4D1-OS-37c-2

確率変数変換の学習によるノンパラメトリックな確率的方策の獲得

横山 裕樹 *1　　岡田 浩之 *2
Hiroki Yokoyama　　Hiroyuki Okada

*1*2玉川大学脳科学研究所
Tamagawa University Brain Science Institute

For reinforcement learning algorithms with non-parametric policies, we propose a method to learn random variable transformation whose resultant distribution maximizes a particular criterion. In our approach, the policy PDF is not explicitly represented, whereas the gradients of that can be directly estimated.

1. はじめに

不確実性を含む環境下でどのように行動則を獲得していくかは，ロボティクスにおいて重要な課題である．この問題に対する主要なアプローチの一つに，Q 学習 [1] などの強化学習アルゴリズムを構成する value-iteration がある．Value-iteration では各状態において各行動を取る価値を表す行動価値関数を推定し，それを利用して行動を決定する．行動の選択は行動価値関数を行動について最大化することを必要とするので，連続値を持つ行動は計算量の面で実装が困難であることが多い．

別のアプローチである policy gradient（例えば REINFORCE algorithm[2]）は，value-iteration と異なり方策を行動の確率分布として直接表現し，最適な方策を探索する．このアプローチでは，獲得された分布のサンプルとして行動を生成するため，連続値を持つ行動も比較的容易に生成することができる．しかし，容易にサンプルを生成できる分布は限られており，方策の空間として正規分布などのパラメトリックな分布族が用いられることが多い．この制約を解消するために，ノンパラメトリックな方策を学習する様々な試み（[3, 4] など）がなされてきたが，一般に行動生成のための計算量はパラメトリックな分布と比較して増大する．このため，これらのアプローチでは行動生成時に扱いやすい分布（例えば [4] では正規分布）で近似したり，結局離散的な分布（例えばボルツマン分布 [3]）を用いる必要がある．

本稿では，行動生成を容易にしつつ，ノンパラメトリックな確率的方策を学習するための手法を提案する．本提案では，図 1 に示すように，直接最適な行動の分布を獲得する代わりに，既知分布から所望の分布への確率変数変換を探索する問題として学習を定式化する．

2. 確率変数変換の学習

Actor-critic などで用いられる確率的方策の表現は，$a_t \sim N(\mu(s_t), \sigma^2(s_t))$，つまり平均と分散が状態 s_t の関数となっている正規分布であることが多い．言い換えれば，行動 a_t の値は，$\mu(s_t)$ と $\sigma(s_t)$ を何らかの関数近似器で計算した後に，$n_t \sim N(0,1)$ を加えることによって

$$a_t = \mu(s_t) + \sigma(s_t) n_t. \tag{1}$$

のように決定される．したがって，当然のことではあるが，$\mu(s_t)$ と $\sigma(s_t)$ の計算過程によらず $p(a_t|s_t)$ は常に正規分布となる．

連絡先: 横山裕樹，hiroki.yokoyama@okadanet.org

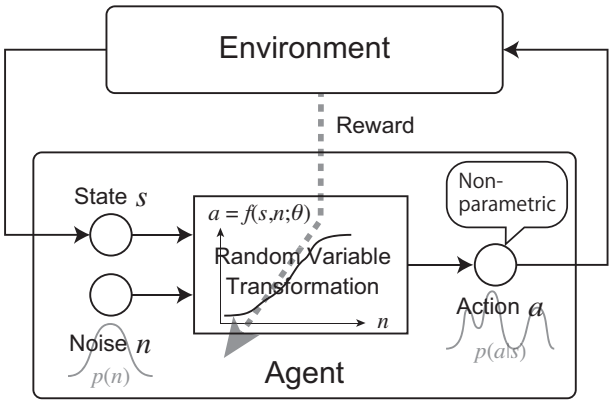

図 1: 提案手法の概念図.

本研究は，既知の分布を持つノイズ n_t を関数近似器の入力の一部として扱うことを提案する．すなわち，

$$a_t = f(s_t, n_t; \theta) \tag{2}$$

によって a_t を生成する．ここで θ は関数近似器のパラメータを表す．簡単のため，以後 $n_t \sim N(0,1)$ とする．s と θ を固定すれば，式 (2) は n から a への変数変換とみることができ，n の分布に対して a の分布が一つ定まる．そして s と θ が変化すれば，両者の分布の対応関係も変化する．例えば，f が n について一次式であれば，a は n と同様に正規分布に従うが，関数近似器に十分な表現力があれば，a の分布は f の形状に応じて様々な形状を取り得る．学習の目的は a の分布が望ましい形状を持つような f を獲得することである．もし最適な f を獲得できれば，後は観測された s_t と生成された n_t から $f(s_t, n_t)$ を計算するだけで所望の分布に従う a_t を生成することができる．

上記の方法で実現される確率的方策は

$$\begin{aligned} p(a_t|s_t;\theta) &= \int p(a_t|n_t, s_t) p(n_t) \mathrm{d}n_t \\ &= \int \delta(a_t - f(s_t, n_t; \theta)) p(n_t) \mathrm{d}n_t. \end{aligned} \tag{3}$$

