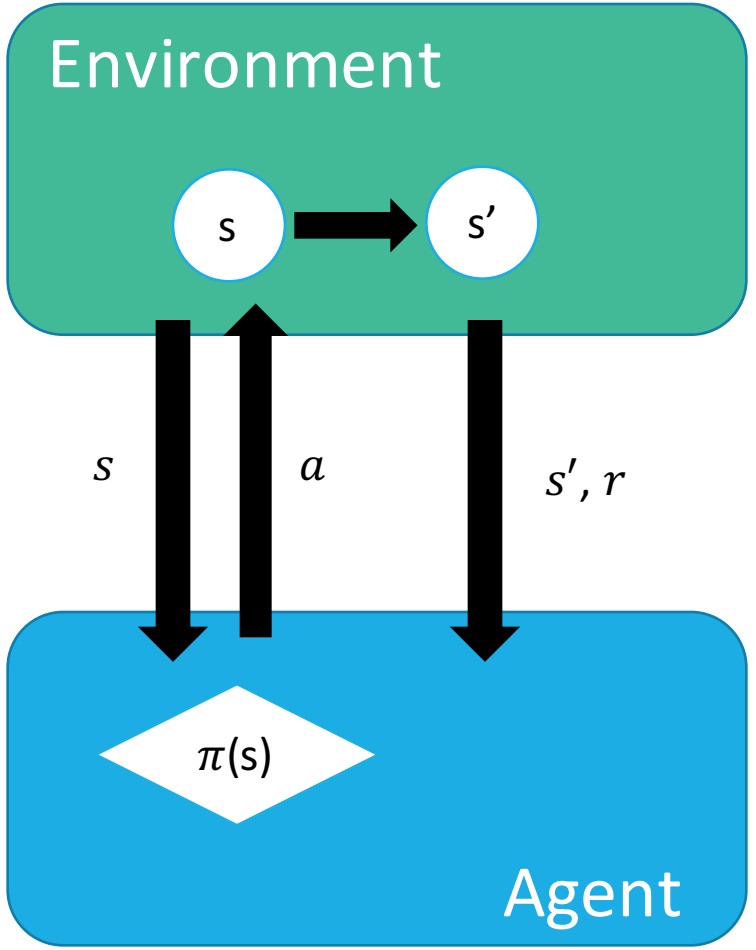


What is RL

- 各Time Step $t = 0, 1, \ldots$ で
  1. Agentは状態 $s_t \in S$ を観測
  2. $\pi(s_t)$ を利用して行動 $a_t \in A$ を選択
  3. 環境は $s_t \rightarrow s_{t+1}$ に遷移する
  4. Agentは環境から 報酬$r_t \in R$ と s_{t+1} を観測

- 目標 : 累積報酬 $\sum_t r_t$ を 最大化する方策 $\pi(s)$ の獲得

  （普通は、減衰係数 γ を導入し、$\sum_t \gamma^{t-1} r_t$とする）



Characteristic of RL

- 教師は存在せず、報酬のみで学習
- Agentは環境についての事前知識はない
  - プランニングとの違い（両方組み合わせることもある, Ex. Guided Policy Search）
- 報酬のフィードバックは遅れてやってくる
  - ある瞬間の行動が正しかったかの判断が難しい
- 時間という概念 (sequential, non i.i.d data)
- Agentのある時刻での行動が後に影響を与える
- Markov decision Process
  - マルコフ性 : $P(s_{t+1}, r_t \mid s_t, a_t, s_{t-1}, a_{t-1} \ldots s_0, a_0) = P(s_{t+1}, r_t \mid s_t, a_t)$
  - マルコフ性を満たす環境下でのAgentの意思決定をMDP (Markov decision Processes)と呼ぶ

Applications of RL

- ゲームプレイ (ex. Atari, Backgammon, Go)
  - ゲームに勝つ、スコアを獲得することが目標
- ヘリコプター制御
  - 墜落せずに飛行することが目標
- ロボットの歩行制御
  - 倒れずに進むことが目標
- 金融取引
  - リスクを小さく資金を増やすことが目標
- 電力最適化
  - 閾値を超えない範囲で効率化
- 対話システム
  - 会話が破綻させないことが目標

Example

Atari

- 目標 : ゲームのスコアの最大化

- 状態 s_t : ゲーム画面 (markov性を持たせるために4timestep分の画面)
- 行動 a_t : ボタン入力
- 報酬 r_t : スコア

Components of RL

- 方策 :
  - Agentのふるまいを決める関数

- 価値関数:
  - 各状態や行動がどれくらい良いのかを評価する関数

Value Function

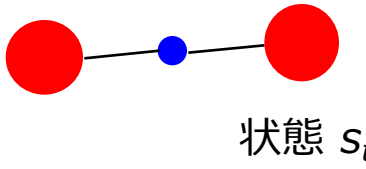
- この状態はどれくらい良いのか (囲碁、将棋でいうと盤面の評価)
- この状態でこの行動を取ると将来どうなりそうか

