Eligibility Traces for Off-Policy Policy Evaluation

Precup et al. (2000)

2020/09/27 Tokuma Suzuki @ OPE 勉強会

アジェンダ

- バックグラウンド
- MDP の記法
- importance sampling
- Pre-Decision Algorithm
  - MC method
  - TD method
- Tree Backup Algorithm
- Empirical Comparison

研究の背景

- 強化学習における最大のトレードオフ: **探索と活用**
  - 貪欲方策を取り続けたいが、他の行動をためさないと最適な方策を見つけられない...
- 最適方策を見つける方法
  - on-policy learning
  - off-policy learning
    - 何らかの方策から生成されたデータで最適方策を学習

現代的モチベーション

- 探索と活用以外にも off-policy learning が必要な場面が存在
    - 別の方策の性能を見積もりたいが、実運用は高コスト...
        - 金銭的コスト
        - 性能が悪くユーザーが離れるリスク
- 現在の方策の下で収集されたデータを用いて異なる方策の性能を評価したい！
    - 最適な方策を探すコストを低下させる事が可能

記法

- m: エピソード
- T_m: エピソードmの継続期間
- $s_t \in S$: tにおける状態, Sは有限集合
- $a_t \in A$: tにおける行動, Aは有限集合
- $p^a_{s,s'}$: 状態 s において行動 a を選択した際の状態推移確率
- r^a_s: 状態 s において行動 a を選択した際の利得
- $\pi : S \times A \to [0, 1]$: 方策

定義

- 方策πにおける状態行動価値関数

$$Q^{\pi}(s,a) = E_{\pi}[r_1 + \gamma r_2 + \cdots + \gamma^{T-1} r_T \mid s_0 = s, a_0 = a]$$

- target policy, π: 評価したい方策
- behavior policy, b: データを生成する方策
  - b がソフトである: $b(s,a) > 0 \ \forall s, a$
- 方策bによって生成されたデータを用いてQ^{π}を推定する方法を検討

参考: その他の方法との違い

- TD 法: on-policy
  - ある方策のもとでの価値を推定
- Q-learning: off-policy learning
  - **貪欲方策における**価値を推定し最適方策を学習
  - behaivior と target ともに時間を通じて変化
- 提案手法: off-policy policy evaluation
  - **異なる方策における**価値を推定
  - 各方策は不変で最適方策を求めることが主眼ではない

importance sampling

- target, behavior policy が生成するデータの分布の違いを補正

$$
\begin{aligned}
E_d\{x\} &= \int_x x d(x) dx = \int_x x \frac{d(x)}{d'(x)} d'(x) dx \\
&= E_{d'}\left\{ x \frac{d(x)}{d'(x)} \right\} \\
&\approx \frac{1}{n} \sum_{i=1}^{n} x_i \frac{d\left(x_i\right)}{d'\left(x_i\right)}
\end{aligned}
$$

- この推定量は consistent かつ unbiased. (Rubinstein, 1981)

Weighted importance sampling

- 重みを考慮することで実用上より高速で安定的に推定可能
  - 重み: $\frac{d(x_i)}{d'(x_i)}$

$$\frac{\sum_{i=1}^{n} x_i \frac{d(x_i)}{d'(x_i)}}{\sum_{i=1}^{n} \frac{d(x_i)}{d'(x_i)}}$$

- consistent but biased estimator of $E_d[x]$.
- このあたりが IPW に関係してくる？

Estimating Q function

- bの下での状態行動価値関数Q^bからQ^πを推定したい
  $Q^b(s,a) = E_b[r_1 + \gamma r_2 + \cdots + \gamma^{T-1} r_T \mid s_0 = s, a_0 = a]$
- $Q^\pi(s,a)$の *first-visit importance sampling estimator*

$$Q^{IS}(s,a) \stackrel{\text{def}}{=} \frac{1}{M} \sum_{m=1}^{M} R_m w_m$$

- weighted importance sampling

$$Q^{ISW}(s,a) \stackrel{\text{def}}{=} \frac{\sum_{m=1}^{M} R_m w_m}{\sum_{m=1}^{M} w_m}$$