となる．ここで $f(s_t, n_t; \theta)$ の計算は決定論的であるので，これをディラックのデルタ関数 $\delta(\cdot)$ を用いて $p(a_t|n_t, s_t) = \delta(a_t - f(s_t, n_t; \theta))$ と表現している．

多くの強化学習アルゴリズムは，期待収益

$$R(s,a) = E\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k} \;\middle|\; s_t = s, a_t = a\right] \tag{4}$$

（$0 < \gamma < 1$ は未来の報酬にどれだけ重みを置くかを定める割引率）を最大化するような確率的方策を求める問題として定式化される．この場合，目的関数は

$$E = \int R(s_t,a_t)\int \delta(a_t - f(s_t,n_t;\theta))p(n_t)\mathrm{d}n_t\mathrm{d}a_t. \tag{5}$$

と書ける．この目的関数には，内部に多変量関数を持つデルタ関数が含まれている．次節では，このような関数の微分を勾配法を構成するために，より扱いやすい形に変形することを考える．

2.1 デルタ関数の方向微分

ここでは，上記の目的関数を一般的な形式で表し，その方向微分について考察する．式 (5) は

$$\int_{\mathbb{R}^2}\delta'(\psi(x,y))\phi(x,y)\mathrm{d}x\mathrm{d}y \tag{6}$$

のような形式をとっていることがわかる (ここでは重積分を $\int$ で表す)．このような式を扱うために，一般化ストークスの定理

$$\int_D \mathrm{d}w = \oint_{\partial D} w \tag{7}$$

を微分形式 $w = -v_y f g\mathrm{d}x + v_x f g\mathrm{d}y$ について適用する．ここで f, g はスカラー場，$\boldsymbol{v} = (v_x, v_y)^\top$ はベクトル場である．したがって，

$$\int_D (\nabla_{\boldsymbol{v}} f)g\mathrm{d}x\wedge\mathrm{d}y = \oint_{\partial D} w - \int_D (f\cdot(\nabla_{\boldsymbol{v}} g) + (\mathrm{div}\,\boldsymbol{v})fg)\mathrm{d}x\wedge\mathrm{d}y, \tag{8}$$

のように部分積分の一般化が得られ，これによって f の方向微分を消去することができる．式 (8) の f に $\delta'(\psi)$ を，g に ϕ を代入することで，

$$\begin{aligned}\int_D \delta'(\psi)\phi\,\mathrm{d}x\wedge\mathrm{d}y &= \oint_{\partial D}\delta(\psi)\frac{\phi}{\nabla_{\boldsymbol{v}}\psi}(-v_y\mathrm{d}x + v_x\mathrm{d}y)\\ &- \int_D \delta(\psi)\left(\frac{\nabla_{\boldsymbol{v}}\psi\nabla_{\boldsymbol{v}} - \boldsymbol{v}^\top H_\psi \boldsymbol{v} - \nabla_{\nabla_{\boldsymbol{v}}\boldsymbol{v}}\psi}{(\nabla_{\boldsymbol{v}}\psi)^2} + \frac{\mathrm{div}\,\boldsymbol{v}}{\nabla_{\boldsymbol{v}}\psi}\right)\phi\,\mathrm{d}x\wedge\mathrm{d}y,\end{aligned} \tag{9}$$

が得られる．ここで最後の積分は，式 (6) の δ を δ' に替えたものとみることができる．経路積分は残差であり，次節で述べるように，一定の条件のもとで無視することができる．

2.2 更新則

(x,y) に (a,n)，ψ に $a - f(s,n;\theta)$，ϕ に $R(s,a)f_\theta(s,n;\theta)p(n)$ を代入し，ベクトル場を定数 $(v_a, v_n) = (0,1)$ とすると，以下の式が得られる．

$$\begin{aligned}\frac{\partial}{\partial\theta}E[r] &\approx \lim_{D\to\mathbb{R}^2}\int_D \delta(a-f)\left(\frac{f_{nn} - f_n\frac{\partial}{\partial n}}{f_n^2}\right)(Rf_\theta p(n))\mathrm{d}a\mathrm{d}n\\ &= E\left[R\frac{(f_{nn} - (\log p(n))' f_n)f_\theta - f_n f_{n\theta}}{f_n^2}\right]\end{aligned} \tag{10}$$

ここで $f_n = \frac{\partial f}{\partial n}, f_{nn} = \frac{\partial^2 f}{\partial n^2}, f_{n\theta} = \frac{\partial^2 f}{\partial n\partial\theta}$ である．If converges sufficiently fast as , the residual $p(n)$ が $n\to\pm\infty$ に対して十分な速度で収束するならば，領域 D が大きくなる（$D\to\mathbb{R}^2$）とき

$$\oint_{\partial D}\delta(a-f)\frac{Rf_\theta p(n)}{f_n}\mathrm{d}a \tag{11}$$

も同様に収束する．このことから以下の更新式が考えられる．

$$\theta \leftarrow \theta + \alpha_t r_t\frac{(f_{nn} - (\log p(n))' f_n)f_\theta - f_n f_{n\theta}}{f_n^2} \tag{12}$$