状態価値関数 $\mathrm{V}^{\pi}(s)$

状態 s において、そこから方策πに従って行動したとき、最終的に獲得できる累積報酬を予測する関数

$$\mathrm{V}^{\pi}(s) = \mathbb{E}_{s\sim P, r\sim R, a\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k} \middle| s_t = s\right]$$

時刻tで状態s_tにいる価値

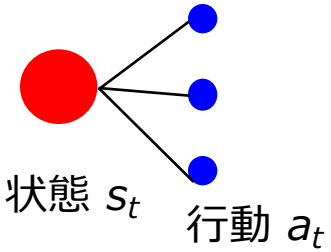


行動価値関数 $Q^{\pi}(s,a)$

状態 s において、 行動aを取り、そこから方策πに従って行動したとき、最終的に獲得できる累積報酬を予測する関数

$$\mathrm{Q}^{\pi}(s,a) = \mathbb{E}_{s\sim P, r\sim R, a\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k} \middle| s_t = s, a_t = a\right]$$

時刻tで状態s_tにいるとき行動a_tをとる価値



Classification of RL

- Model free or Model based
  - 環境の情報(ダイナミクス $P(s_{t+1}|s_t, a_t)$等)を使用するかどうか
- On-policy or Off-policy
  - 意思決定に使用される方策を直接改善するかどうか
- Value based or Policy based
  - 価値関数を学習するか、明示的に方策を学習するか

Value based or Policy based

- Value-based
  - 価値関数を学習，Implicitな方策を利用
  - Ex. TD-learning (Sarsa, Q-learning, TD(λ))
- Policy-based
  - 価値関数を学習せず, 方策を直接学習する
  - Ex. REINFORCE, Guided policy search
- Actor-Critic
  - 価値関数を学習し、それをもとに方策も学習
  - Ex. DDPG,

TD Learning (Temporal-Difference Learning)

Bellman Equation

累積報酬 = 即時報酬 + その先の未来でもらえる期待報酬

$$\mathrm{V}^{\pi}(s) = \mathbb{E}_{\pi,s}[\, r_t + \gamma\, V^{\pi}(s')]$$

$$\mathrm{Q}^{\pi}(s,a) = \mathbb{E}_{\pi,s}[\, r_t + \gamma\, Q^{\pi}(s',\pi(s'))]$$

- Bellman Equationを利用して価値関数を更新

$$\mathrm{V}^{\pi}(s) \leftarrow \alpha\, \mathrm{V}^{\pi}(s) + (1-\alpha)\,\{\, r_t + \gamma\, V^{\pi}(s')\}$$

$$\mathrm{Q}^{\pi}(s,a) \leftarrow \alpha \mathrm{Q}^{\pi}(s,a) + (1-\alpha)\{r_t + \gamma\, Q^{\pi}(s',\pi(s'))\}$$

- エピソードの終了を待たずに即時更新できるのが強い
- 脳はこんなようなことをやっているらしい

Q-learning

最も基本的な強化学習アルゴリズムの一つ (model-free, off-policy, value-base)

- 行動価値関数を最適化
- Greedyな方策$\pi(s) = \arg\max_a Q(s,a)$を利用
- 更新式

$$\mathrm{Q}^{\pi}(s,a) \leftarrow \alpha \mathrm{Q}^{\pi}(s,a) + (1-\alpha)\{r_t + \gamma \max_{a'} Q(s',a')\}$$

