Dreamer方式のPolicy更新手法

GesonAnko

自己紹介

- げそん (GesonAnko)

  X(Twitter)@GesonAnkoVR

- ML集会（水曜 21:30～） 主催

  AI に脳みそを焼かれた人間

  PythonでML関係ツールの作成

  VRChatに P-AMI<Q> っていう自律機械知能を作ったよ。

目次

発表目的

- VConf24で用いる行動生成器(Policy)の更新手法について、チーム内で共通の理解を得ること。

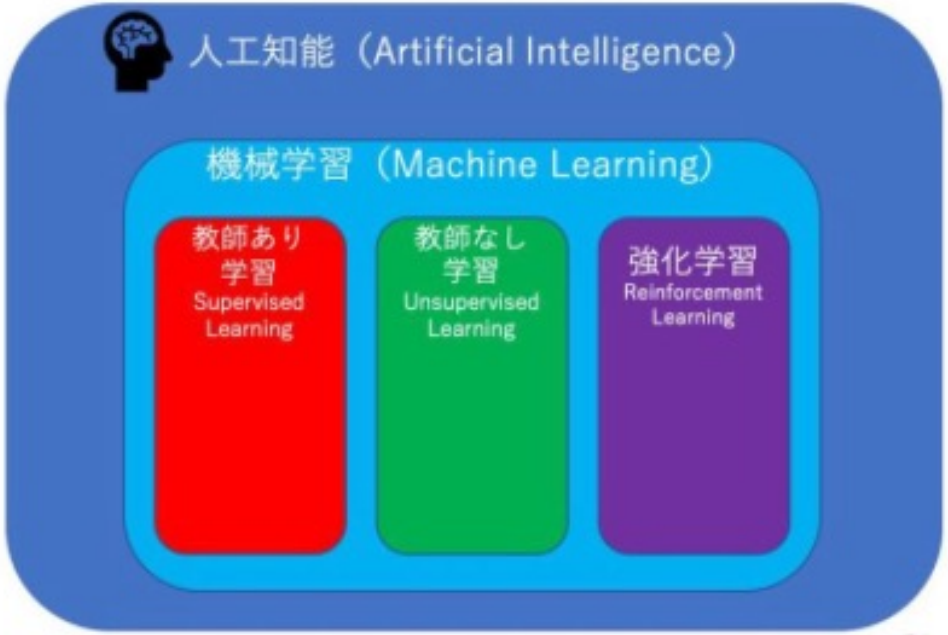
- それにより、スムーズな実装および振舞いを分析のための的確な実験手法を考案・実施できるようにするため。

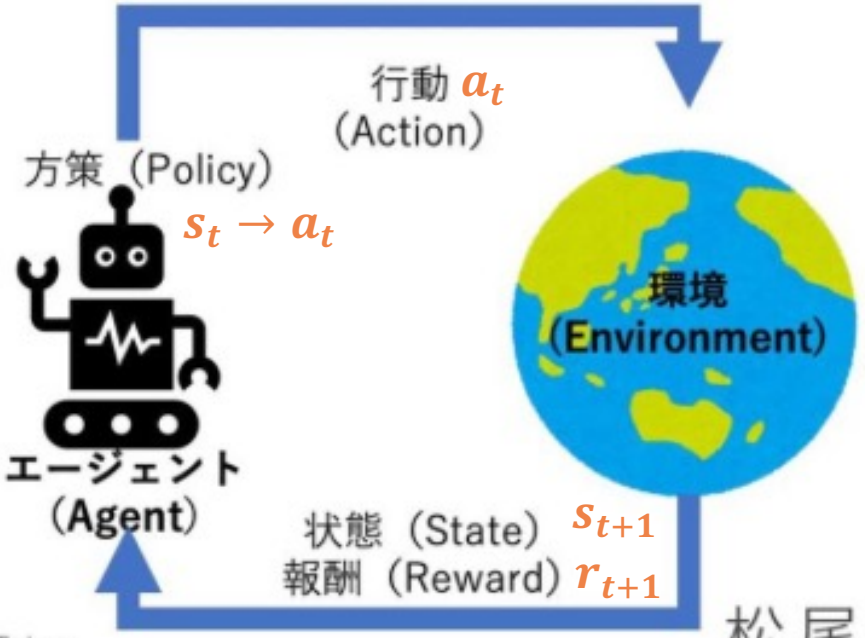
強化学習について

強化学習

行動生成器

- AIの部分技術である機械学習の中で，**意思決定則（方策）を学習する手法**
- エージェントが，ある環境（タスク）の中で目標を達成するために
  どのように意思決定をして行動すればよいのかを定式化して学習
- エージェントは環境を探索して得た状態遷移や**報酬**のデータを元に方策を
  学習　**報酬の総和（収益）を最大化させることが目的！**

