これからの強化学習

第2章 強化学習の発展的理論

Yuma Yamakura

目次

- 2.1 統計学習の観点から見たTD学習
- 2.2 強化学習アルゴリズムの理論性能解析と
  ベイズ統計による強化学習のモデル化
- 2.3 逆強化学習(Inverse Reinfocement Learning)
- 2.4 試行錯誤回数の低減を指向した手法
  : 経験強化型学習 XoL
- 2.5 群強化学習法
- 2.6 リスク考慮型強化学習
- 2.7 複利型強化学習

2.3 逆強化学習(Inverse Reinforcement Learning)

2.3.1 報酬設計問題

2.3.1 強化学習の優位性

- 環境のダイナミクスが未知でも適用できる
  ⇔動的計画法
- 行動の正解･不正解が既知である必要はない
  ⇔教師付き学習(NNなど)
- 報酬の遅れを許容
  →リアルタイム性が悪くても学習可能
  (一連のエピソードに対して報酬を付与,
  モンテカルロ的な思考)

2.3.1 強化学習の優位性

- 環境のダイナミクスが未知でも適用できる
  ⇔動的計画法
- 行動の正解･不正解が既知である必要はない
  ⇔教師付き学習(NNなど)
- 報酬の遅れを許容
  →リアルタイム性が悪くても学習可能
    (一連のエピソードに対して報酬を付与,
    モンテカルロ的な思考)

これらは**報酬**という**設計者が決めるパラメータ**による性質

2.3.1 報酬決定の難しさ

• 報酬とは？

感覚的には, MDPにおける一連の遷移の良し悪し

→直観的にはわかりやすい(はず)

• 実問題(状態空間が大規模なもの)

報酬の決定は難しい(ほかの機械学習でいう

ハイパーパラメータの決定のようなもの)

2.3.1 報酬決定の難しさ

• どういう点が難しいか？

状態空間が大きいと, 初期状態から終端状態までに
かかる時間が膨大に(次元の呪い)

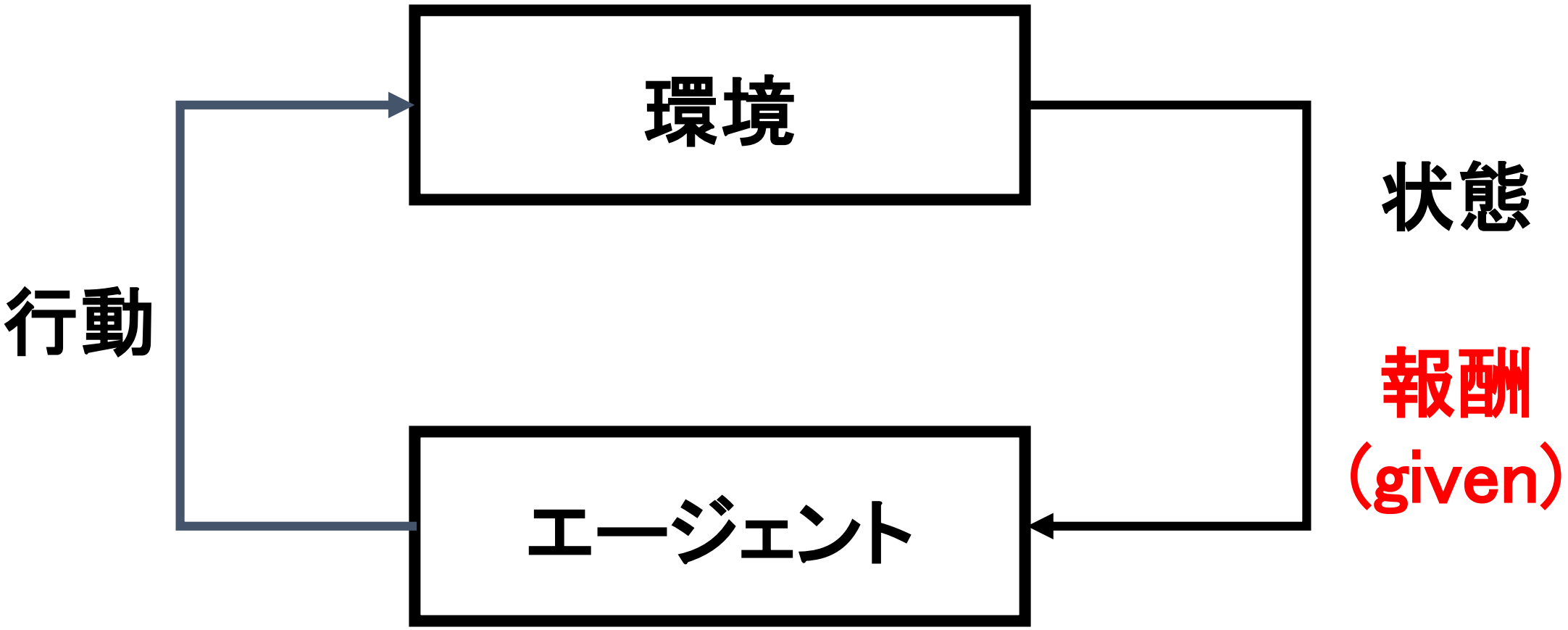
1)終端状態になるまでに時間がかかる
　→良い報酬が方策に影響を与えるまでに時間がかかる
2)マルチエージェント系で,
　どの行動がよい報酬に結び付いたかわからない
　→その報酬は良い設定だったのか？