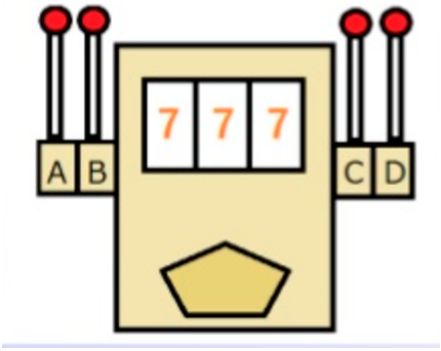
- 常にQ値の高い行動を選択するというpolicyを採用することで学習が効率化

→ 常に高いのしか選ばなかったらまだ選んだこと無いけど良い行動が学習されないのでは？

Exploitation-Exploration Trade-off

Multi-armed bandit

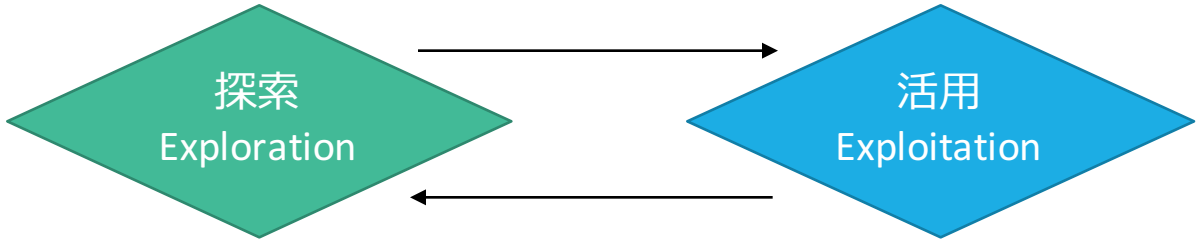
- 腕が数本あるスロットマシン
- それぞれの腕毎に出てくる賞金額とその確率は違う
- 限られた回数で多くの賞金がほしい



バカ：1回目で腕Aで1000円当たった → 腕Aしか回さない

賢者：腕Aは確かに悪くない。けど他にもいいのがあるかもしれないから試してみたい

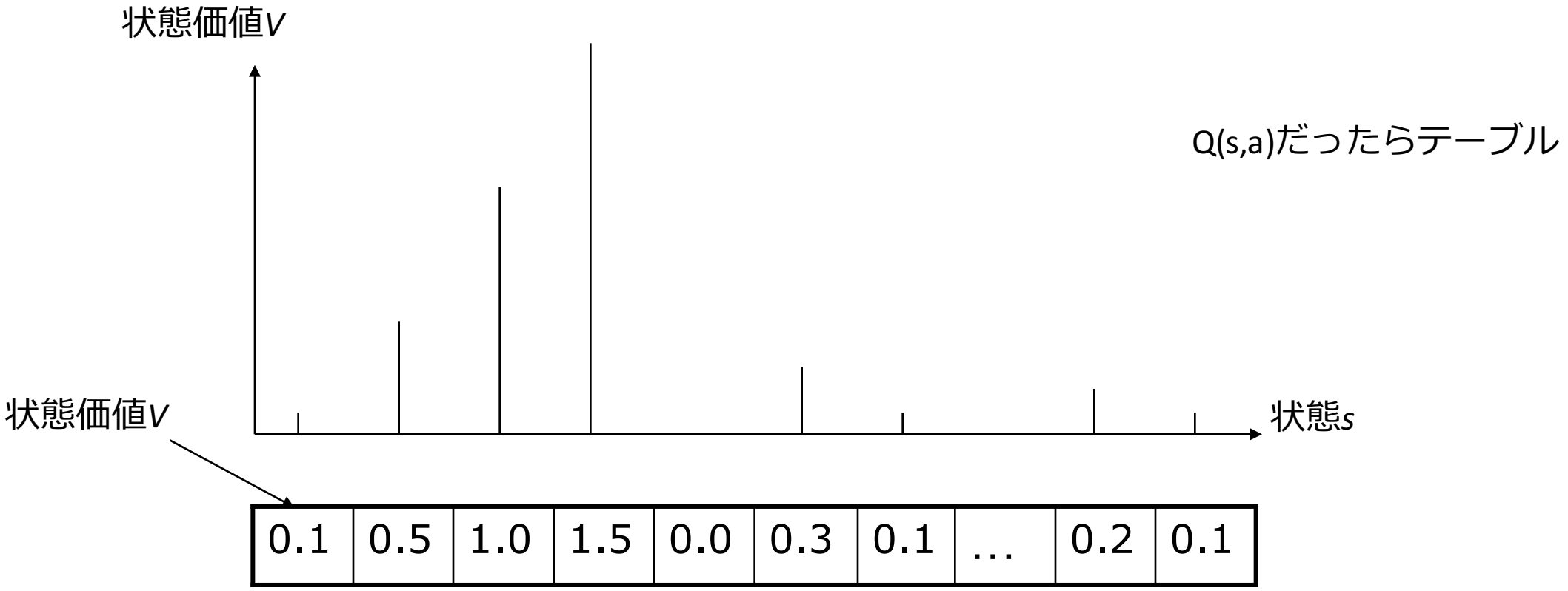
→ Exploitation-Exploration Trade off



Q-learning

- Greedy法
  - $\pi(s) = \arg\max_a Q(s,a)$
- ϵ - Greedy法
  - 確率 ϵ で ランダムな a : Exploration
  - 確率 1 - ϵ で $\arg\max_a Q(s,a)$: Exploitation
  - 多くの場合、学習と共にϵを小さくしていく（バランスを気をつけないと、局所解に）
- Softmax行動基準
  - $X_i = Q(s,a_i)$とすると、$p(a_i) = \frac{e^{X_i/T}}{\sum_i e^{X_i/T}}$
  - 探索においても、価値の比によって選ばれ方が変わる

