

解 説

線形可解マルコフ決定過程を用いた順・逆強化学習

内 部 英 治

株式会社国際電気通信基礎技術研究所脳情報研究所 プレインロボットインタフェース研究室*

Forward and Inverse Reinforcement Learning Based on Linearly Solvable Markov Decision Processes

Eiji Uchibe

Dept. of Brain Robot Interface, ATR Computational Neuroscience Labs.*

概要

連続時間系での順強化学習では最適制御則を求めるためには非線形の偏微分方程式であるハミルトン・ヤコビ・ベルマン方程式を解く必要があるが、ほとんどの場合解析的に解くことは困難である。離散時間系の定式化の場合も同様で、このことが強化学習を実問題に適用する上での問題の一つとなっている。近年、線形可解マルコフ決定過程と呼ばれる問題のクラスが提案され、目的関数を規定する即時コスト関数の一部をカルバックライブラーダイバージェンスによって表現することで、ハミルトン・ヤコビ・ベルマン方程式を線形化できることが示された。この解説では、線形可解マルコフ決定過程のロボット制御への適用について説明し、この枠組みにおいて、学習済みの制御則の合成理論に基づいた学習の高速化と観測された状態行動系列からコスト関数を推定する逆強化学習問題が実現できることを紹介する。

1. は じ め に

強化学習は環境との相互作用を通して最適な制御則を獲得するための手法で、ヒトやラットなどの学習過程を記述する計算モデルとしても注目されている。ただし実用化には、(1) タスクごとに最適制御則を一から学習する必要があり、学習済みの制御則を再利用する理論的枠組みがない、(2) タスクに適したコスト関数をどのように準備すればよいかの指針がない、といった点を解決する必要がある。

これに対し、近年開発された線形可解マルコフ決定過程 (Linearly solvable Markov Decision Process: LMDP)^{1,2)} と呼ばれる定式化が上記二つの問題に有効であることが示された。通常の強化学習や最適制御ではマルコフ決定過程のもとで導出される非線形のハミルトン・ヤコビ・ベルマン (Hamilton-Jacobi-Bellman: HJB) 方程式を解く必要があるが、LMDP では即時コスト関数と環境やロボットのダイナミクスを巧妙に制限することで HJB 方程式を線形化する。これにより最適制御則を求めることが簡単化され、CG 分野におけるアニメーション生成や携帯電話の基地局に対する

チャネル割り当て問題、ロボット制御など幾つかの分野で利用されるようになってきている³⁻⁶⁾。また LMDP はカルバックライブラー (Kullback Leibler: KL) 制御⁷⁾や経路積分に基づく強化学習^{8,9)}と密接な関係があり、これらはすべて HJB 方程式の線形化に基づいている。特に最適制御問題を機械学習におけるグラフィカルモデルの推論といった確率推論問題に帰着できることが示され¹⁰⁾、最適制御理論と強化学習や機械学習をリンクさせる理論として注目されている。そこで本解説では松原による解説¹¹⁾でカバーされていない連続時間系での LMDP を紹介し、実ロボットへの応用を述べる。

3 節では (1) の問題を解決するための LMDP の枠組みにおける制御則の合成理論を紹介する。これは線形微分方程式の解の重ね合わせ理論をベースにしており、複数の学習済みの最適制御則から別の問題の厳密な最適制御則を導出することが可能になる。すなわち、終端コストのみが異なる複数のタスクに対する最適制御則が与えられ、新しいタスクが終端コストの重み付き線形和で記述されるとき、新しいタスクに対する HJB 方程式の解は学習済みの HJB 方程式の解の線形和で記述され、結果として新しいタスクの最適制御則も学

* 〒 619-0288 京都府相楽郡精華町光台二丁目 2 番地 2

習済みの最適制御則の重ね合わせで厳密に記述できる。LMDP を用いた制御則の合成は簡単な計算機シミュレーション上で有効性が示されている^{3,12)}が、本節では四脚ロボットの歩行課題に適用した結果¹³⁾を述べる。

4 節では (2) の問題を解決するために、LMDP と類似したコスト関数の制約に基づく方法である Dynamic Policy Programming (DPP)¹⁴⁾ の枠組みを用いたモデルフリー逆強化学習法¹⁵⁾を紹介する。逆強化学習^{16,17)}とは観察された行動が最適制御則から生成されたものと仮定し、それに対応するコスト関数を推定する逆問題で、模倣学習を実装するための強力な手段であると同時に、学習している生物の行動を解析するための有望なアプローチである。Dynamic Policy Programming のもとで単純化されたベルマン方程式が学習前後の確率的制御則の比の対数、コスト関数、状態価値関数によって記述できることを示す。学習前後の確率的制御則の比は密度比推定の手法¹⁸⁾によって直接推定することができ、正則化付き最小二乗法と組み合わせることでコスト関数と状態価値関数を同時に推定することが可能になる。実ロボットを用いた視覚ベースのナビゲーション課題に適用し、従来法よりも効率よくコスト関数を推定することができることを示す。

2. 線形可解マルコフ決定過程を用いたロボット制御

2.1 マルコフ決定過程

まず連続時間・連続状態行動系における最適制御を Todorov の方法²⁾に従って概説する。 $\mathcal{X}$ を状態空間、 $\mathcal{U}$ を連続行動空間とする。時刻 t における状態を $\mathbf{x}(t) \in \mathcal{X}$ 、ロボットが実行する行動を $\mathbf{u}(t) \in \mathcal{U}$ とする。このとき、ロボットは即時コスト $c(\mathbf{x}(t), \mathbf{u}(t))$ を受け取り、以下の確率微分方程式

$$d\mathbf{x} = \mathbf{a}(\mathbf{x})dt + \mathbf{B}(\mathbf{x})(\mathbf{u}dt + \sigma d\boldsymbol{\omega}), \quad (1)$$

にしたがって状態遷移する。ここで $\boldsymbol{\omega}$ はブラウンノイズ、 σ はスケーリングパラメータである。式 (1) は状態 $\mathbf{x}$ に関して非線形、行動 $\mathbf{u}$ に関して線形で、ノイズは $\mathbf{B}(\mathbf{x})$ を通して状態に影響する点に注意されたい。

制御器 $\pi(\mathbf{u} | \mathbf{x})$ は状態 $\mathbf{x}$ で行動 $\mathbf{u}$ を選択する条件付き確率として定義される。目的はロボットを目標状態 $\mathbf{x}_g \in \mathcal{X}_g \subseteq \mathcal{X}$ へ導く最適な制御器 π^* を発見することで、次の目的関数

$$V^\pi(\mathbf{x}, t) = \mathbb{E} \left[\int_t^{T_g} c(\mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau + g(\mathbf{x}(T_g)) \right].$$

ここで $c(\mathbf{x}, \mathbf{u})$ 、 $g(\mathbf{x})$ はそれぞれ即時コストと終端コスト、 T_g は評価時間である。 $V^\pi(\mathbf{x}, t)$ は cost-to-go 関

数、または価値関数と呼ばれる。最適価値関数は

$$V^*(\mathbf{x}, t) = \min_{\pi} V^\pi(\mathbf{x}, t),$$

と与えられる。一般に、最適価値関数は次の最適ハミルトン・ヤコビ・ベルマン (HJB) 方程式

$$-\frac{\partial V^*(\mathbf{x}, t)}{\partial t} = \min_{\mathbf{u}} \left(c(\mathbf{x}, \mathbf{u}) + \mathcal{L}^{(\mathbf{u})}[V] \right), \quad (2)$$

$$V^*(\mathbf{x}, T_g) = g(\mathbf{x}), \quad (3)$$

を満足する^{2,19)}。 $\mathcal{L}^{(\mathbf{u})}$ は線形微分演算子で

$$\mathcal{L}^{(\mathbf{u})}[V] = (\mathbf{a}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u})^\top \frac{\partial V}{\partial \mathbf{x}} + \frac{1}{2} \text{tr} \left(\mathbf{B}\mathbf{B}^\top \frac{\partial^2 V}{\partial \mathbf{x}^2} \right),$$