2.3.1 報酬設計

・報酬を設計することは難しい
→報酬を設定する「報酬設計」問題考える

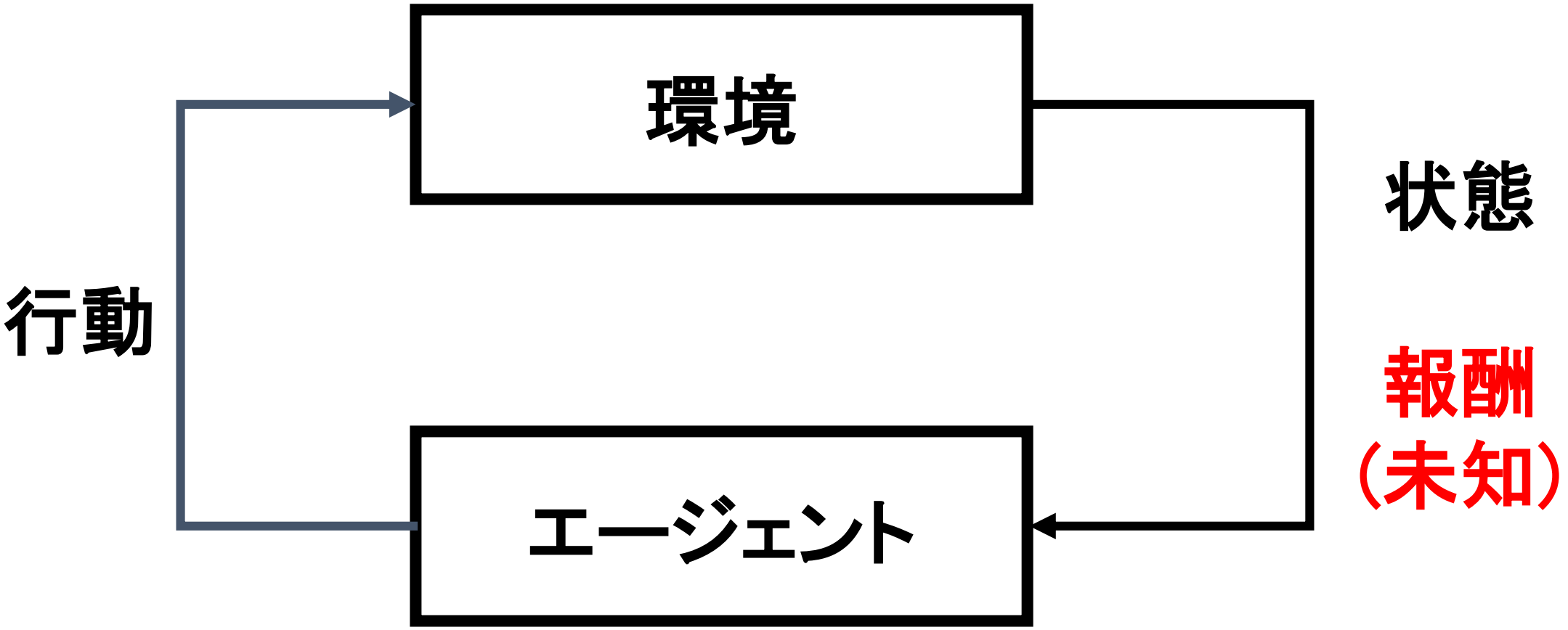
逆強化学習(Inverse Reinforcement Learning, IRL)は
報酬設計のよい方法