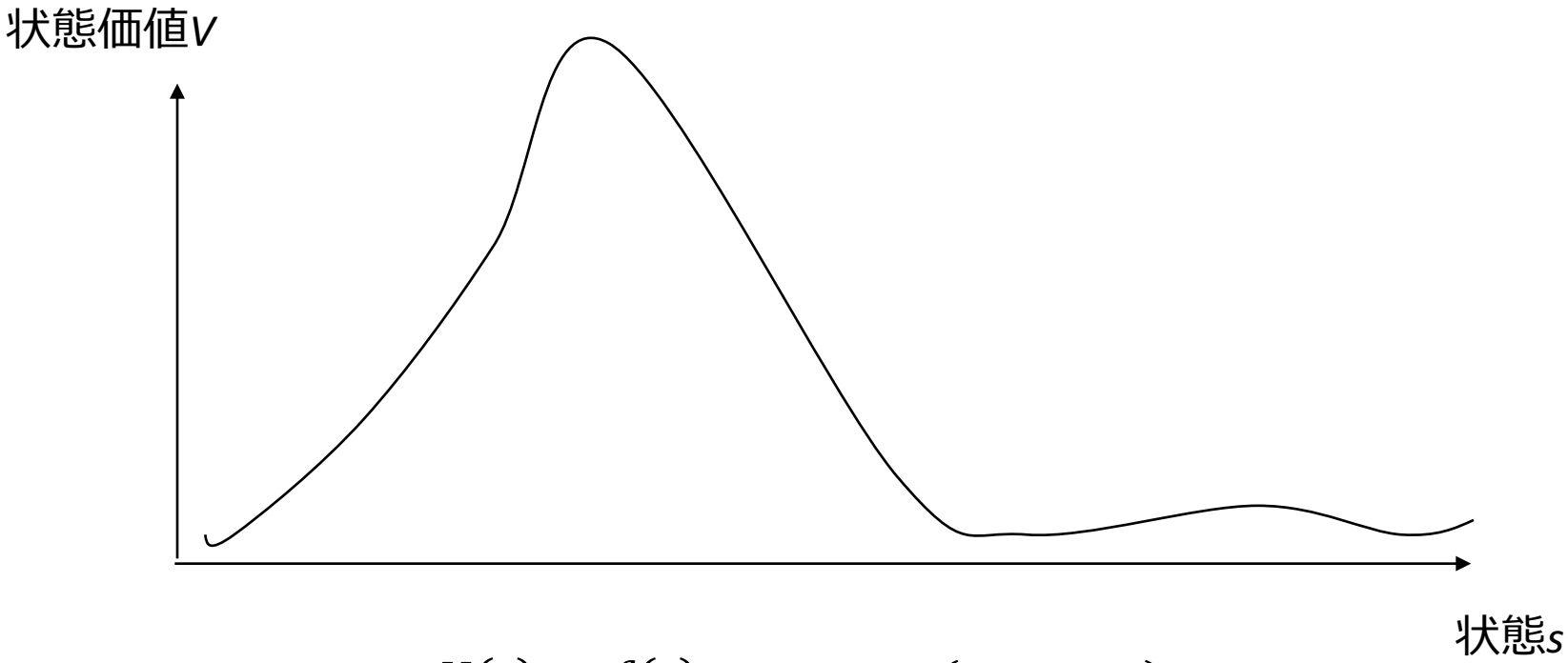
Representation of Value Function



Q(s,a)だったらテーブル

迷路とかの単純な状態空間なら良いが、状態空間が爆発するとき困る

Function Approximation of Value Function



$$V(s) = f(s)\ , \qquad s = (x_0, x_1, \dots)$$

関数 f には、線形モデルだったり、一般化線形モデルだったりが選ばれる

DQN

- Q - learningにおける行動価値関数の関数近似においてCNNを使用
- 価値関数のNNによる近似自体は昔からある
  - TD – Gammon [Tesauro, 1994]

⇒ じゃあ何が新しいのか

- Q-learningは、関数近似に線形なモデルを利用し、適切な学習率を設定し十分な探索が行われた場合収束することが理論的に保証されている

→ CNNのような複雑な非線形モデルの場合収束が保証されない

→ 頑張って収束させた!! (そのためのいろんな工夫)

Experience Replay

強化学習 × DNN の難しさ

- DNNを学習させる際、データサンプルはi.i.d.である必要

→ But! 強化学習では、

  - 似たような状態入力が続く (入力に相関)
  - ゲームの進行状況によるサンプルのdistributionの変化 → 学習が不安定
    - 一度覚えたことを忘れてしまう