と定義される。最適制御理論やモデルベース強化学習では式 (1) が既知として、HJB 方程式を解くことが中心的課題となる。ただし HJB 方程式は価値関数に関して非線形であり、一般に解くのが困難である。

2.2 線形可解マルコフ決定過程

HJB 方程式が厳密に解ける問題のクラスとして知られているのが線形二次ガウス型レギュレータ (Linear Quadratic Gaussian: LQG) で、このとき HJB 方程式はリッカチ微分方程式を解く問題に帰着し、最適制御則は線形フィードバックの形で与えられる。これに対し線形可解マルコフ決定過程 (LMDP) では、コスト関数の行動依存コストの部分だけを二次形式にした

$$c(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + \frac{1}{2\sigma^2} \mathbf{u}^\top \mathbf{u}, \quad (4)$$

を用いる。 $q(\mathbf{x})$ は非負の状態依存コスト関数である。このとき式 (2) の右辺は解析的に最小化でき、式 (2)、(3) はそれぞれ

$$-\frac{\partial Z(\mathbf{x}, t)}{\partial t} = \mathcal{L}^{(0)}[Z] - q(\mathbf{x})Z(\mathbf{x}, t), \quad (5)$$

$$Z(\mathbf{x}, T_g) = \exp(-g(\mathbf{x})), \quad (6)$$

となる。ここで $Z(\mathbf{x}, t)$ は好適度関数 (desirability function)

$$Z(\mathbf{x}, t) = \exp(-V^*(\mathbf{x}, t)), \quad (7)$$

と呼ばれる。定義より $Z(\mathbf{x}, t) \in [0, 1]$ である。

離散時間系でも同様に HJB 方程式に対応するベルマン方程式は

$$Z(\mathbf{x}) = \exp(-hq(\mathbf{x}))\mathcal{G}[Z](\mathbf{x}), \quad (8)$$

$$Z(\mathbf{x}_g) = \exp(-g(\mathbf{x}_g)), \quad \mathbf{x}_g \in \mathcal{X}_g, \quad (9)$$

と簡単化される。 $\mathcal{G}$ は積分演算子

$$\mathcal{G}[f](\mathbf{x}) = \int p^0(\mathbf{x}' | \mathbf{x}) f(\mathbf{x}') d\mathbf{x}', \quad (10)$$

で、状態遷移確率 $p^0(\mathbf{x}' | \mathbf{x})$ は式 (1) を時間刻み幅 h で離散化して得られるガウス分布

$$p^{\mathbf{u}_k}(\mathbf{x}_{k+1} | \mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k+1} | \boldsymbol{\mu}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{x}_k, h\boldsymbol{\Sigma}(\mathbf{x})), \quad (11)$$

である。ここで $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ は平均 $\boldsymbol{\mu}$ 、共分散行列 $\boldsymbol{\Sigma}$ のガウス分布で、

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{x}, \mathbf{u}) &= h(\mathbf{a}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u}), \\ \boldsymbol{\Sigma}(\mathbf{x}) &= \sigma \mathbf{B}(\mathbf{x})^\top \mathbf{B}(\mathbf{x}), \end{aligned}$$

である。表記を簡単にするため、 $\mathbf{x}_k = \mathbf{x}(hk)$, $\mathbf{u}_k = \mathbf{u}(hk)$ としている。導出は文献^{1, 11)}を参照されたい。式 (11) の遷移確率は $\mathbf{u} = \mathbf{0}$ のとき制御されていないダイナミクス (uncontrolled dynamics) と呼ばれ、それ以外は制御されたダイナミクス (controlled dynamics) と呼ばれる。

式 (5), (8) は価値関数を指数変換した好適度関数に関して線形方程式[†]である。ただし境界条件がなければ自明解 (すべての $\mathbf{x}$ に対して $z(\mathbf{x}) \equiv 0$) に対して成立することに注意されたい。この枠組みを線形可解マルコフ決定過程 (Linearly solvable Markov Decision Process) と呼び、以降 LMDP と呼ぶ。LMDP の枠組みでは、最適制御則は連続時間、離散時間系ともに

$$p^{\mathbf{u}^*}(\mathbf{x}' | \mathbf{x}) = \frac{p^0(\mathbf{x}' | \mathbf{x})Z(\mathbf{x}')}{\mathcal{G}[Z](\mathbf{x})}, \quad (12)$$

と与えられる。特に式 (1) のダイナミクスのもとでは最適制御則は

$$\mathbf{u}^*(\mathbf{x}, t) = \sigma^2 \mathbf{B}(\mathbf{x})^\top \frac{\partial \ln Z(\mathbf{x}, t)}{\partial \mathbf{x}}, \quad (13)$$

となる。

2.3 制御器の学習

最適制御則 (13) を求めるには式 (5), (6) または (8), (9) を満足する好適度関数 $Z(\mathbf{x}, t)$ と $\mathbf{B}(\mathbf{x})$ を求める必要がある。ここでは何らかのモデル学習法^{20, 21)}によって既に式 (1) のダイナミクス $\mathbf{a}(\mathbf{x})$, $\mathbf{B}(\mathbf{x})$ が推定されていると仮定し、それをもとに $Z(\mathbf{x}, t)$ を学習する方法を述べる。好適度関数を N_z 個の基底関数を用いた関数近似器

$$Z(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \sum_{j=1}^{N_z} w_j f(\mathbf{x}, \boldsymbol{\theta}_j) = \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}), \quad (14)$$

[†] 誌面の都合上、有限時間区間の積算コスト最小問題についてのみ示すが、平均コスト最小問題についても同様に線形化できる。ただし強化学習で広く使われている割引積算コスト最小化問題に対しては線形化できないことに注意されたい。

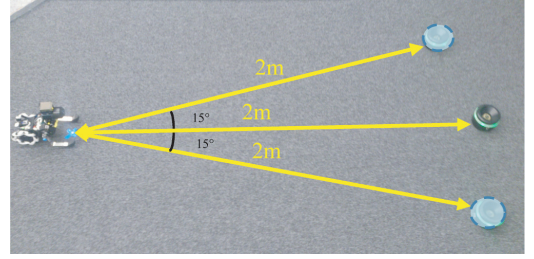


Fig. 1 Initial position of the robot and the battery packs in the test phase.

