5 深層強化学習 – A3cアルゴリズム

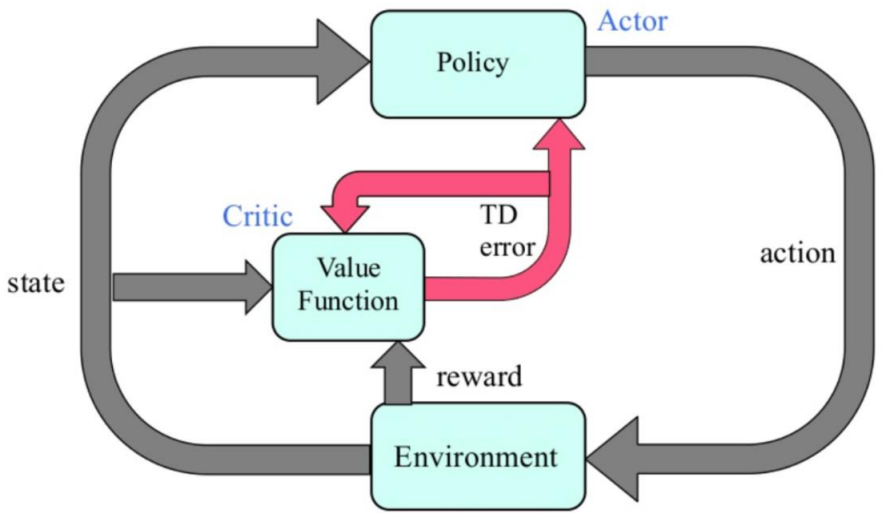
Asynchronous Advantage Actor-Critic（A3C） とは

- 強化学習の学習法の一つ; DeepMindのVolodymyr Mnih(ムニ)のチームが提案
- 特徴：**複数のエージェントが同一の環境で非同期に学習**すること

【“A3C”の名前の由来： 3 つの”A” 】

- **Asynchronous** ➡ 複数のエージェントによる**非同期な並列学習**
- **Advantage** ➡ 複数ステップ先を考慮して更新する手法
- **Actor** ➡ 方策によって行動を選択
- **Critic** ➡ 状態価値関数に応じて方策を修正

※Actor-Criticとは
行動を決めるActor（行動器）を直接改善しながら、方策を評価するCritic（評価器）を同時に学習させるアプローチ

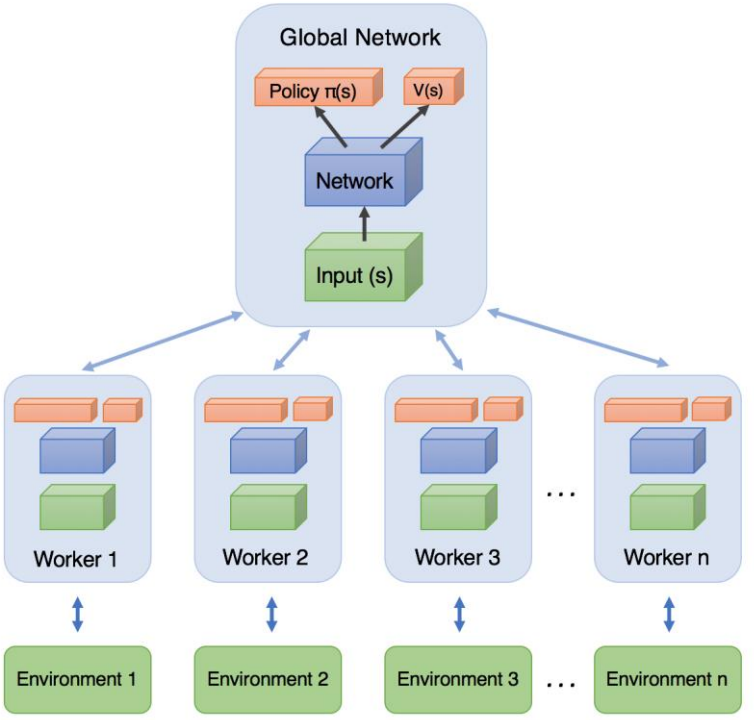


引用：Sutton, Berto, “Reinforcement Learning – an introduction.” 1998

【A3Cによる非同期 (Asynchronous) 学習の詳細】

- 複数のエージェントが並列に自律的に、rollout (ゲームプレイ) を実行し、勾配計算を行う
- その勾配情報をもって、好き勝手なタイミングで共有ネットワークを更新する

- 各エージェントは定期的に自分のネットワーク (local network) の重みをglobal networkの重みと同期する
- 共有ネットワーク = パラメータサーバ