- そもそも学習に大量サンプル必要
  - 環境とのインタラクションを何度もしなければならない → サンプルを集めるのに非常に時間がかかる

→ どこか(Replay buffer)に経験をサンプル保存しておいて、
更新するときはそこから確率的に取り出して更新しよう！！

Off-policyにのみ適用可能

Value based or Policy based

- Value-based
  - 価値関数を学習，Implicitな方策を利用
  - Ex. TD-learning (Sarsa, Q-learning, TD(λ))
- Policy-based
  - 価値関数を学習せず, 方策を直接学習する
  - Ex. REINFORCE, Guided policy search
- Actor-Critic
  - 価値関数を学習し、それをもとに方策も学習
  - Ex. DDPG,

Explicit Policy

- Value-based
  - Implicitな方策 ex. Q-learning $\pi(s) = \arg\max_a Q(s,a)$

→ 行動a の取りうる選択肢が非常に多かったら？

→ 確率的な方策を表現したかったら？

→ ロボットの制御みたいに連続値を扱いたかったら？

$\pi(s)$を明示的に導入

Policy gradient method
Guided policy search
.. Etc.

Explicit Policy

- Advantages:
  - 収束しやすい
  - 連続空間、高次元空間に適用可能
  - デモンストレーションから学習することも可能
  - 探索を直接的にコントロールできる
  - 確率的な方策も学習可能
- Disadvantages:
  - 局所解に陥りやすい
  - 方策の評価の効率が悪いことが多い
- Policyの学習方法は大量にある (policy search)

Policy Search

Model-Free Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]}, r_{1:T}^{[i]} \right) \right\}$$

to directly update the policy

Optimization methods:

- Policy Gradients [Williams et al. 992, Peters & Schaal 2006, Rückstiess et al 2008]
- Natural Gradients [Peters & Schaal 2006, Peters & Schaal 2008, Su, Wiestra & Peters 2009]
- Expectation Maximization [Kober & Peters 2008, Vlassis & Toussaint 2009]
- Information-Theoretic Policy Search [Daniel, Neumann & Peters 2012, Daniel, Neumann & Peters, 2013]
- Path Integral Control [Theoudorou, Buchli & Schaal 2010, Stulp & Sigaud 2012]
- Stochastic Search Methods [Hansen 2012, Mannor 2004]

Model-Based Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]} \right) \right\}$$

to estimate a model

Optimization methods:

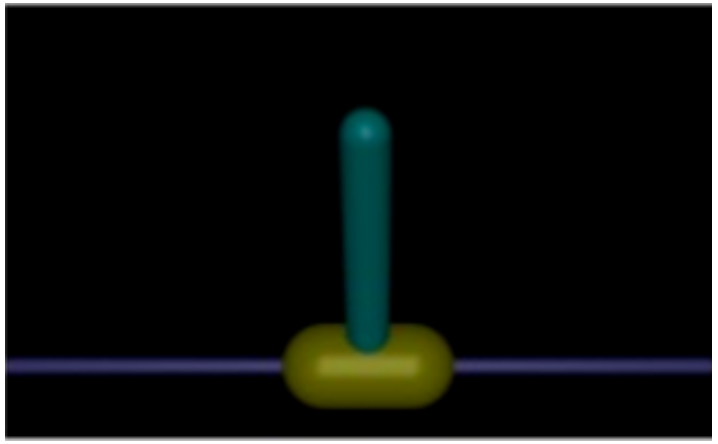
- Any model-free method with artificial samples [Kupscik , Deisenroth, Peters & Neumann, 2013]
- Analytic Policy Gradients [Deisenroth & Rasmussen 2011]
- Trajectory Optimization [Levine & Koltun 2014]

Example

Cart pole balancing (Mujoco simulator)

- 目標 : 棒を垂直に維持する

- 状態 : ゲーム画面、もしくは速度、回転角等の物理量
- 行動 : 入力トルク
- 報酬 :$\cos(\theta)$ (θ : 棒の角度)



Policy Gradient Method

これまた古典的なアルゴリズムだがよく使用される

- 目的関数

$$J(\pi_\theta) = \int_S \rho^\pi \int_A \pi(s,a) r(s,a)\, da\, ds = \mathbb{E}_{s\sim\rho, a\sim\pi_\theta}[r(s,a)]$$

- $J(\pi_\theta)$を最大にするような方策π_θを求めたい

→　そのために、$\nabla_\theta J(\pi_\theta)$を求めて$\pi_\theta$を更新したい

Policy gradient

- $\nabla_\theta J(\pi_\theta)$を求める方法として
  - Likelihood ratio method
  - Value gradient