を用いて表現する。ここで w_i は重み、 $f(\mathbf{x}, \boldsymbol{\theta}_i)$ は放射状基底関数

$$f(\mathbf{x}; \boldsymbol{\theta}_j) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_j)^\top \mathbf{S}_j(\mathbf{x} - \mathbf{m}_j)\right),$$

$\boldsymbol{\theta}_j = \{\mathbf{m}_j, \mathbf{S}_j\}$ で、 $\mathbf{m}_j$ は中心位置、 $\mathbf{S}_j$ は精度行列である。また $\mathbf{w}, \mathbf{f}$ は重みと基底関数をまとめたベクトルである。このとき積分演算子 (10) は解析的に計算できる。式 (8), (9) を満足するように $\{w_j, \boldsymbol{\theta}_j\}_{j=1}^{N_z}$ を学習する。詳細は文献⁵⁾を参照されたい。

2.4 実験

LMDP の実環境における最適制御則の性能を評価するために、Fig. 1 に示す環境で車輪型移動ロボットを用いた視覚ナビゲーション課題を実施した。環境にはランドマークとして3色LEDを上部に取り付けた電池パックがあり、ロボットは搭載されたパン・チルトカメラから得られる画像情報をもとにランドマークまで移動する。このロボットは著者が使用していたロボット「サイバーローデント」²²⁾の後継機で、外部電池からの充電行動とUSB接続による交配機能を装備している。

Fig. 2 に制御アーキテクチャを示す。主に三つの制御器から構成され、一つは視覚サーボ制御器で、ランドマークをできるだけ視野内に捉えるようパン・チルトカメラを制御する。二番目は探査行動制御器で、ランドマークを見失った場合に実行される。三番目はナビゲーション制御器で、左右の車輪を制御し環境内を動き回る。この制御器をLMDPによって実装する。行動は両車輪への目標回転角速度 $\mathbf{u} = [u_{\text{left}}, u_{\text{right}}]^\top$ で、状態は画像上でのランドマークの位置 x_{cx}, x_{cy} と幅と高さ x_{ax}, x_{ay} 、さらに視覚サーボ制御器によって制御されたパン・チルトカメラの角度 $x_{\text{tilt}}, x_{\text{pan}}$ をまとめた6次元ベクトル $\mathbf{x} = [x_{cx}, x_{cy}, x_{ax}, x_{ay}, x_{\text{tilt}}, x_{\text{pan}}]^\top$ である。これらの状態・行動は

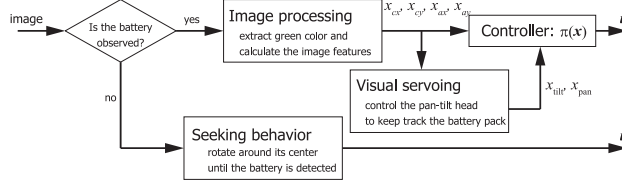


Fig. 2 Control architecture.

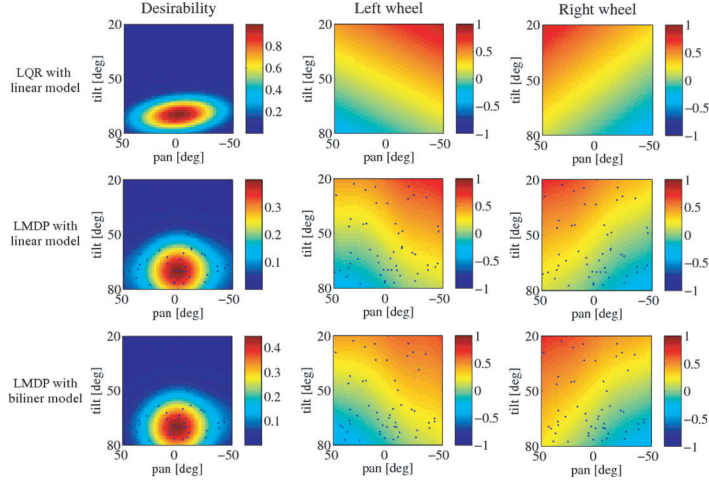


Fig. 3 The obtained desirability functions and the optimal policies in the navigation task. LQR with the linear model is at the top, LMDP with the linear model in the middle, LMDP with the bilinear model at the bottom, $Z(\mathbf{x})$ on the left, $u_{\text{left}}^*(\mathbf{x})$ on the center, $u_{\text{right}}^*(\mathbf{x})$ on the right.

$$-1 \leq x_{cx}, x_{cy}, x_{\text{tilt}}, x_{\text{pan}} \leq 1,$$

$$0 \leq x_{ax}, x_{ay} \leq 1, \quad -1 \leq u_{\text{left}}, u_{\text{right}} \leq 1,$$

とスケールされている。この実験では式 (1) のダイナミクス $\mathbf{a}(\mathbf{x})$, $\mathbf{B}(\mathbf{x})$ を線形モデルまたは双線形モデルによって表現し、パラメータを最小二乗法で同定した。状態依存コストとして目標状態 $\mathbf{x}_g$ で最小となるような二次形式コストと逆ガウシアン型コスト

$$q_1(\mathbf{x}) = \alpha (\mathbf{x} - \mathbf{x}_g)^\top \Sigma_{\text{cost}}^{-1} (\mathbf{x} - \mathbf{x}_g),$$

$$q_2(\mathbf{x}) = \alpha (1 - \exp(-q_1(\mathbf{x}))),$$

を準備した。 α はコストのスケール定数である。またダイナミクス (1) の近似モデルとして線形モデル、双線形モデルを準備した。ここで線形モデルと二次形式コスト $q_1(\mathbf{x})$ を用いた場合、LMDP は最適レギュレータ (Linear Quadratic Regulator, LQR) と本質的に等価であることに注意されたい。

Fig. 3 に二次形式コスト関数 $q_1(\mathbf{x})$ を用いて獲得された好適度関数と最適制御則の典型例を示す。それぞ

れの図において x, y 軸はパン・チルトカメラの関節角 $x_{\text{pan}}, x_{\text{tilt}}$ で、それ以外の 4 つの状態量は目標状態の値に設定している。また中央と下の行の図中の青点は好適度関数の近似に用いた基底関数の中心位置 $\mathbf{m}_i$ である。LQR で獲得された最適制御則は線形状態フィードバックであるが、LMDP では好適度関数を式 (14) のような関数によって近似しているため、図のような非線形フィードバックになっている。また、適合度の近似については式 (1) を線形モデルで表現した場合と双線形モデルで表現した場合に大きな差は見られなかった。

次に異なるコスト関数やダイナミクスの近似モデルによって得られた制御則を比較する。ただしコスト関数が異なるため、総コストのかわりに現在の状態と目標状態の差の L1 ノルムを評価として用いる。結果を Fig. 4 に示す。すべての制御器が約 10 秒ほどでロボットを目標状態に到達させている。目標状態近傍での挙動は LQR と LMDP で異なり、LQR による線形状態フィードバックでは目標状態近傍での制御量が小さく車輪と床との摩擦に打ち勝つことができなかったのに

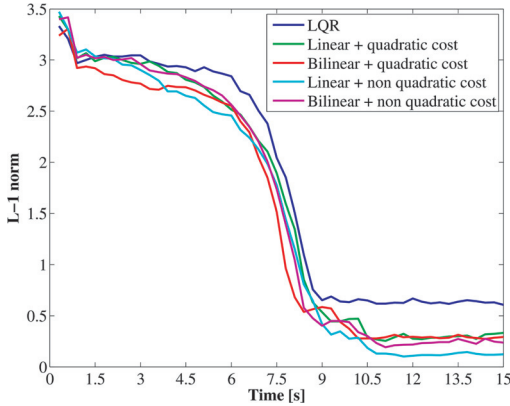


Fig. 4 Trajectories of the L-1 norm between the current and goal states.