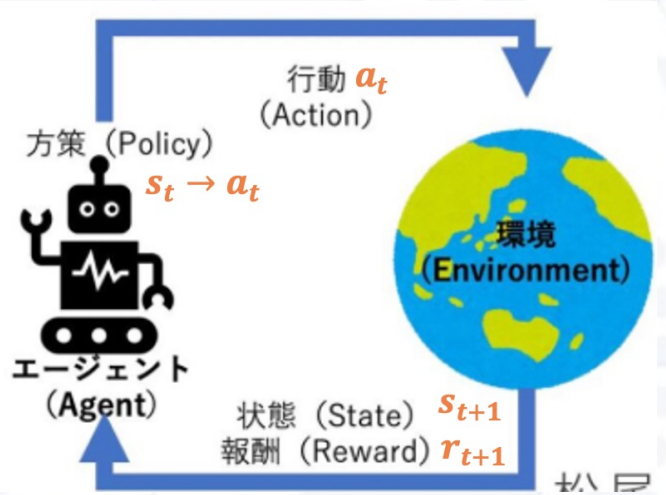




今井翔太, “誰でもわかる強化学習”より。詳しくはそちらを参照

強化学習の基本設定

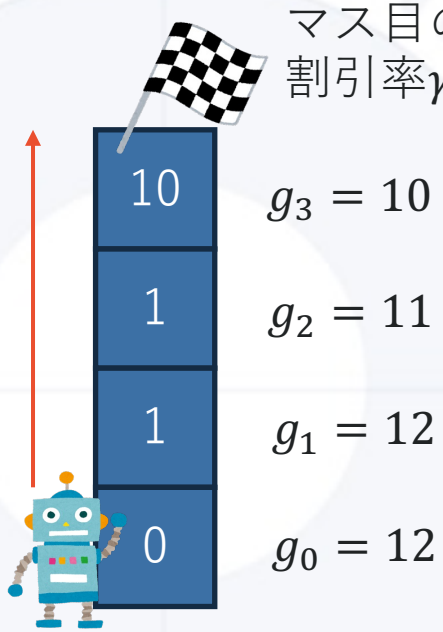
- 登場する概念
  - **状態s_t** ∈ 全状態の集合 S
  - **行動a_t** ∈ 全行動の集合A
  - **方策分布$\pi(\cdot|s_t)$**, 行動は確率的に生成される。 $a_t \sim \pi(a_t|s_t)$
  - 次の状態s_{t+1}を得られる確率（分布）$p_s(s_{t+1}|s_t, a_t)$
  - **報酬r_t** $\in \mathbb{R}$
  - 報酬r_{t+1}を得られる確率(分布）$p_r(r_{t+1}|s_t, a_t)$
  - **経験軌道$\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots\}$**



  - 即時報酬r_{t+1}ではなく、長期的な利益を最大化したい。
    - 報酬の総和（収益）を最大化しよう
    - **収益$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$**
      - **割引率$\gamma \in [0,1]$**: どれだけ未来報酬を重視するか。無限和を収束させる。

収益の推定

- 収益g_tは無限和である。
  - $g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$
    - 直接得ることが難しい…
- 状態価値関数 $V^{\pi}(s_t) \approx g_t$登場
  - 収益を推定する関数。（推定バイアスは乗るよ！）
  - $\boldsymbol{V^{\pi}(s_t) = r_{t+1} + \gamma V^{\pi}(s_{t+1})}$
  - Temporal Difference (TD) 誤差による更新式
    - $\delta_t = r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$
    - $V^{\pi}_{new}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\delta_t$, αは学習率
      - １ステップに分解して近似できる！



λ-Returnについて（n-step Return）

1. n-step Return

- nステップの報酬を用いて収益 $g_t^{(n)}$ を推定する。
  → 価値関数のバイアスを小さくするため。
  - $g_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V^{\pi}(s_{t+n})$