Likelihood Ratio Method

目的関数 $$J(\pi_\theta) = \int_S \rho^\pi \int_A \pi(a|s)\, r(s,a)\, da\, ds$$

$\pi(a|s)$は確率的な方策

求めたい policy gradient

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi \int_A \nabla_\theta \pi(a|s)\, r(s,a)\, da\, ds$$

$$= \mathbb{E}_{s\sim\rho, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\, r(s,a)\right]$$

$$\cong \frac{1}{M}\left[\nabla_\theta \log \pi_\theta(a|s)\, r(s,a)\right]$$

$$\nabla_\theta \pi(a|s) = \pi(a|s)\frac{\nabla_\theta \pi(a|s)}{\pi(a|s)}$$

$$= \pi(a|s)\, \nabla_\theta \log \pi_\theta(a|s)$$

Score function
と呼ぶらしい

- サンプリングによって勾配を推定できる！
- パラメータ化された確率分布を微分したいがReparametrization trickが使えないとき最近よく見る
  - Hard Attentionとか

Likelihood Ratio Method

- 結局 : $\nabla_\theta J(\pi_\theta) \cong \frac{1}{M}\left[\nabla_\theta \log \pi_\theta(a|s)\, r(s,a)\right]$
- $r(s,a)$
  - 実際に得られた報酬を利用
    - REINFORCEと呼ばれる
    - 真の報酬であるのでUnbiasだが、サンプリングなので勾配推定の分散が大きい
  - 行動価値関数 $Q^\pi\,(s,a)$による近似を利用
    - Varianceは下がるが近似なので当然biasが大きい。うまいことやる必要
  - 両方使う
    - Control Variate
      - 期待値を求めたいものから解析的に計算できるbaselineを引いて分散を小さくする
    - Baselineに状態価値関数を利用し、Advantage functionを
      - Advantage function $A^\pi(s,a) = \ Q^\pi\ (s,a) - \ V^\pi\ (s) \cong \mathrm{r} + \gamma\, V^\pi(s') - \mathrm{V}^\pi(s)$

Value Gradient

- 行動価値関数 $Q^{\pi}(s,a)$ の勾配を直接使って更新したい
  - $Q^{\pi}(s,a)$が大きくなる方向に a を動かそう
  - Experience Replayが使える！

- Deterministic policy gradient [Silver et al., ICML, 2015]
  - 決定的な方策 $\mu_{\theta}(s)$ を導入し、$Q^{\mu}(s,\mu_{\theta}(s))$ を直接 θ で偏微分することで勾配を推定する
  - $\nabla_{\theta}Q^{\mu}(s,\mu_{\theta}(s)) = \nabla_{a}Q^{\mu}(s,a)\cdot\nabla_{\theta}\mu_{\theta}$
  - DDPG (Deep deterministic policy gradient) [Lillicrap et al., ICLR, 2016]は Q, μをCNNでモデル化したもの
- Learning Continuous Control Policies by Stochastic Value Gradients
  - Reparametrization TrickによりStochastic policy にも適用可能に

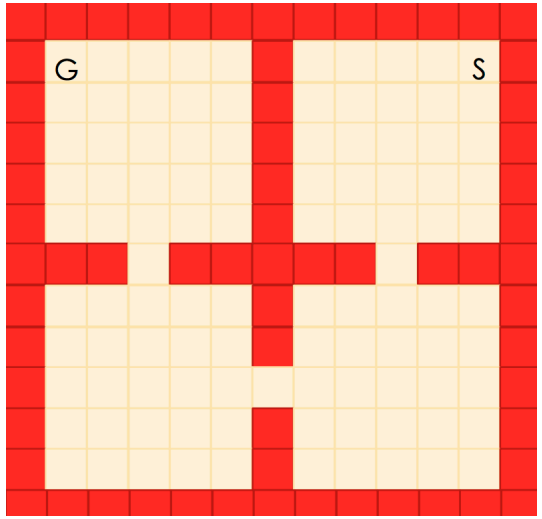
Policy Gradient Method ~summary~

- Likelihood ratio
  - Bias 小・Variance大
  - On-policy
  - 学習はある程度安定するが、大量のデータサンプルが必要

- Value gradient
  - Bias 大・Variance 小
  - Off-policy
  - Experience Replay が使えるためサンプル efficient だが、ハイパラ調整が闇

Design of Reward

- ゴールまで行きなさいという問題
  - 例 1）ゴールに近づいたら (+1), ゴールで高い報酬 (+100)
    → たどり着くまでかなり時間がかかる
  - 例 2) ポテンシャルに応じた報酬設定
  - → 高速に学習は進むが、ハイパラ増える、設計大変