対し, LMDP では非線形フィードバックにより目標状態にさらに近づけたことが確認できる。

3. 線形可解マルコフ決定過程を用いた制御則の重ね合わせ

この節では LMDP を用いる利点の一つである制御則の重ね合わせについて説明する。これによってすでに学習済みの最適制御則を組み合わせて新しい最適制御則を構成することができ、最初から学習するよりも大幅に時間を短縮できる。

3.1 制御則の重ね合わせ

はじめに述べたとおり, LMDP における制御則の合成は線形方程式の解の重ね合わせ理論に基づいている。いま, ロボットのダイナミクスが式 (1) で表現でき, N 種類のタスクを達成するための最適制御則 $\mathbf{u}_i$ ($i = 1, \dots, N$) が 2 節で紹介した LMDP によって得られていると仮定する。以降, これらのタスクを要素タスクと呼ぶことにする。本節の主題は学習済みの最適制御則を用いて新しいタスクに対する最適制御則を求めることである。すなわち, 新しいタスクに対する最適制御則が学習済みの制御則の重み付き和で表現できることを示す。

線形方程式の重ね合わせ理論を適用するためには, 式 (5), (8) の方程式がすべての要素タスクと新しいタスクで同一である必要がある。よって即時状態依存コスト関数 $q(\mathbf{x})$ はすべてのタスクで共通の要素だけを考慮したものになり, タスクの違いは終端コスト関数によって記述する必要がある。要素タスクの終端コストを $g_i(\mathbf{x})$, 新しいタスクの終端コストを $g'(\mathbf{x})$ とし,

$$\exp(-g'(\mathbf{x})) = \sum_{i=1}^N w_i \exp(-g_i(\mathbf{x})), \quad (15)$$

と与えられるとする。式 (15) はガウス関数を基底関数とする RBF ネットワークであり, 要素タスクが十分に多ければ任意の終端コストを表すことができる。 w_i は結合重みである。このとき, 新しいタスクに対する好適度関数は重ねあわせの理論より

$$Z'(\mathbf{x}, t) = \sum_{i=1}^N w_i Z_i(\mathbf{x}, t), \quad (16)$$

と与えられる。この新しいタスクは要素タスクの合成によって表現されるため, 以降合成タスクと呼ぶことにする。対応する合成タスクの最適制御則は

$$\begin{aligned} \mathbf{u}^*(\mathbf{x}, t) &= \sigma^2 \mathbf{B}(\mathbf{x})^\top \frac{\partial \ln Z'(\mathbf{x}, t)}{\partial \mathbf{x}} \\ &= \sum_{i=1}^N m_i(\mathbf{x}, t) \mathbf{u}_i^*(\mathbf{x}, t), \end{aligned} \quad (17)$$

となる。ここで制御則の混合係数は時間 t と状態 $\mathbf{x}$ の関数で

$$m_i(\mathbf{x}, t) = \frac{w_i Z_i(\mathbf{x}, t)}{\sum_j w_j Z_j(\mathbf{x}, t)},$$

と定義する。これより合成タスクの制御則も要素タスクの制御則の重み付き和で表現されたことになるが, 重みは状態と時間の関数になっていることに注意されたい。

3.2 好適度関数のモデルフリー学習

2 節では環境のダイナミクスを明示的に学習した後で簡化した HJB 方程式から好適度関数を求めたが, ここではダイナミクスを明示的には学習しないモデルフリーの方法を述べる。これは最小二乗法に基づく強化学習法^{23, 24)}の考えを適用したもので, 式 (14) で表現した適合度関数の近似器の重みパラメータ w_i を学習するものである。

状態の系列 $\mathbf{x}_0, \dots, \mathbf{x}_{N_{\text{step}}}$, $hN_{\text{step}} = T_g$ に対して, 行列 $\mathbf{A}$ とベクトル $\mathbf{b}$ をエピソードごとに

$$\mathbf{A} \leftarrow \mathbf{A} + \sum_{k=0}^{N_{\text{step}}} \phi_k (\phi_k - \exp(-q(\mathbf{x}_k)) \phi_{k+1})^\top, \quad (18)$$

$$\mathbf{b} \leftarrow \mathbf{b} + \phi_{N_{\text{step}}} \exp(-g(\mathbf{x}_{N_{\text{step}}})),$$

と更新する。ここで $\phi_k = \phi(\mathbf{x}_k, hk)$ で, $\phi_{N_{\text{steps}}+1} = \mathbf{0}$ とする。これより好適度関数のパラメータを $\mathbf{w} = \mathbf{A}^{-1} \mathbf{b}$ と決定する。通常の最小二乗法に基づく強化学

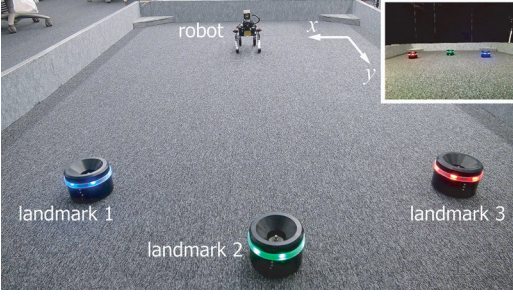


Fig. 5 The quadruped robot and three landmarks in the experimental field. The top right image represents the view from the robot.

習法と比較すると、式 (18) の $\exp(-q(\mathbf{x}_k))$ は状態依存の割引率に相当する。 $\mathbf{A}$ はすべての要素タスク間で同一であるのに対し、 $\mathbf{b}$ は要素タスクごとに計算する必要があることに注意されたい。

3.3 四脚ロボットの歩行課題

モデルフリーの適合度学習と制御則の合成の有効性を検証するために Fig. 5 に示す各脚が 2 関節である四脚ロボットを用いた視覚誘導型の歩行課題を実施した。ただしロボットの構造を単純にするためにサーボモータを腰関節につけ、膝関節は板ばねを用いた受動関節になっている。また 2.4 節のロボットと同一のパン・チルトカメラを搭載している。環境中には三つの LED 付きランドマークがロボットの前に置かれている。プリミティブタスクはスタート地点から一つランドマークへの移動である。

Fig. 6 に制御アーキテクチャを示す。状態ベクトル $\mathbf{x} = [\theta_{\text{hip}}^T, \mathbf{p}^T]^T$ は 6 つの要素からなり、 $\theta_{\text{hip}} \in \mathbb{R}^4$ は現在の腰関節の角度、 $\mathbf{p} = [x, y]^T$ はグローバル座標系でのロボットの位置である。各腰関節の目標角度 $\theta_{\text{hip},d} \in \mathbb{R}^4$ は

$$\theta_{\text{hip},d} = \mathbf{u} + \theta_{\text{CPG}}, \quad (19)$$

と与えられる。ここで $\mathbf{u}$ は強化学習モジュールから計算される行動、 $\theta_{\text{CPG}} \in \mathbb{R}^4$ は Central Pattern Generator (CPG) からの出力である。CPG の役割は特に学習の初期段階でロボットの転倒を防止しつつ、効率的に学習データを収集するのに重要で、実装として修正 Hopf 振動子²⁵⁾を用いた。

要素タスクとしてはスタート地点から赤・青・緑のランドマークの場所までの歩行で、即時コストは全ての状態で $q(\mathbf{x}) = 0.95$ とする定数を用い、終端コストは Fig. 7 に示すような目標状態が最小となる二次形式コスト $q_1(\mathbf{x})$ を用いる。一方、合成タスクは青と緑の

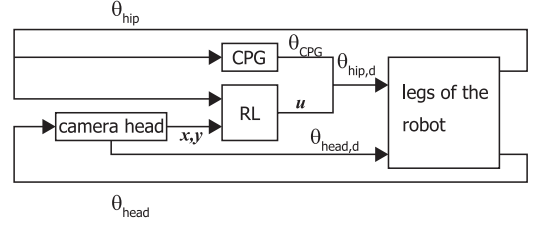


Fig. 6 Control architecture.

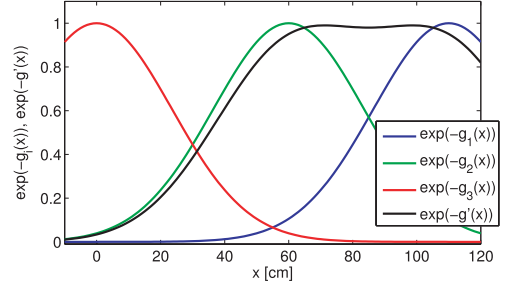


Fig. 7 Three terminal cost functions of primitive tasks (blue, green, and red) and that of the composite task (black).