エピソードを利用した価値推定

- M: (s, a)を訪問したことのあるエピソード数
- $R_m \overset{\text{def}}{=} r_{t_m+1} + \gamma r_{t_m+2} + \ldots + \gamma^{T_m - t_m - 1} r_{T_m}$
  - エピソードmでの(s, a)のペアが出現以降の割引現在価値
- $w_m \overset{\text{def}}{=} \frac{\pi_{t_m+1}}{b_{t_m+1}} \frac{\pi_{t_m+2}}{b_{t_m+2}} \cdots \frac{\pi_{T_m}}{b_{T_m-1}}$
  - w_m: importance sampling のウェイト, 方策による選択確率の調整
- t_m: エピソードmにおいて初めて(s, a)を訪問した時刻

より効率的で実装が容易な方法を提案！

Per-Decision Algorithm

- step-by-step に重点サンプリングを実行 → 計算量削減
  - 今期の利得は将来の $\frac{\pi}{b}$ に依存しないことを利用して分散を低下させる

$$Q^{PD}(s,a) \stackrel{\text{def}}{=} \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{T_m-t_m}\gamma^{k-1}r_{t_m+k}\prod_{i=t_m+1}^{t_m+k-1}\frac{\pi_i}{b_i}$$

- 歴史に沿って各利得に重みをつける

Theorem 1

The per-decision importance sampling estimator Q^{PD} is a consistent unbiased estimator of Q^{π}.

Eligibility-Trace version

- TD 法に importance sampling の考え方を利用
- ランダムネスの少ない TD 法を利用するため、更に分散を低下させられる
  - MC: 考慮するすべての時点の利得,行動, 状態遷移に依存
  - TD: 一部の時点の利得,行動, 状態遷移に依存
- 紹介するアルゴリズムは online だが、更新が offline の場合のときにQ^{π}に確率収束
  - theorem 2 参照
- まずは TD 法と Eligibility-Trace の復習

TD 法

online である方策の下での価値関数を推定

- $Q(s_t, a_t)$: s_tにおいてa_tを選んだときの価値 ← 更新対象
  - bのもとで行動a_tを選択 → 報酬rとs_{t+1}が決定
  - $Q(s_t, a_t)$だと思っていたものが実際には$r + \gamma Q(s_{t+1}, a_{t+1})$だった
  - 2 つの差は異なる時点での(s_t, a_t)の予測価値の差異！(temporal difference)
  - これを用いて価値を更新していく方法

$$Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha(r_t + Q(s_{t+1}, a_{t+1}))$$

Eligibility Trace(ラフイメージ)

- TD 法は更新幅が少なく,初期値の影響が大きいため効率が良くない
    - heuristic を導入して更新幅を増幅しよう！
- 予測価値の差が生まれた原因について 2 つの heuristic を導入
    - Frequency heuristic:　頻繁に起こるものを重視
    - Recency heuristic: 最近起きたものを重視

厳密でない説明なので興味がある場合は foward-view, backward-view でググって

Algorithm 1

Online, Eligibility-Trace Version of Per-Decision Importance sampling

1. 各状態について eligibility trace を更新

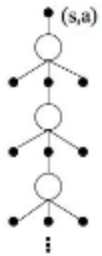
$$e_t(s,a) = e_{t-1}(s,a) \underbrace{\gamma\lambda}_{\text{recency}} \underbrace{\frac{\pi(s_t,a_t)}{b(s_t,a_t)}}_{\text{importance sampling}}, \ \forall s,a$$

$$\underbrace{e_t(s_a) = 1, \text{ iff } \ t = t_m(s,a)}_{\text{frequency}}$$

2. TD 誤差を計算: $\delta_t = r_{t+1} + \gamma \frac{\pi(s_t,a_t)}{b(s_t,a_t)} Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)$
3. 行動状態価値関数を更新: $Q_{t+1} = Q_t(s,a) + \alpha e_t(s,a)\delta_t \quad \forall s,a$

Tree Backup Algorithm

- behavior policy に対して non-starving のみ仮定
  - ある(s,a)に 2 度と訪問しない時点に決して到達しない
  - 非定常, 非マルコフ, 未知でもよい
- target policy に従いつづけた場合にあり得る可能性をすべて考慮