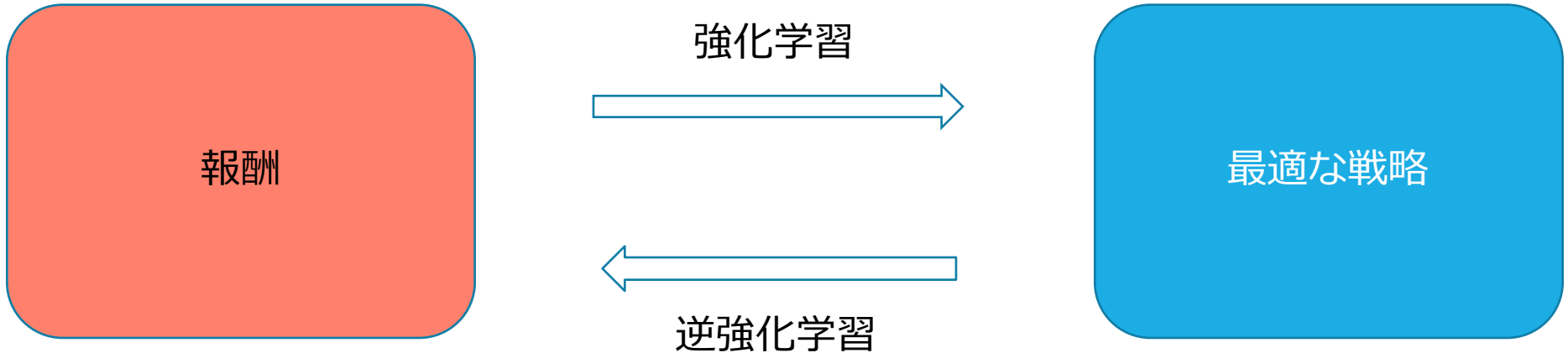


⇒ 報酬の設計に関する研究

- 高速に探索が進むような報酬の生成方法の研究
  - Reward shaping
  - 徒弟学習
  - 逆強化学習

Inverse Reinforcement Learning

- 強化学習では、与えられる報酬＝良さを元に最適な戦略を推定する
- 逆強化学習では、最適な戦略から報酬＝良さを推定する



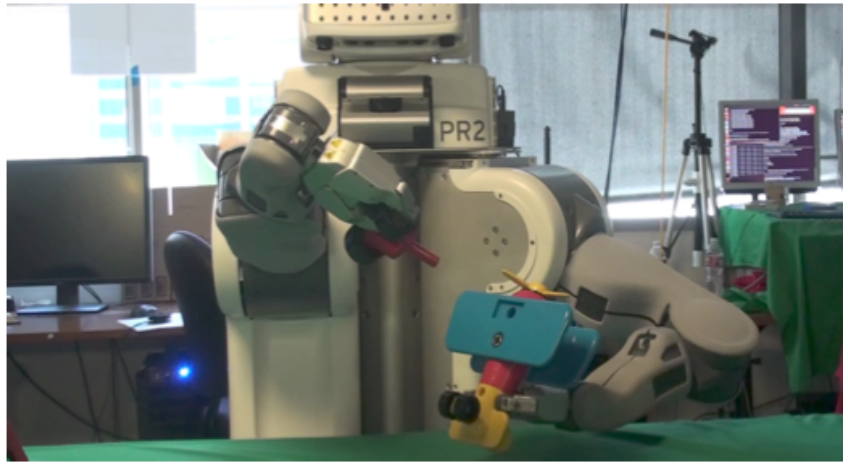
Recent DRL ①

Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection
[Sergey Levine et al., 2016]

- Googleがロボットアーム14台並列に動かして物体のグラスピングを学習させてたやつ
- 概要
  1. 様々な状況でモーターを動かしてみて、成功したか失敗したかのサンプルを保存
  2. 保存したサンプルを利用して、Prediction Network $g(I_t, v_t)$ { I_t: 視覚情報, v_t: サーボへの命令 }を supervised-learning
  3. サーボへの命令は、サンプリングベースで$g(I_t, v_t)$が高くなるのを選ぶ
- 強化学習とよく言われるが、 self-supervised learningと呼ばれる自分でデータサンプルを集めて学習する枠組み

Recent DRL ②

End-to-End Training of Deep Visuomotor Policies
[Sergey Levine, 2015]



- UCバークレーのロボットが最初はランダムだが、だんだん鍵穴に鍵させるようになるやつ
- Guided Policy Search, iLQG
  - Policy based
  - TD学習とかとは全く異なるアプローチ
  - 近傍のダイナミクスを線形近似し、ローカルに最適解を解析的に解く、制御理論という感じ
  - 強化学習自体広い意味を持つのでこれも含まれる

Recent DRL ③

Asynchronous Methods for Deep Reinforcement Learning
[Mnih et al., 2016]

- 非同期に多数のエージェントを走らせてパラメータを同時に更新することでサンプル数を確保すると同時に入力の相関をなくすことができる

→ Experience Replayを使う必要がない

→ on-policyなRLアルゴリズムが使用可能!!