2.3.1 強化学習のイメージ



報酬をもとに方策を学習

2.3.1 逆強化学習のイメージ



方策(エキスパートの行動)をもとに報酬を学習

2.3.1 逆強化学習の種類

- Russellの逆強化学習法

最適な行動系列と環境モデルがgiven

- Ngの逆強化学習法

有限状態空間→線形計画法　無限状態空間→モンテカルロ法

- Abbeelの逆強化学習法 : projection法

報酬関数の推定過程で最適方策を獲得(見習い学習)

2.3.2 Ngの逆強化学習法
：有限状態空間を対象とする場合

2.3.2 Ngの逆強化学習(有限状態空間)

先ほども言った通り…有限状態空間→線形計画法で解く！

各状態$s_i(i = 1, \dots, M)$における最適行動a_1がgiven
行列P_aは行動aの状態遷移確率で, $M \times M$行列

$P_{ii'}^{a}$　：状態s_iから行動aで状態$s_{i'}$に遷移する確率

$P_a(i)$: P_aのi行目のベクトル

λ : ペナルティ係数

以上がgiven

2.3.2 Ngの逆強化学習(有限状態空間)

LP問題の定式化

$$maximize: \sum_{i=1}^{N} \min_{a \in a_2,\ldots,a_k} \left\{ \left(P_{a_1}(i) - P_a(i) \right) \left(I - \gamma P_{a_1} \right)^{-1} R \right\} - \lambda \|R\|_1$$

$$subject\ to: \left(P_{a_1} - P_a \right) \left(I - \gamma P_{a_1} \right)^{-1} R \geq 0$$

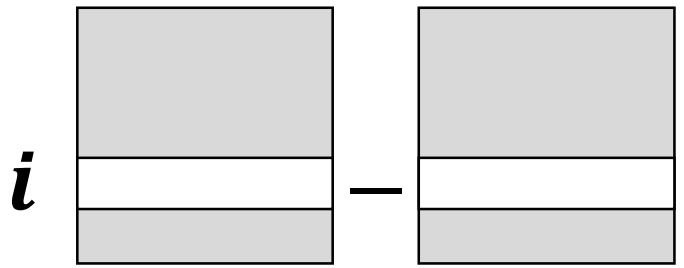
2.3.2 Ngの逆強化学習(有限状態空間)

$$\left(P_{a_1}(i) - P_a(i)\right) \quad \left(I - \gamma P_{a_1}\right)^{-1} \quad R$$

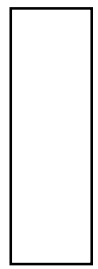
$$\| $$

$$I + \gamma P_{a_1} + \left(\gamma P_{a_1}\right)^2 + \cdots$$

$$\downarrow$$



各遷移でどの状態に
いるかを表すイメージ



2.3.2 Ngの逆強化学習(有限状態空間)

LP問題の定式化

$$maximize: \sum_{i=1}^{N} \min_{a \in a_2,\ldots,a_k} \left\{ \left(P_{a_1}(i) - P_a(i) \right) \left(I - \gamma P_{a_1} \right)^{-1} R \right\} - \lambda \|R\|_1$$

理想との報酬のずれ

理想の報酬のずれを最小化

Lasso的なやつ(Rの成分の抑え込み)