Tree Backup Algorithm

- n-step tree-backup estimator

$$Q_n^{TB}(s,a) = \frac{1}{M}\sum_{m=1}^{M} \underbrace{\gamma^n Q(s_{t_m+n}, a_{t_m+n}) \prod_{i=t_m+1}^{t_m+n} \pi_i}_{\text{実際にとった行動から得られる価値}} + \underbrace{\sum_{k=t_m+1}^{t_m+n} \gamma^{k-t_m+1} \prod_{i=t_m+1}^{k-1} \pi_i \left(r_k + \gamma \sum_{a \neq a_k} \pi(s_k, a) Q(s_k, a) \right)}_{\text{取られなかった行動から得られる価値}}$$

- importance sampling をしないで済む
- per-decision と同様に eligibility-trace を利用できる

Algorithm 2

Online, Eligibility-Trace Version of Tree Backup

1. 各状態について eligibility trace を更新

$$e_t(s,a) = e_{t-1}(s,a) \underbrace{\gamma\lambda}_{\text{recency}} \pi(s_t,a_t),\ \forall s,a$$

$$\underbrace{e_t(s_a) = 1, \text{ iff } \ t = t_m(s,a)}_{\text{frequency}}$$

2. TD 誤差を計算: $\delta_t = r_{t+1} + \gamma \sum_{a \in A} Q(s_{t+1},a) - Q(s_t,a_t)$
3. 行動状態価値関数を更新: $Q_{t+1} = Q_t(s,a) + \alpha e_t(s,a)\delta_t \quad \forall s,a$

オフライン版はQ^πに確率収束(定理 3)

Per-Decision と Tree Backup の違い

- どちらも歴史に依存して価値を更新していくという点では同じ
  - Per-Decision: 実際に起こった利得を用いて更新
  - Tree backup: その時点で起こりうるすべての可能性を考慮
- 実験したところ tree backup のほうが良い性能！
  - バンディットに IPW を適用する話につながるのかこれ？