ランドマークの中間地点に移動することで、対応する終端コストを Fig. 7 の黒線で示す。式 (15) の終端コストの結合重みは $w_1 = w_2 = 0.86, w_3 = 0$ である。要素タスクの数が少ないため、合成タスクの終端コストの形状は要素タスクの終端コストの形状と異なっていることに注意されたい。

3.4 実験結果

式 (17) によって三つの要素タスクの制御則を合成して得られる最適制御則を評価する。比較のために以下の制御則を準備した。

1. **composite policy $\mathbf{u}'_z$** : 式 (17) によって合成された制御則。
2. **composite policy with additional learning $\mathbf{u}'^*_z$** : 合成された制御則 $\mathbf{u}'_z$ を Fig. 6 の CPG からの出力と置き換え、さらに追加学習を LMDP によって実施して得られる制御則。
3. **optimal policy learned from scratch $\mathbf{u}^*_z$** : 通常の LMDP によって一から LMDP によって学習することで得られる制御則。
4. **value-weighted policy $\mathbf{u}'_v$** : 好適度関数ではなく価値関数を重み付き和によって合成した価値関数から導出された制御則。

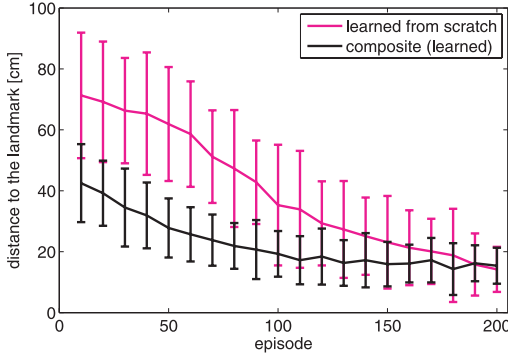


Fig. 8 Learning curves of composite tasks. The error bars represent the standard deviation of 10 experimental runs.

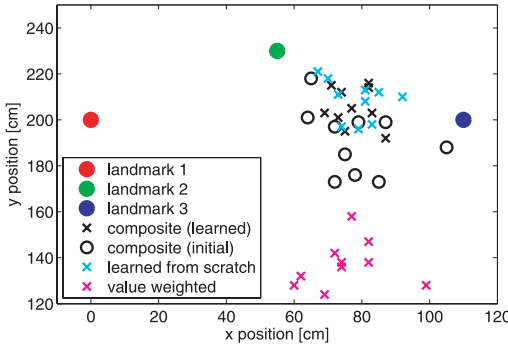


Fig. 9 Comparison among the composite policy, the optimal policy learned from the composite cost function, and the composite policy designed in the value function space.

ここで $\mathbf{u}_z^*$ と $\mathbf{u}_z^*$ の学習には同じ終端コスト関数 g' を用いている。

2., 3. の学習速度を比較するために, 1 エピソード終了時点でのロボットの位置と目標位置と距離をプロットしたものを Fig. 8 に示す. ここで黒線の追加学習による結果の 10 エピソード時点での性能が 1. の合成された制御則の性能であることに注意されたい. これより合成によって得られ制御則 $\mathbf{u}_z^*$ は一から学習した制御則 $\mathbf{u}_z^*$ とは異なっていることが確認できる. これはロボットのダイナミクスが厳密には式 (1) には従っていないことと, 要素タスクの好適度関数の近似誤差によるものと考えられる. しかし合成された制御則 $\mathbf{u}_z^*$ をベースとして追加学習することで, 一から学習するよりも効率よく制御則を学習できていることがわかる.

Fig. 9 に各制御則によって実現されたロボットの最終位置を示す. シアの十字は $\mathbf{u}_z^*$ の結果で, 合成タ

スクにおける最適制御則の結果に相当する. 黒丸は $\mathbf{u}_z^*$ による最終位置で, その分布は $\mathbf{u}_z^*$ によって得られる分布とは異なっており, 合成した制御則は最適制御則には異なっていることが確認できる. 黒の十字は $\mathbf{u}_z^*$ による最終位置で, 追加学習によって最適制御則と同等の分布が得られている. 一方, 価値関数の線形和から得られる制御則 $\mathbf{u}_v^*$ によって得られた最終位置の分布は $\mathbf{u}_z^*$ によって得られた分布とは大きく異なっていることから, 価値関数の合成は有効でないことも確認できる.

4. Dynamic Policy Programming を用いた逆強化学習

この節ではコスト関数の制限が逆強化学習の不良設定性を緩和し, 最終的に密度比推定の問題に帰着することができることを示す. これにより順強化学習問題を繰り返し解くことなく, コスト関数を直接推定できることを示す.

4.1 逆強化学習のためのベルマン方程式

前節までと異なり, ここでは離散時間・割引コスト最小化問題として目的関数が定式化される場合, つまり価値関数が

$$V^\pi(\mathbf{x}) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k c(\mathbf{x}_{t+k}, \mathbf{u}_{t+k}) \mid \mathbf{x}_t = \mathbf{x} \right],$$

と与えられる場合を考える. ここで γ は割引率, P_T は状態遷移確率で, 式 (11) のガウス分布を任意の分布に一般化したものである. コスト関数を

$$c(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + \text{KL}(\pi(\cdot \mid \mathbf{x}) \parallel \pi_0(\cdot \mid \mathbf{x})), \quad (20)$$

と制限する. ここで $\text{KL}(\pi(\cdot \mid \mathbf{x}) \parallel \pi_0(\cdot \mid \mathbf{x}))$ は二つの確率的制御則の間の KL ダイバージェンスである. $\pi_0(\mathbf{u} \mid \mathbf{x})$ はベースライン制御則で, 探索のための初期制御則とみなすことができる. 二つの制御則が共にガウス分布のとき, KL ダイバージェンスは式 (4) の第 2 項のような $\mathbf{u}$ に関する二次形式になる. 逆強化学習の目的はコスト関数の一部である状態依存コスト $q(\mathbf{x})$ を推定することである.

式 (20) のコスト関数のもとでは, 2.2 節と同様にベルマン方程式を簡単化することができ, 最適価値関数 $V^*(\mathbf{x})$ と最適制御則 $\pi(\mathbf{u} \mid \mathbf{x})$ は

$$-\ln \frac{\pi(\mathbf{u} \mid \mathbf{x})}{\pi_0(\mathbf{u} \mid \mathbf{x})} = q(\mathbf{x}) - V^*(\mathbf{x}) + \gamma \int P_T(\mathbf{y} \mid \mathbf{x}, \mathbf{u}) V^*(\mathbf{y}) d\mathbf{y}, \quad (21)$$

を満足する．この関係式は Dynamic Policy Programming と呼ばれる手法¹⁴⁾によって最初に使用されたが、ここではこの関係式が逆強化学習に非常に有効であることを述べる．

ただし、逆強化学習は一般に不良設定問題であり、コスト関数を式 (20) の形に制限しても唯一には決定できない．具体的には、コスト関数を定数 C によって

$$q'(\mathbf{x}) = q(\mathbf{x}) + C, \quad (22)$$

と変更し、対応する状態価値関数を

$$V^{*'}(\mathbf{x}) = V^*(\mathbf{x}) + \frac{C}{1-\gamma},$$

と変更しても密度比 $\pi(\mathbf{u} | \mathbf{x})/\pi_0(\mathbf{u} | \mathbf{x})$ は同一となる．

4.2 密度比推定

紹介する逆強化学習法¹⁵⁾は二つの要素から構成される．一つは式 (21) の左辺の制御則の比率を推定する密度比推定¹⁸⁾で、もう一つは推定された密度比の対数を目標値として、コスト関数と状態価値関数を推定する最小二乗法である．