これを最大化するようなRは, 有効な報酬

(小さすぎる報酬は学習に使いにくい)

2.3.2 Ngの逆強化学習(有限状態空間)

LP問題の定式化

$$subject\ to: \left(P_{a_1} - P_a\right)\left(I - \gamma P_{a_1}\right)^{-1} R \geq 0$$

これは理想の報酬のずれが正であることを示す
→行動a_1が最適となる保証

※これは状態空間の次元数と同じ次元

2.3.3 Abbeelの逆強化学習法：projection法

2.3.3 エキスパートとは？

・エキスパートの定義
各状態で最適な行動をとるエージェントのこと

エキスパートによる一連の行動の系列
→最適な行動の系列になるはず！
→その行動系列を得られるような強化学習を行える
　報酬を設計すればよいのでは？

2.3.3 Markov決定過程(MDP)

- MDP $< S, A, T, \gamma >$
  - S : 有限状態集合
  - A : 行動集合
  - $T = \{P_{sa}\}$: 状態遷移確率
  - γ : 割引率

特徴量$\phi : S \to [0, 1]^k$となるベクトル, 状態を表す

特徴期待値$\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]$

: 方策πに従ったときの期待割引累積特徴量

2.3.3 Abbeelの逆強化学習

- Abbeelの逆強化学習

エキスパートの特徴期待値との差がτ以下になる特徴期待値を学習を通して得ることができる報酬関数を推定

- 推定法
  - max-margin法 ・・・ Qpsolverを用いる
  - projection法 ・・・ 正射影ベクトルを用いる

2.3.3 projection法

- パラメータτは**エキスパートの特徴期待値$\hat{\mu}_E$**と**推定した報酬による特徴期待値μ^i**の差が十分近づいたことを判定するための閾値

1. ランダムに選んだ方策 π^0 から特徴期待値 $\mu^0 = \mu(\pi^0)$ を計算する
2. 各特徴期待値に対する重み $w^1 = \mu_E - \mu^0$ を計算し，$i = 1$ とする
3. 終了条件 $t^i \leq \tau$ を満たすまで以下を繰り返す：
   (a) 報酬関数 $R = w^i \cdot \phi$ と強化学習を用いて，最適方策 π^i を求める
   (b) 最適方策 π^i から特徴期待値 $\mu^{(i)} = \mu(\pi^i)$ を計算し，$i = i + 1$ とする
   (c) エキスパートの特徴期待値への射影ベクトル $\bar{\mu}^{i-1}$ を以下の式で計算する

$$\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^{\mathrm{T}}(\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^{\mathrm{T}}(\mu^{(i-1)} - \bar{\mu}^{(i-2)})}(\mu^{(i-1)} - \bar{\mu}^{(i-2)})$$

   (d) $w^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$，$t^{(i)} = ||\mu_E - \bar{\mu}^{(i-1)}||_2$

$$\hat{\mu}_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi\left(s_t^{(i)}\right)$$