- バイアスと分散
  - nが小さいと価値関数の推定が大きい：バイアス大/分散小
  - nが大きいと経験の軌道の分散が大きい：バイアス小/分散大

- では、全ての n について計算してしまえばいい。
  - → λ-Returnへ。

λ -Returnについて

- Eligibility Trace Decay $\lambda \in [0,1]$
  - 等比級数の無限和 $1+\lambda+\lambda^2+\cdots=\frac{1}{1-\lambda}$ を用いて$g_t^{(n)}$をn=1から減衰総和。
    - $g_t^{[\lambda]}=(1-\lambda)(g_t^{(1)}+\lambda g_t^{(2)}+\cdots\lambda^{n-1}g_t^{(n)}+\cdots)$
  - 実際は有限ステップNまでの経験で打ち切って近似する。
  - 実装する時は再帰展開して後方から$g_{t+N}^{[\lambda]}$を計算する。
    - $g_t^{(n)}=r_{t+1}+\gamma r_{t+2}+\cdots+\gamma^{n-1}r_{t+n}+\gamma^n V^\pi(s_{t+n})$
    - $g_t^{[\lambda]}=(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}g_t^{(n)}$
      - $=r_{t+1}+\gamma\{(1-\lambda)V^\pi(s_{t+1})+\lambda g_{t+1}^{[\lambda]}\}$

        λが大きいと実報酬の比重が大きい。小さいと推定の比重が大きい。
    - 参考 https://ai.stackexchange.com/questions/11783/how-can-the-lambda-return-be-defined-recursively

Dreamer方式の方策と価値関数の更新

Dreamer [D. Hafner+, 2019, 2023]の採用理由

- 実時間上で動作する自律機械知能のPolicyの学習をスケールするため。
  - 経験データから獲得した 世界モデル [D. Ha+, 2018] のなかで 行動し（Dreaming)、空想の経験データを生成する。それを用いて学習する。

    → **現実時間に縛られない。計算機パワーで大規模化可能。**

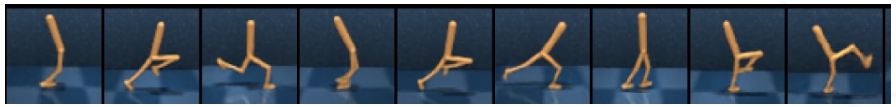
  - 今まで採用していた PPO [J. Schulman+, 2017]はモデルフリー手法であり、学習データを集めるのに環境と直接インタラクションする必要がある。
  - さらに、**収束にも時間がかかる。学習効率が悪い。**

登場モデル

- 方策π_θと価値関数V_ϕ^π
  - $a_t \sim \pi(a_t|s_t), v_t = V_\phi^\pi(s_t)$
  - それぞれパラメータθ, ϕをもつ深層モデル。これを更新する。

- 重要: Forward Dynamics f
  - $f(s_t, a_t) \rightarrow p_s(s_{t+1}|s_t, a_t),\ p_r(r_{t+1}|s_t, a_t)$
  - 経験軌道データτを元に世界の状態遷移 $s_t, a_t \rightarrow r_{t+1}, s_{t+1}$を近似した、**微分可能な**（深層）モデル

  - 状態と行動から、確率的に 報酬および次の状態を予測する。



Dreamerの学習

- 空想上の軌道（夢の中）で学習する。
  - Hステップの軌道をForward Dynamicsで生成

    $s_0 \leftarrow$ *初期状態を与える。（実経験データなどから）*

    for t *in* $0 \ldots H-1$ *step do:*

    $a_t \sim \pi_\theta(a_t|s_t)$

    $p_s, p_r \leftarrow f(s_t, a_t)$

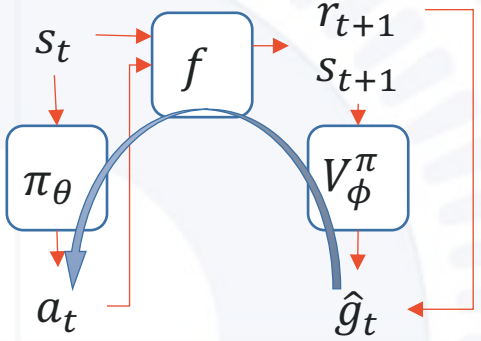
    $\hat{s}_{t+1} \sim p_s(s_{t+1}|s_t, a_t),\ \hat{r}_{t+1} \sim p_r(r_{t+1}|s_t, a_t)$