状態遷移に関する次の二種類のデータセット

$$\mathcal{D}^0 = \{(\mathbf{x}_j^0, \mathbf{u}_j^0, \mathbf{y}_j^0)\}_{j=1}^{N^0}, \quad \mathcal{D}^\pi = \{(\mathbf{x}_i^\pi, \mathbf{u}_i^\pi, \mathbf{y}_i^\pi)\}_{i=1}^{N^\pi},$$

が与えられたとする．ここで N^0 と N^π はそれぞれデータセット $\mathcal{D}^0$, $\mathcal{D}^\pi$ の要素数で、これらは

$$\begin{aligned} \mathbf{u}_j^0 &\sim \pi_0(\cdot | \mathbf{x}_j^0), & \mathbf{y}_j^0 &\sim P_T(\cdot | \mathbf{x}_j^0, \mathbf{u}_j^0), \\ \mathbf{u}_i^\pi &\sim \pi(\cdot | \mathbf{x}_i^\pi), & \mathbf{y}_i^\pi &\sim P_T(\cdot | \mathbf{x}_i^\pi, \mathbf{u}_i^\pi), \end{aligned}$$

と生成されている． $\mathcal{D}^0$ は学習前のベースライン制御則 π_0 を使って得られた状態遷移の集合、 $\mathcal{D}^\pi$ は最適制御則 π を使って得られた状態遷移の集合である． P_T が既知であれば $\mathbf{y}_j^0, \mathbf{y}_i^\pi$ は必要ないことに注意されたい．

次に密度比の対数 $\ln \pi(\mathbf{u} | \mathbf{x})/\pi_0(\mathbf{u} | \mathbf{x})$ をデータセット $\mathcal{D}^0$, $\mathcal{D}^\pi$ から直接推定する．次の二通りの分解が考えられる．

$$\ln \frac{\pi(\mathbf{u} | \mathbf{x})}{\pi_0(\mathbf{u} | \mathbf{x})} = \ln \pi(\mathbf{u} | \mathbf{x}) - \ln \pi_0(\mathbf{u} | \mathbf{x}), \quad (23)$$

$$= \ln \frac{\pi(\mathbf{x}, \mathbf{u})}{\pi_0(\mathbf{x}, \mathbf{u})} - \ln \frac{\pi(\mathbf{x})}{\pi_0(\mathbf{x})}. \quad (24)$$

最初の分解 (23) は条件付き確率密度の対数の差になっていて、ロボット制御のような制御則 π_0, π が明示的に計算できる場合に適している．データセット $\mathcal{D}^0, \mathcal{D}^\pi$ から π^0, π を推定する方法としては、Least Squares Conditional Density Estimation (LSCDE)¹⁸⁾ などがある．

二番目の分解 (24) は二つの密度比の対数の差になっ

ている．この分解の利点は $\pi(\mathbf{x}) = \pi_0(\mathbf{x})$ となる問題設定では、第 1 項 $\ln \pi(\mathbf{x})/\pi_0(\mathbf{x})$ の推定をする必要がないことである．密度比 $\pi(\mathbf{x})/\pi_0(\mathbf{x}), \pi(\mathbf{x}, \mathbf{u})/\pi_0(\mathbf{x}, \mathbf{u})$ を直接推定する方法としては、ロジスティック回帰を用いた LogReg 法がある¹⁸⁾．

4.3 コスト関数と価値関数の推定

いったん密度比 $\pi(\mathbf{u} | \mathbf{x})/\pi_0(\mathbf{u} | \mathbf{x})$ を推定した後、正則化付き最小二乗法によって状態依存コスト関数 $q(\mathbf{x})$ と価値関数 $V(\mathbf{x})$ を推定することができる．いま関数 $\hat{R}(\mathbf{x}, \mathbf{u})$ を $\hat{R}(\mathbf{x}, \mathbf{u}) \approx -\ln \frac{\pi(\mathbf{u} | \mathbf{x})}{\pi_0(\mathbf{u} | \mathbf{x})}$ とし、 $q(\mathbf{x})$ と $V^*(\mathbf{x})$ の線形関数近似器を

$$\hat{q}(\mathbf{x}) = \mathbf{w}_q^\top \boldsymbol{\psi}_q(\mathbf{x}), \quad \hat{V}^*(\mathbf{x}) = \mathbf{w}_V^\top \boldsymbol{\psi}_V(\mathbf{x}),$$

とする．ここで $\boldsymbol{\psi}_q(\mathbf{x}), \boldsymbol{\psi}_V(\mathbf{x})$ は基底関数ベクトル、 $\mathbf{w}_q$ と $\mathbf{w}_V$ は重みパラメータであり、目的関数

$$\begin{aligned} J(\mathbf{w}_q, \mathbf{w}_V) = & \frac{1}{2} \sum_{j=1}^{N^\pi} \left[\mathbf{w}_V^\top (\gamma \boldsymbol{\psi}_V(\mathbf{y}_j^\pi) - \boldsymbol{\psi}_V(\mathbf{x}_j^\pi)) \right. \\ & \left. + \mathbf{w}_q^\top \boldsymbol{\psi}_q(\mathbf{x}_j^\pi) - \hat{R}(\mathbf{x}_j^\pi, \mathbf{u}_j^\pi) \right]^2 \\ & + \lambda_w \|\mathbf{w}_q\|_1 + \lambda_w \|\mathbf{w}_V\|_1, \end{aligned}$$

を最小化することによって最適化される．ここで λ_w は正則化定数である．コスト関数が非負であるという条件は考慮していないことに注意されたい．非負条件を満足するコスト関数が必要であれば、式 (22) において $C = -\min_{\mathbf{x}} \hat{q}(\mathbf{x})$ とすれば良い．

4.4 数値シミュレーションによる検証

この手法を評価するために、倒立振り子の振り上げおよび安定化課題²⁶⁾を考える．振り子の状態は角度 θ と角速度 ω の 2 次元ベクトル $\mathbf{x} = [\theta, \omega]^\top$ で表現され、運動方程式は Kinjo et al.⁵⁾ の設定を用いた．振り子の振り上げ・安定化課題における逆強化学習の目的は、ある最適制御則によって与えられたスタート地点 $\mathbf{x}_0 = [\pi, 0]^\top$ からゴール地点 $\mathbf{x}_g = [0, 0]^\top$ までの状態の系列から、その最適制御則に対応するコスト関数を推定することである．学習前のデータセット $\mathcal{D}^0$ は Gauss 分布を使った単純な制御則 $\pi_0(\mathbf{u} | \mathbf{x}) = \mathcal{N}(\mathbf{u} | 0, \sigma_0^2)$ を使い、学習後のデータセット $\mathcal{D}^\pi$ は真のコスト関数を $q(\mathbf{x}) = 1 - \cos \theta$ として順強化学習によって計算した最適制御則を用いて収集した．手法としてはモデルベースの連続時間強化学習法²⁶⁾を用いた．

密度比の推定方法により、我々の手法にはいくつかのバリエーションがある．具体的には式 (23) の分解を使った LSCDE-IRL と式 (24) の分解を使った LogReg-IRL が我々の手法であり、従来法であるモデルベース MaxEnt-IRL²⁷⁾、モデルフリー RelEnt-IRL²⁸⁾、モデ

Table 1 Comparison among IRL methods.