図 2.3.1 projection 法　アルゴリズム

2.3.4 大規模計画問題への適用

2.3.4 大規模計画問題の報酬設計の問題

- 最適方策を導く報酬関数が複数存在するかもしれない
- 状態空間が膨大なので,
  制約条件が増えすぎて実行可能解が得にくい

⇒何らかの対策が必要

2.3.4 大規模計画問題の報酬設計の問題の解決法

- 最適方策を導く報酬関数が複数存在するかもしれない
  ⇒・多目的最適化問題のパレート解から良いものを選択
    ・学習効率を明示的に考慮した目的関数の導入

- 状態空間が膨大なので,
  制約条件が増えすぎて実行可能解が得にくい
  ⇒制約条件の緩和

2.3.4 報酬関数の評価：学習効率

報酬関数を1つに絞り込む
⇒別の報酬関数の導入

学習効率を最大にする報酬関数を探索する問題として定式化
⇒制約条件を緩和しながら遺伝的アルゴリズム(GA)で解く

2.3.4 学習効率の定義と目的関数

“最適” : 最適方策に収束する速さについて

学習効率のdef : $QLstep(R, epi)$
ある報酬関数RのもとでQ学習を行った際に, ある
エピソード数epiが終わるまでに要した累積ステップ数

⇒目的関数　$\min_{R} QLstep(R, epi)$
　とすればよい

2.3.4 GAへの適用

GAの各個体の遺伝子　：報酬関数R

各染色体：1つの状態の報酬

GAでは目的関数を直接扱えない

⇒制約条件を破った数に比例したペナルティを与える

2.3.4 GAの計算時間の削減

計算時間の削減のために, 各個体を2段階に分けて進化

1. 制約条件をすべて満たす解の探索
   制約条件に関するペナルティのみを用いて解を全て探索
   ⇒普通のGAにはないステップ

2. 1.を満たす解の中から学習効率を改善する解の探索
   個体の報酬でQ学習をして$QLstep(R, epi)$を計算
   ⇒学習効率に比例した評価値を与える

2.3.4 二段階進化のメリット

メリット

1.の制約条件を満たさない解は最適行動の獲得が保証されないので, 時間がかかる

⇒除外すると計算時間が早くなる

2.3.4 GAIRL

GAと逆強化学習を組み合わせた手法

Start Given：最適行動
1. 逆強化学習の制約条件を求める
2. GA で報酬関数を更新
 (a) 報酬関数が制約条件を満たすとき：
  ・この報酬関数を用いて Q 学習し，総ステップ数を評価値にセット
 (b) 報酬関数が制約条件を満たしていないとき：
  ・制約条件を破った個数分のペナルティを評価値にセット
3. 終了条件を満たしていれば End へ，そうでなければ 2. へ
End

図 2.3.2　GAIRL の疑似コード

2.3.4 逆強化学習の制約条件の緩和

Ngの逆強化学習の制約条件

$$subject\ to: \left(P_{a_1} - P_a\right)\left(I - \gamma P_{a_1}\right)^{-1} R \geq 0$$

は状態と行動の数に比例して状態空間が増加
⇒緩和しなきゃ大規模問題では使い物にならない

2.3.4 逆強化学習の制約条件の緩和

Ngの逆強化学習の制約条件

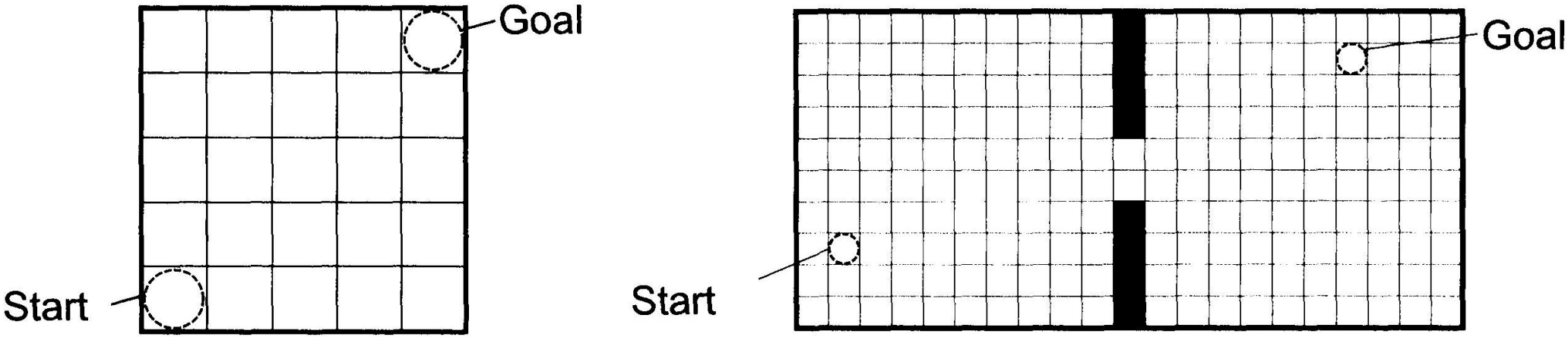
$$subject\ to: \left(P_{a_1} - P_a\right)\left(I - \gamma P_{a_1}\right)^{-1} R \geq 0$$