- Advantage functionを用いたActor-Criticを非同期で走らせた結果、CPUで１日たった時点で他手法を大きく上回る
  (A3C : Asynchronous Advantage Actor Critic)

Recent DRL ④

Continuous Deep Q-Learning with Model-based Acceleration (NAF)

- Q-learningを連続空間に適用可能に
  - Advantage部分とState-Valueに分け、Aの方を二次形式の形で推定することで$\arg\max_u Q(s,u)$ を計算

    Normalized Advantage Function

$$Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q) = A(\boldsymbol{x},\boldsymbol{u}|\theta^A) + V(\boldsymbol{x}|\theta^V)$$

$$A(\boldsymbol{x},\boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))^T \boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))$$

- iLQGのように近傍のダイナミクスを推定することで
  Model-basedの手法を取り入れたい
  - Imagination rollout

Recent DRL ⑤

Trust Region Policy Optimization (TRPO)

- Policy based
- $D_{KL}(\theta_{old}, \theta) \leq \delta$ (Trust Region)という範囲において、サンプリングによって計算された期待コストを最小化するθを制約付き最適化問題を解くことで求め逐次的に更新

- サンプリング法
  - Single Path : s_0からπ_θに従い軌跡を生成する方法
  - Vine: ランダムシミュレーションにより生成する方法

- 学習の流れ
  - Single pathかVineでサンプル（軌跡と報酬）を集める
  - サンプルから制約条件と目的関数を構築
  - 制約付き最適化問題を解く
    - 中身はよくわからなかったです

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right] \qquad (14)$$

$$\text{subject to}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\,\|\,\pi_\theta(\cdot|s))\right] \leq \delta.$$

Recent DRL ⑥

Q-prop

- Actor-Critic
- Off-policyに学習するcriticをbaselineとして利用して、
  On-policyなpolicy gradientをsample efficient かつunbiasに利用する方法

おまけ

Connecting Generative Adversarial Networks and Actor-Critic Methods [DeepMind, 2016]

- GANとActor-Criticの関係
  - GANは、Actorが報酬に影響を与えない状況下でのActor-Criticと同じなのでは？

- それぞれで使用される学習テクニックを比較、お互いに利用できないか
  - GAN
    - Freezing learning, Label smoothing, Historical averaging, Minibatch discrimination, BN
  - Actor-Critic
    - Experience Replay, Target networks, Entropy regularization, Compatibility?

Reference

- David silverの講義資料
  - http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf
- Policy search資料
  - http://icml.cc/2015/tutorials//PolicySearch.pdf

Recent Technique

- Prioritized Experience Replay
  - 重要な経験サンプル(TD-errorが大きかったもの)を優先的にサンプルするExperience Replay
- Deep Reinforcement Learning with Double Q-Learning
  - DQNのDouble Q-learning版

Applications of RL

- ヘリコプター制御 http://heli.stanford.edu/
  - 人間のエキスパートの技を学習させるために、逆強化学習 (Inverse Reinforcement Learning）を利用
- Windows ヘルプ文書理解 http://groups.csail.mit.edu/rbg/code/rl/
  - Windowsのヘルプに出てくる操作手順の自然言語に対応する操作を学習します。従来であれば教師付き学習の枠組で研究されていた問題ですが「指示通りの操作が不可能」な場合に対して負の報酬を割り当て、どこまで指示に従った操作ができるかを強化学習によって学習することで、
    教師データがない、あるいは少量しかない場合でも、精度の高い学習ができることを示しました。
- 音声対話システム http://mi.eng.cam.ac.uk/research/dialogue/
  - Cambridge大の Young らが開発した音声対話システムでは、ユーザーがシステムにしてほしい要求を隠れた状態と考え、ユーザーの発話内容を観測することで要求を明らかして対応する POMDP 問題と考えることで、最も適切な応答を学習します。ノイズによって発話内容が正しく観測できない場合でも、従来のシステムに比べはるかに良い応答が実現できることが示されており、今後の実用化が期待されます。
- 構文解析 http://www.umiacs.umd.edu/~hal/SPIRL/10-07-acl-spirl.pdf
  - 構文解析を高速化するためには、すべての可能性を探索するのではなく、より文らしくなるような候補を先に探索することが有効です。
    解析途中にどの選択肢を選ぶかを行動と考え、逆強化学習の枠組で解くことで、この高速化が達成できることを示しました。
    同様のテクニックは、自然言語処理に限らず、探索問題全般に適用可能であり、面白いアプローチとして注目を集めています。
- 滞納債務の取り立て(論文 http://www.prem-melville.com/publications/constrained-reinforcement-learning-kdd2010.pdf、
  講演動画 http://videolectures.net/kdd2010_abe_odcucrl/)