手法	提案手法	MaxEnt IRL	RelEnt IRL	OptV
モデルフリーか?	モデルフリー	モデルベース	モデルフリー	モデルベース
データの単位	状態遷移	軌跡全体	軌跡全体	状態遷移
順強化学習が必要か	必要なし	必要あり	必要なし	必要なし

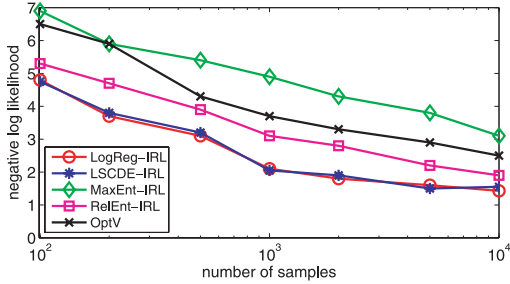


Fig. 10 Negative log likelihood in the inverted pendulum IRL task.

ルベース OptV²⁹⁾ と比較する。これらの違いを Table 1 に示す。モデルベース手法では状態遷移確率 P_T を必要とするが、これはデータセット D^0, D^π から推定し、積分の計算は Metropolis Hastings 法を用いた。

この実験では割引率 γ も未知とし、正則化パラメータも含めて交差検定法によって最適化した。基底関数 $\psi_q(\mathbf{x})$ はコスト関数についての事前知識に基づき設計するのが望ましいが、状態価値関数を表現する基底関数 $\psi_V(\mathbf{x})$ の設計は問題である。この実験では簡単のためガウスカーネルを用いる。カーネルの中心位置はデータセット D^π からランダムに 50 個選択した。

逆強化学習の不良設定性のため、真のコスト関数と推定されたコスト関数の差を直接比較するのは適切ではない。そこで、推定されたコスト関数から順強化学習により最適制御則 $\hat{\pi}(u|x)$ を推定し、テストデータセット D_{te} に対する負の対数尤度を推定されたコスト関数の評価値とする。Fig. 10 に各手法の負の対数尤度を比較したものを示す。横軸は各データセットのサンプル数 $|D^0|, |D^\pi|$ で、縦軸が負の対数尤度で小さいほど良い。LSCDE-IRL, LogReg-IRL は従来法よりも確率的制御則を正確に復元できていることが確認できる。この課題では LSCDE, LogReg による密度比推定の結果に大きな差が見られなかったため、類似したコスト関数と状態価値関数が推定された。RelEnt-IRL は $\mathbf{x}_0$ から $\mathbf{x}_g$ までの軌跡をサンプルとするので、サンプル数

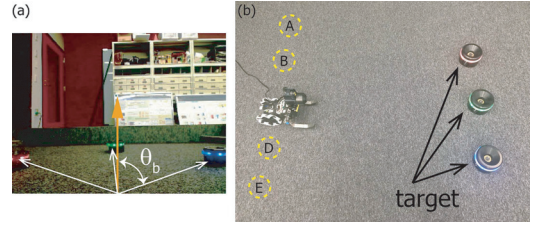


Fig. 11 Experimental field of the inverse reinforcement learning task. The current position of the robot is one of the starting positions, C.

が少ない場合は我々の手法よりも負の対数尤度が大きくなっているが、サンプル数が増加するにつれ両者はほとんど差がなくなっていった。OptV と MaxEnt-IRL はサンプル数が少ないときの性能が特によくなかった。

4.5 実ロボットのナビゲーション課題

次に Fig. 11 に示す実ロボットを使った視覚ベースのナビゲーション課題を使って検証する。図中の A, B, C, E の 4 箇所のスタート地点の 1 か所から目標地点まで実験者が遠隔操作により合計 100 回誘導して、データセット D^π を収集する。 D^0 も遠隔操作によってランダムにロボットを制御して収集する。ロボットは搭載されたカメラから 3 種類 (赤・緑・青) のランドマークに関連する画像特徴 $[x_r, y_r, N_r, x_g, y_g, N_g, x_b, y_b, N_b]$ を計測する。ここで x, y はランドマークの画像上での位置 N は画素数で、添え字の r, g, b はそれぞれ赤・緑・青のランドマークに対応する。この 6 個の特徴量を状態 $\mathbf{x}$ とする。

コスト関数の基底関数ベクトル $\psi_q(\mathbf{x})$ の要素は各ターゲットとの相対角度と大きさをもとに $\psi_q(\mathbf{x}) = [f_g(\theta_r), f_s(N_r), f_g(\theta_g), f_s(N_g), f_g(\theta_b), f_s(N_b)]^T$ と構成した。ここで f_g はガウス関数でそれぞれのターゲットが正面に見えるときに、 f_s はシグモイド関数でターゲットが大きく見えるときにそれぞれ 1 に近づくように設定した。一方、状態価値関数の基底関数ベクトル $\psi_V(\mathbf{x})$ の各要素はガウスカーネルを用い、中心位置は D^π からランダムに 200 個選択した。

この実験ではモデルベース手法で確率分布の正規化定数の計算を必要とする OptV, MaxEnt-IRL はコスト関数がうまく推定できなかったため、誌面の都合上 LSCDE-IRL, LogReg-IRL, RelEnt-IRL についてのみ比較した。Fig. 12 に各手法で推定されたコスト関数の重みを示す。全ての手法で、目標地点に対応する特徴量に関する重みを負に設定することで、ロボットが目標地点に到達したときに最小となるコスト関数を推

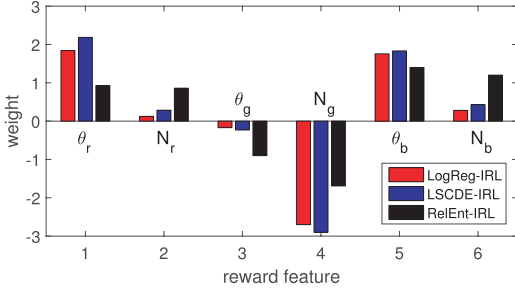


Fig. 12 Estimated weights of the cost function approximator.

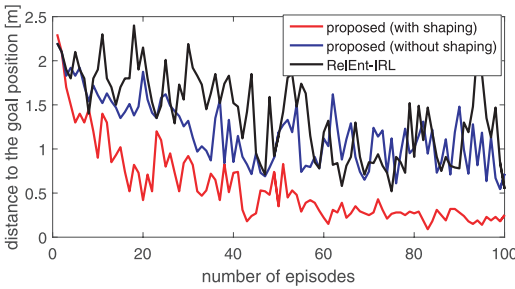


Fig. 13 Comparison of the learning speed with and without shaping rewards.

定していることがわかる。式 (22) を使ったコスト関数の補正はしていないため、推定されたコスト関数 $\hat{q}(\mathbf{x})$ は負となる場合もあることに注意されたい。

LogReg-IRL によって推定されたコスト関数を評価するために、逆強化学習の訓練データには含まれないスタート地点 D からのナビゲーションを Doya のモデルベース順強化学習法²⁶⁾ を用いて学習させた。とくに推定されたコスト関数 $\hat{q}(\mathbf{x})$ だけを用いた場合と、報酬関数のシェーピング理論¹⁶⁾ を用いたコスト $\hat{q}(\mathbf{x}) + \gamma \hat{V}^*(\mathbf{y}) - \hat{V}^*(\mathbf{x})$ を用いた場合の学習効率の違いを調査した。結果を Fig. 13 に示す。性能は各エピソード終了時点での目標地点との距離を用いた。シェーピングを用いた方がコスト関数だけを用いた場合よりも早く学習できていることが確認できる。RelEnt-IRL は状態価値を推定しないため、シェーピングは実現できないことに注意されたい。