各状態で最適な行動a_{opt}が複数存在すれば,
どの最適行動を選択してもよい

$$subject\ to: \left(P_{a_1} - P_a\right)\left(I - \gamma P_{a_1}\right)^{-1} R \geq 0 \quad \forall a \in A \setminus a_{opt}$$

これで$a_{opt} \cap a_1$の数だけ制約条件を削減

2.3.4 シミュレーション

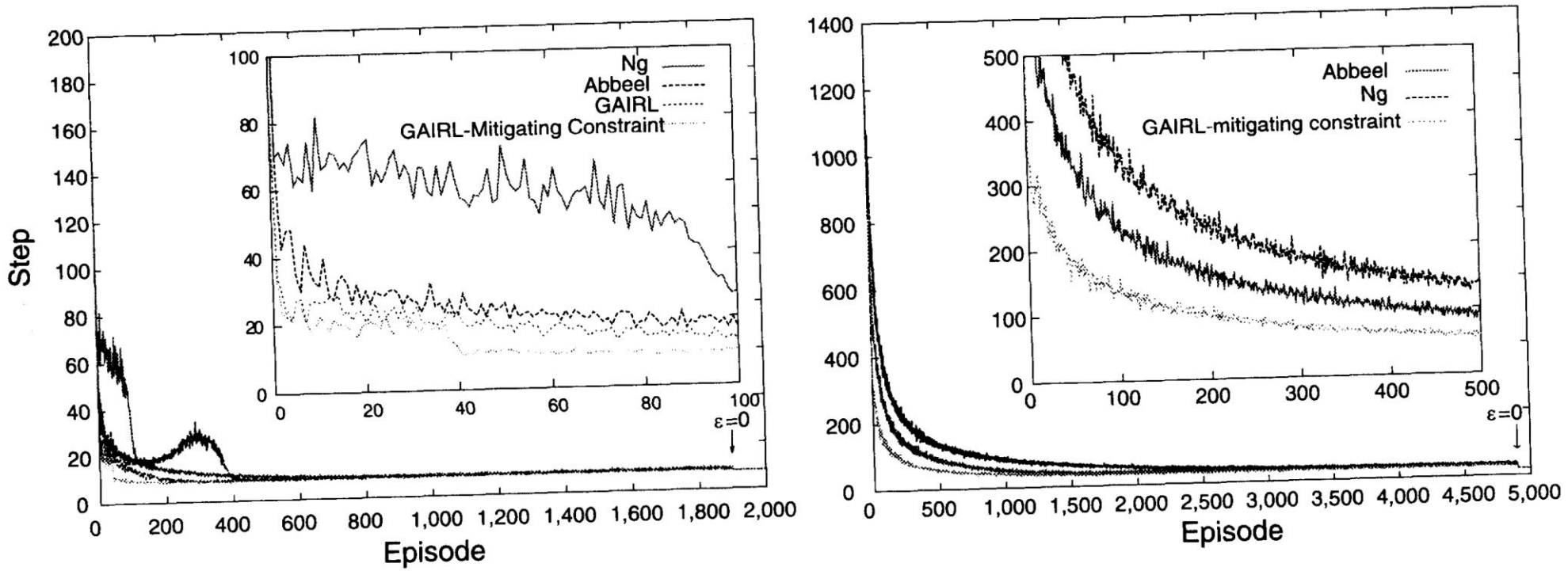


（a） 5×5-GridWorld　　（b） Two-Rooms

図 2.3.3　実験環境

※パラメータは本参照

2.3.4 シミュレーション結果



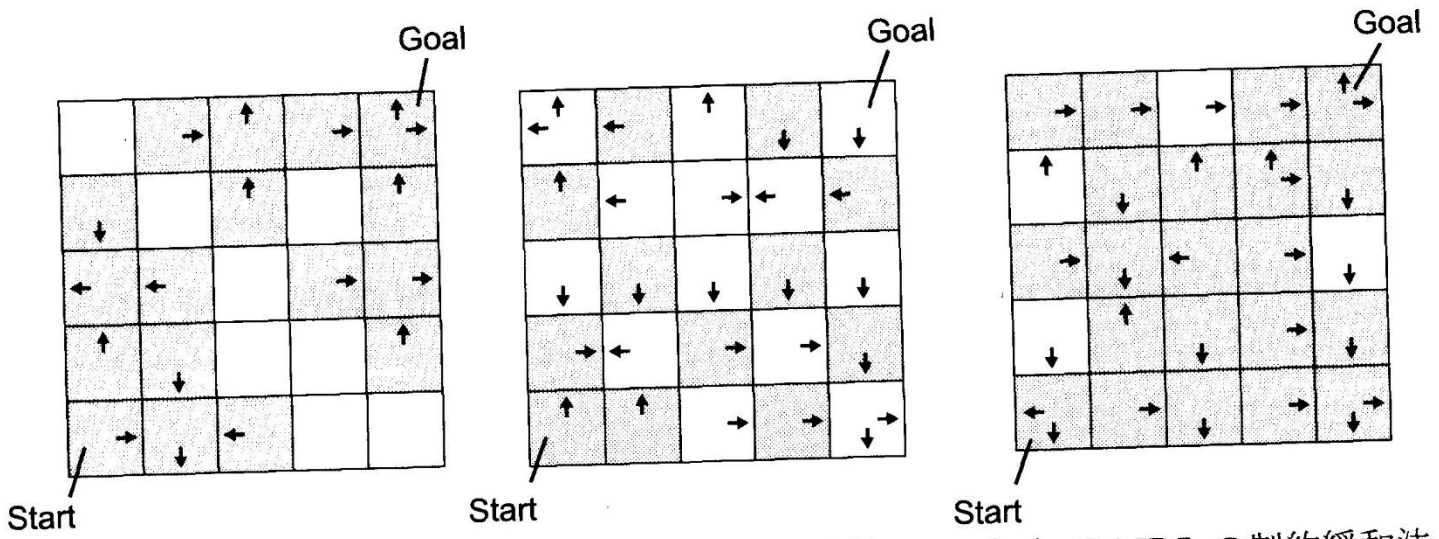
（a） 5×5-GridWorld

（b） Two-Rooms

図 2.3.4　最適な報酬関数抽出と制約緩和の効果：学習曲線の比較

GAIRL(緩和)>Abbeel>Ng の順で最適解を発見

2.3.4 報酬関数の考察



（a）Ng の方法　（b）Abbeel の方法　（c）GAIRL の制約緩和法

図 2.3.5　即時報酬を最大化する行動 (5 × 5-GridWorld)

矢印：次の状態に遷移するときに得られる即時報酬を最大化する行動