引用：https://pylessons.com/A3C-reinforcement-learning/

参考論文：https://arxiv.org/abs/1602.01783

並列分散エージェントで学習を行うA3Cのメリット

① 学習が高速化
② 学習を安定化

②について：

- 経験の自己相関が引き起こす学習の不安定化は、強化学習の長年の課題
- DQNは Experience Replay (経験再生) 機構を用いてこの課題を解消
  - バッファに蓄積した経験をランダムに取り出すことで経験の自己相関を低減
  - しかし、経験再生は基本的にはオフポリシー手法でしか使えない
- **A3Cはオンポリシー手法**であり、**サンプルを集めるエージェントを並列化することで自己相関を低減**することに成功した

【A3Cの難しいところ】

- Python言語の特性上、非同期並列処理を行うのが面倒
- パフォーマンスを最大化するためには、大規模なリソースを持つ環境が必要

【A2Cについて】

- A3Cの後に**A2C**という手法が発表された
- A2Cは同期処理を行い、Pythonでも実装しやすい
- 各エージェントが中央指令部から行動の指示を受けて、一斉に１ステップ進行し、中央指令部は各エージェントから遷移先状態の報告を受けて次の行動を指示する
- **性能がA3Cに劣らない**ことがわかったので、その後よく使われるようになった

５深層強化学習 – A3cアルゴリズム

【分岐型 Actor-Critic ネットワーク】

一般的なActor-Criticでは、方策ネットワークと価値ネットワークを別々に定義し、別々のロス関数（方策勾配ロス/価値ロス）でネットワークを更新する

A3Cは**パラメータ共有型のActor-Critic**：１つの**分岐型のネットワークが、方策と価値の両方を出力**し、たった１つの「**トータルロス関数**」でネットワークを更新

【A3Cのロス関数】

ロス関数は３項目で表せる：

アドバンテージ方策勾配、価値関数ロス、方策エントロピー

Total loss =– アドバンテージ方策勾配 +α・価値関数ロス –β・方策エントロピー

係数αとβはハイパーパラメータ

特に係数βは探索の度合いを調整するハイパーパラメータ

5 深層強化学習 – A3cアルゴリズムのアドバンテージ方策勾配項

- 方策勾配法では、 θ をパラメータに持つ方策 π_θ に従ったときの期待収益 ρ_θ が最大になるように、 **θ を勾配法で最適化**する
- **方策勾配定理**により、パラメータの更新に用いられる**勾配 $\nabla_\theta \rho_\theta$** は、以下の式で表される

式1 $$\nabla_\theta \rho_\theta = \mathbb{E}[\nabla_\theta \log \pi(a|s,\theta)(Q^{\pi_\theta}(s,a) - b(s))]$$

b(s) は価値の「ベースライン」：これを引くことで、推定量の分散が小さくなり学習の安定化が期待できる

- （式1）の中の $Q^{\pi\theta}(s, a) - b(s)$ に**アドバンテージ関数**を設定する
- A3Cのアルゴリズムの特徴としては、勾配を推定する際に、b(s)の推定には価値関数$V^{\pi\theta}(s)$、Q(s,a)の指定には、**kステップ先読みした収益**（式2）を用いる

式2 $$\sum_{i=0}^{k-1} \gamma^i R_{t+i+1} + V_\theta^{\pi_\omega}(X_{t+k})$$

つまり、下記（式3）の期待値が（式1）の勾配と等価

式3 $$\nabla_\theta \log \pi(a|s;\theta)(\sum_{i=0}^{k-1} \gamma^i R_{t+i+1} + V_\theta^{\pi_\omega}(s_{t+k}) - V_\theta^{\pi_\omega}(s_t))$$

5 深層強化学習 – A3cのAtari 2600 における性能検証

- Atari 2600において、人間のスコアに対して規格化した深層強化学習の結果
- **A3Cは、16 CPUコアのみ使用、GPU使用なし**　（1~4日間の訓練）
- 他のエージェントはNvidia K40 GPUを使用（8~10日間の訓練）

https://arxiv.org/pdf/1602.01783

結論：**<u>より短い訓練時間でGPUなしでも、A3Cのスコアが顕著に高い</u>**

1 日間の訓練で、CPUのみでも、Dueling D-DQNの精度に到達

手法 Method	訓練時間 Training Time	平均 Mean	中央値 Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

５深層強化学習 – （参考）A3cアルゴリズムの方策エントロピー項

$$-\sum_a \pi(a_t|s_t)\log\pi(a_t|s_t)$$

- **方策のランダム性の高い（＝エントロピーが大きい）方策**に**ボーナスを与える**ことで、方策の収束が早すぎて**局所解に停滞する事態を防ぐ**効果がある
- 方策エントロピー項の追加は、方策関数の正則化効果が期待できる

（追記）方策のランダムさの指標
ある状態sの入力について出力される行動の採用確率が (a1, a2, a3, a4) = [0.25 0.25, 0.25, 0.25] の場合と、 (a1, a2, a3, a4) = [0.85, 0.05, 0.05, 0.05] の場合では、前者のほうが方策のエントロピーが大きい状態である

※A3Cの実装の参考（TensorFlow Blogより）
https://blog.tensorflow.org/2018/07/deep-reinforcement-learning-keras-eager-execution.html