5. まとめと今後の課題

本解説では非線形 HJB 方程式の線形化に基づく LMDP のロボット制御への応用について述べた。ダ

イナミクスと即時コスト関数に制限が必要であるため任意のタスクに適用できるわけではないが、HJB 方程式の線形化は非常に強力で、はじめに述べた二つの問題に対して新しい解法を与えてくれる。3 節で述べた制御則の合成理論は、要素タスクの即時コストが同一であるという条件が必要であった。そのため要素タスクの違いは終端コストによってのみ特徴づけられるという点が大きな問題である。この解説では合成した制御則を初期制御則として追加学習することで学習に要する時間を短縮したが、問題を時間的に部分問題に分割し、個々の部分問題に合成理論を適用する方法³⁰⁾ が提案されている。

4 節では Dynamic Policy Programming の枠組みの下でのモデルフリー逆強化学習法を紹介した。この手法は密度比の推定と、コスト関数と状態価値関数を推定する正則化付き最小二乗法によって効率よく計算できる。また、データセットは状態遷移の組から構成されるので、状態の軌跡全体を単位とする手法と比べるとデータ収集の点からも利点がある。また推定された状態価値関数からシェーピング報酬を作ることで、順強化学習の学習効率を改善することができた。Dynamic Policy Programming のかわりに LMDP における単純化されたベルマン方程式を用いて 4 節のような逆強化学習法を構成することも可能である。このとき密度比は学習前後の状態遷移確率になり、行動を直接観測できない実験データに対しても適用可能である。

誌面の都合上説明しなかったが、その他に興味深いテーマとして制御と推論の双対性が LMDP では拡張されていることである^{10, 31)}。これは良く知られた LQG とカルマンフィルタの双対性を拡張したもので、これを利用した新しいアルゴリズムの開発が進められている。LMDP や KL 制御、経路積分に基づく強化学習はその有効性にもかかわらず、国内ではあまり研究されていないが、本解説によって LMDP の研究者が増えることを期待する。

謝辞

本解説記事で紹介した内容は著者が沖縄科学技術大学院大学に在籍時の結果で、特に 2 節の内容は当時学生であった金城健さんと沖縄科学技術大学院大学の銅谷賢治教授との共同研究の成果である。ここに感謝の意を表します。本研究は MEXT/JSPS 科研費 24500249, 26120727 の助成を受けたものです。

参 考 文 献

- 1) Todorov, E. (2007): Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems* 19, pp.1369–1376.
- 2) Todorov, E. (2009): Efficient computation of optimal actions, *PNAS*, Vol.106, No.28, pp.11478–11483.
- 3) da Silva, M., Durand, F., Popović, J. (2009): Linear Bellman combination for control of character animation, *ACM Transactions on Graphics*, Vol.28, No.3.
- 4) Li, A., Schrater, P.R. (2013): Efficient Learning in Linearly Solvable MDP Models. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence*, pp.248–253.
- 5) Kinjo, K., Uchibe, E., Doya, K. (2013): Evaluation of linearly solvable Markov decision process with dynamic model learning in a mobile robot navigation task, *Frontiers in Neurobotics*, Vol.7, No.7.
- 6) Burdelis, M.A.P., Ikeda, K. (2014): Estimating passive dynamics distributions in linearly solvable Markov decision processes from measured immediate costs in reinforcement learning problems, *SICE Journal of Control, Measurement, and System Integration*, Vol.7, No.1, pp.48–54.
- 7) Kappen, H.J., Gómez, V., Opper, M. (2012): Optimal control as a graphical model inference problem, *Machine Learning*, Vol.87, No.2, pp.159–182.
- 8) Kappen, H. (2005): Linear theory for control of nonlinear stochastic systems, *Physical Review Letters*, Vol.95, No.20.
- 9) Theodorou, E., Buchli, J., Schaal, S. (2010): A generalized path integral control approach to reinforcement learning, *Journal of Machine Learning Research*, Vol.11, pp.3137–3181.
- 10) Theodorou, E., Todorov, E. (2012): Relative entropy and free energy dualities: Connections to Path Integral and KL control. In *Proc. of the 51st IEEE Conference on Decision and Control*, pp.1466–1473.
- 11) 松原崇充 (2015) : 確率最適制御の最近の動向—確率推論による解法, システム/制御/情報, Vol.59, No.10, pp.369–374.
- 12) Todorov, E. (2009): Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems* 22, pp.1856–1864.
- 13) Uchibe, E., Doya, K. (2014): Combining learned controllers to achieve new goals based on linearly solvable MDPs. In *Proc. of IEEE International Conference on Robotics and Automation*, No.1, pp.5252–5259.
- 14) Gheshlaghi Azar, M., Gómez, V., Kappen, H.J. (2012): Dynamic policy programming, *Journal of Machine Learning Research*, Vol.13, pp.3207–3245.
- 15) Uchibe, E., Doya, K. (2014): Inverse Reinforcement Learning Using Dynamic Policy Programming. In *Proc. of IEEE International Conference on Development and Learning and Epigenetic Robotics*, pp.222–228.
- 16) Ng, A.Y., Russell, S. (2000): Algorithms for Inverse Reinforcement Learning. In *Proc. of the 17th International Conference on Machine Learning*, pp.663–670.
- 17) Abbeel, P., Ng, A.Y. (2004): Apprenticeship learning via inverse reinforcement learning. In *Proc. of the 21st International Conference on Machine Learning*.
- 18) Sugiyama, M., Suzuki, T., Kanamori, T. (2012): Density ratio estimation in machine learning, Cambridge University Press.
- 19) Fleming, W.H., Mete Soner, H. (2006): *Controlled Markov Processes and Viscosity Solutions*, Springer, second edition.
- 20) Nguyen-Tuong, D., Peters, J. (2011): Model learning for robot control: a survey, *Cognitive Processing*, Vol.12, No.4, pp.319–340.
- 21) Sigaud, O., Salaün, C., Padois, V. (2011): On-line regression algorithms for learning mechanical models of robots: A survey, *Robotics and Autonomous Systems*, Vol.59, No.12, pp.1115–1129.
- 22) 内部英治, 銅谷賢治 (2007) : サイバーローデントプロジェクト, 日本神経回路学会誌, Vol.14, No.4, pp.293–304.
- 23) Boyan, J.A. (2002): Technical update: Least-squares temporal difference learning, *Machine Learning*, Vol.49, No.2/3, pp.233–246.
- 24) Lagoudakis, M.G., Parr, R. (2003): Least-squares policy iteration, *Journal of Machine Learning Research*, Vol.4, pp.1107–1149.
- 25) Righetti, L., Ijspeert, A.J. (2008): Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proc. of IEEE International Conference on Robotics and Automation*, pp.819–824.
- 26) Doya, K. (2000): Reinforcement learning in continuous time and space, *Neural Computation*, Vol.12, pp.219–245.
- 27) Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K. (2008): Maximum Entropy Inverse Reinforcement Learning. In *Proc. of the 23rd AAAI Conference on Artificial Intelligence*.
- 28) Boularias, A., Kober, J., Peters, J. (2011):

- Relative Entropy Inverse Reinforcement Learning. In Proc. of the 14th International Conference on Artificial Intelligence and Statistics.
- 29) Dvijotham, K., Todorov, E. (2010): Inverse Optimal Control with Linearly Solvable MDPs. In Proc. of the 27th International Conference on Machine Learning, pp.335–342.
- 30) Matsubara, T., Asakura, T., Sugimoto, K. (2015): Dynamic linear Bellman combination of optimal policies for solving new tasks, IE-ICE Trans. Fundamentals, Vol.E.98-A, No.10, pp.2187–2190.
- 31) Theodorou, E. (2015): Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations, Entropy, Vol.17, pp.3352–3375.