色付きマス：最適行動と2番目の行動の差が最大報酬の1/10以上の状態

⇒これが多いほど学習効率が良い

2.3.5 計算量の考察

2.3.5 報酬関数を求めるステップ

(1)状態遷移確率と最適行動を求める
(2)(1)を用いて目的関数と制約条件を求める
(3)線形計画法を用いて報酬関数を求める

それぞれの計算量は？

2.3.5 報酬関数を求めるステップ

(1)状態遷移確率と最適行動を求める(強化学習)
(2)(1)を用いて目的関数と制約条件を求める
(3)線形計画法を用いて報酬関数を求める

(1)の強化学習の計算量は$O(n_s \times n_a)$
n_s : 状態数
n_a : 行動数
(2)は$O(n_s^2)$

2.3.5 各アルゴリズムの計算量

Ng　　: $O(n_s^2 \times n_a^2)$

GAIRL : $O(n_s^2 \times n_a)$

Abbeel : $O(n_s^2 \times n_a^2)$(本間違ってない？)

GAIRLはGAを用いることで高効率化

2.3.6 まとめ

2.3.6 まとめ

本に書いてないけど, ビッグデータが必要な強化学習のひとつということで, 強化学習の強みをつぶしているようにしか思えないという感想でした.
ただ, 何かの模倣をしたい, というモチベーションの分野であれば有効かなぁとも思います.

Ex)人間ロボット, 自動運転 etc...