    - 空想軌道 $\tau = \{s_0,\ a_{0:H-1},\ \hat{s}_{1:H},\ \hat{r}_{1:H}\}$

  - λ – Return $g^{[\lambda]}_{0:H-1}$を軌道τから計算
    - 方策π_θは収益 $g^{[\lambda]}_{0:H-1}$から勾配を計算し、収益を最大化するようθを更新。
    - 価値関数V^π_ϕは誤差 $\left\| V^\pi_\phi(\hat{s}_{0:H-1}) - g^{[\lambda]}_{0:H-1} \right\|_2^2$を最小化するように$\phi$を更新。

ポイント

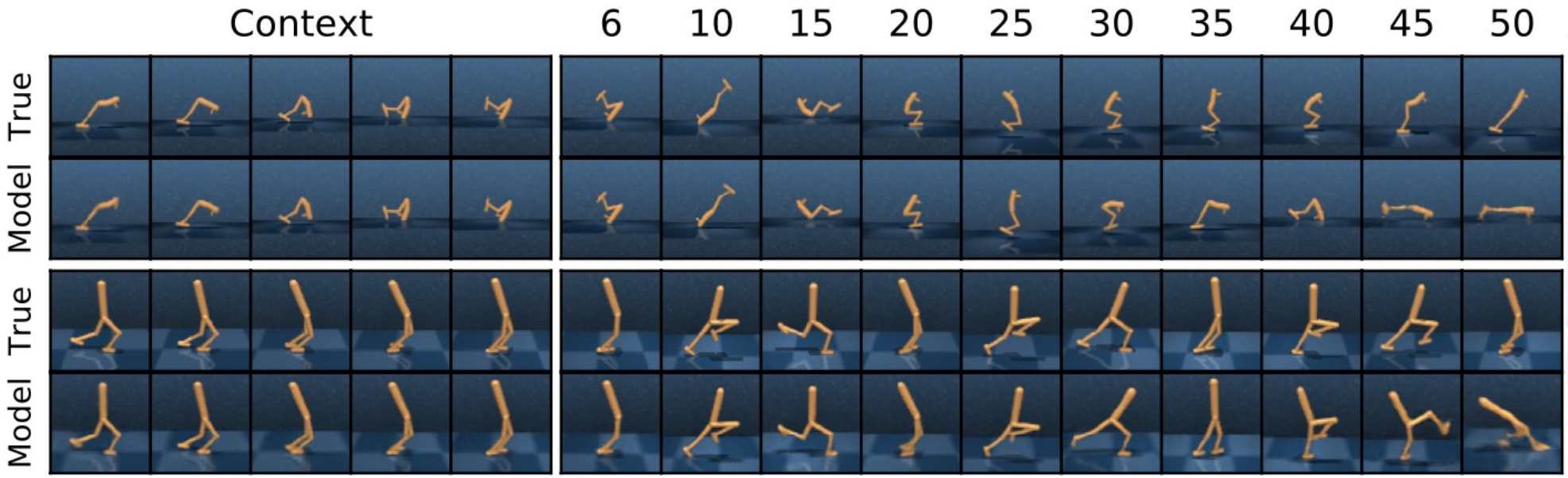
- Forward Dynamics fが微分可能である
  - 推定した収益から**直接勾配を計算**して、方策π_θを更新できる。
  - （一般の強化学習ではこういったことができない）



- fが確率的に予測する。
  - 近似する世界そのものは本質的に確率的であるため。
  - fが決定的に未来を予測してしまうと、fが上手く学習できていない時にπがfの”バグ”をハックして学習してしまう。[D. Ha+, 2018]
    - 不正に報酬を得たり…
    - 現れた敵を消したり…

Dreamer [D. Hafner+, 2018]の実動作について

- 空想の中で行動した潜在空間を再構成し、可視化したもの。
  - 50 ステップにわたり、実際の軌跡と近い予測が崩壊することなく生成できている。



参考文献

- D. Hafner et al, “Dream to Control: Learning Behaviors by Latent Imagination”, arXiv:1912.01603, 2019.
- D. Hafner et al, “Mastering Diverse Domains through World Models”, arXiv:2301.04104, 2023.
- D. Ha and J. Schmidhuber, “Recurrent World Models Facilitate Policy Evolution”, NeurIPS, 2018. https://worldmodels.github.io
- J. Schulman et al, “Proximal Policy Optimization Algorithms”, arXiv:1707.06347, 2017.
- 今井翔太, “誰でもわかる強化学習”, https://speakerdeck.com/imai_eruel/reinforcement-learning-for-everyone
- Alcia Solid, “本質を捉えたデータ分析のための分析モデル入門”, ソシム, 2022.