Empirical Comparison

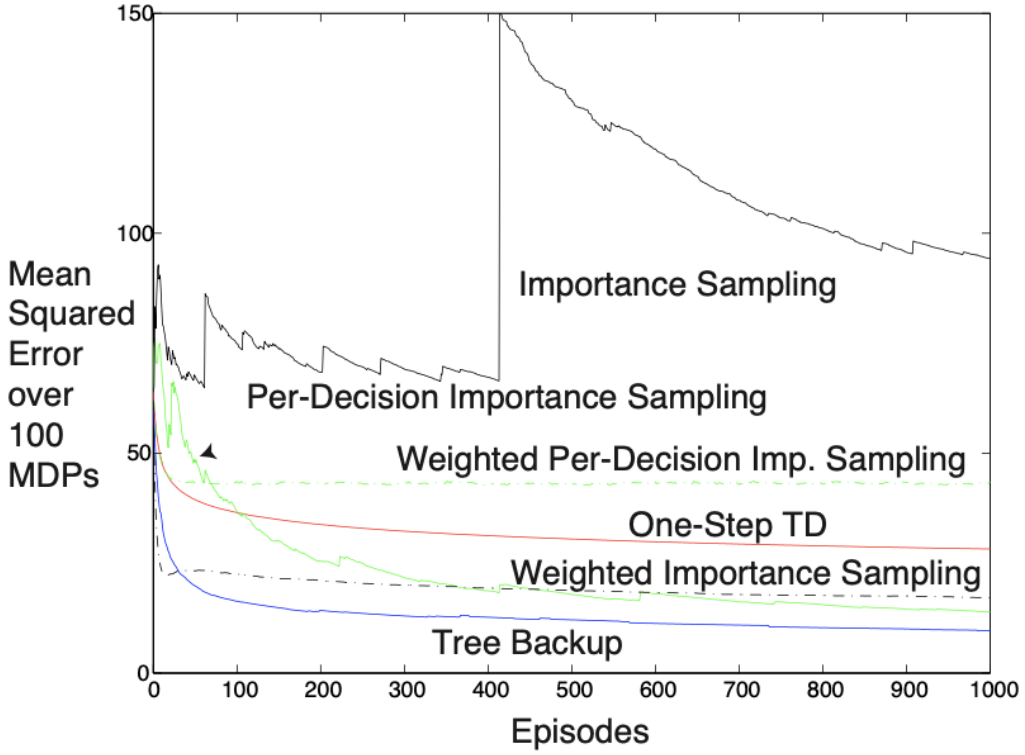
実験設定

- 各推定方法に対し 100 個の MDP をランダムに生成
- 1 terminal state, 100 nonterminal states
- 2 つの行動が存在
  - ランダムに選ばれた 4 つの状態へランダムに移動
- target policy: 1 つめの行動を80%の確率で選択
- behavior policy: 2 種類検討
  - uniform behavior: 半々の確率で選択
  - different behavior: 2 つめの行動を80%の確率で選択

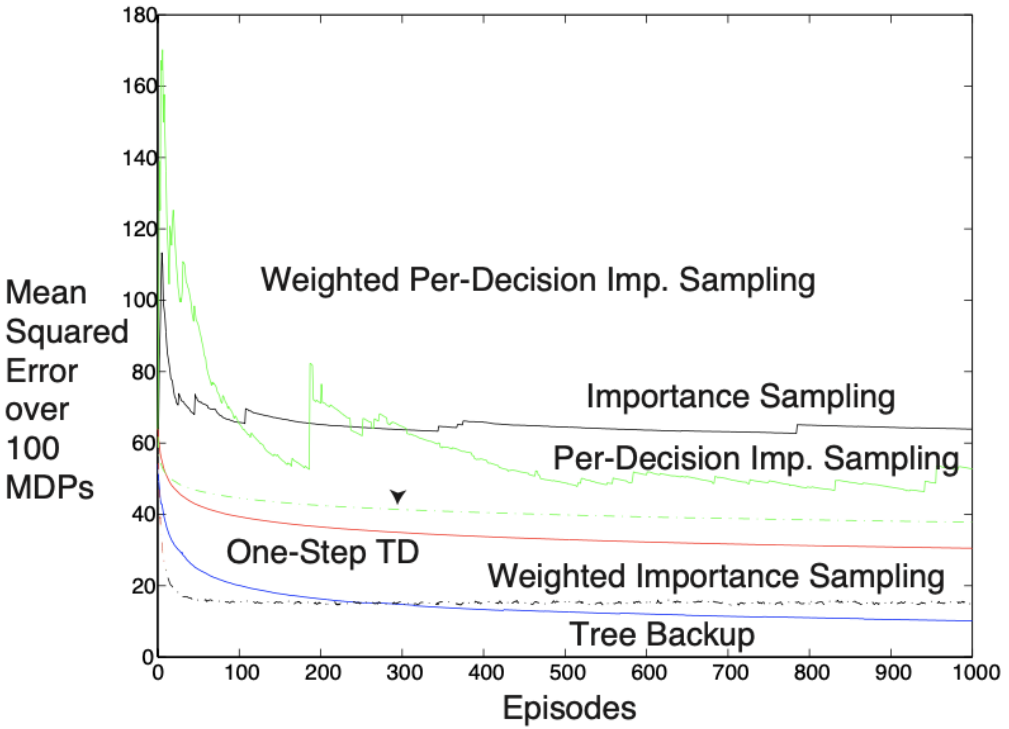
実験設定

- immediate reward: $[0, 1]$から一様にランダムに選ばれる
- 割引率: $\gamma = 1$
- 性能評価
  - 各エピソード終了後各状態-行動ペア, 100 個の MDP の結果を平均
  - 真の状態行動価値との mean-square error で評価

結果



(a) Uniform Behavior



(b) Different Behavior

結果まとめ

- importance sampling: 収束が遅く、分散が大きい
    - weighted の場合は良い結果
- per-decision importance sampling: behavior policy により異なる
    - weighted の場合は性能悪化
- tree-backup: 中長期的に最も効率的
    - 今回の実験では trace の影響が早く薄れすぎないように正規化している
- 全体的に原因の検討はなされていない