ここで α_t は学習率である．いま，$n\sim N(0,1)$ としているので，その確率密度の対数微分 $(\log p(n))'$ は単に $-n$ と表せる．更新則には確率変数が含まれているため，θ の更新は各ステップ毎にランダムとなるが，その繰り返しによって，平均的には目的関数である期待収益が増加することになる．

2.3 TensorFlow による実装

関数近似器として深層学習器を用いる場合，Caffe や TensorFlow などの計算ライブラリの使用を考慮することが多いと思われる．本稿で提案した学習則は，一般にこのようなライブラリで用意されている，パラメータ（式 (12) の θ）に関する一階微分だけでなく，入力（n）に関する二階までの微分をも含んでいるため，これらの使用に際してモデルの定義等に工夫が必要となる．TensorFlow は，計算過程をグラフで表現することにより自動微分を実現しているため，本稿の学習則を比較的容易に実装できる．具体的には，`gradients` という関数によって f の微分 $f_n, f_n n$ を求め，後で自動的に行われる θ に関する微分に際してこれ以上微分されないように `stop_gradient` という関数を用いる．TensorFlow（ここでは `tf` という名前でインポートされている）で実装される最も単純なコードは以下の通りである．

Listing 1: An example of implementation in python with TensorFlow.

```
a = f(s,n)
dfdn = tf.gradients(a, n)[0]
d2fdn2 = tf.gradients(dfdn, n)[0]
E = r*(a*tf.stop_gradient((d2fdn2+n*dfdn)/(dfdn**2))
    - dfdn/tf.stop_gradient(dfdn))
opt = tf.train.GradientDescentOptimizer(learning_rate)
train_step = opt.minimize(-E)
```

変数 `s`，`n`，`r` はそれぞれ状態，ガウスノイズ，報酬に相当する．関数 `f` はニューラルネットワークのような関数近似器を表す．学習は上記コードで用意された `train_step` を繰り返し実行することで実現できる．`f` を定義するパラメータ（結合重みなど）に関する微分は TensorFlow により自動的に求められる．一般の学習アルゴリズムと同様に，実際には学習の安定のため，学習率の減衰などを考慮する必要がある．

3. 数値実験

本研究の提案で最も重要なものは，ガウス分布に限らない，多様な分布を獲得できるということである．これを確認するため，ガウスノイズ n を入力とし，a を出力するニューラルネットワークを学習させ，a が多峰性の分布となるかどうかを検証した．

ニューラルネットワークは 4 階層の全結合型で，第 2 層と第 3 層はそれぞれ 16 個と 8 個の素子を持つ．一変数の確率変数変換は単調増加の関数で表せるので，非単調な関数を獲得することを避けるために，全ての結合重みを非負に制約した．なお，バイアスは負の値を取り得る．また，活性化関数には $\tanh(\cdot)$ を用いた．入力となるノイズ n の分布は標準正規分布（図 2 の点線）とした．出力 a の評価を $r = 0.7e^{-(a+1)^2/4} + 0.7e^{-(a-1)^2/4} - 1.5e^{-a^2/2}$

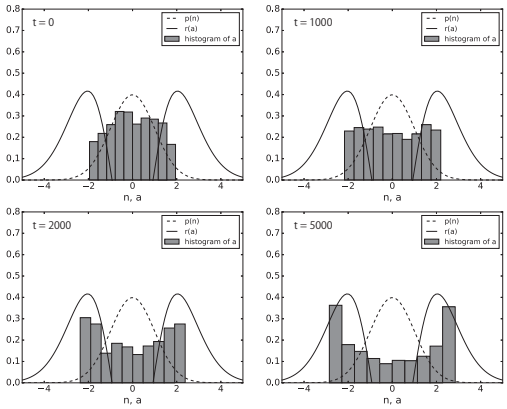

図 2: 数値実験．ノイズの分布 n $N(0,1)$ を点線で，$a = f(n)$ の分布をヒストグラムで示す．実線で示す報酬関数 $r(a)$ に基づき報酬を最大化するように a の分布が変化している．

とした．図 2 の実線のように，この関数には高さの等しいピークが二つ存在するため，常にそれらのどちらかを選ぶという方策が最適となる．

図 2 のヒストグラムは $a = f(n)$ のサンプルの分布である．横軸は n と a で共通となっているため，点線が示す n の分布からヒストグラムが示す a の分布への f による変化を見ることができる．f のパラメータの初期値によるが，この場合では学習初期において a は単峰性分布となっている．提案手法により f のパラメータを学習することで，報酬が負となる 0 付近の頻度が減少し，報酬関数のピーク付近の頻度が増加している．もし方策が正規分布に制限されていれば，学習初期にそれまでの経験に基づいてどちらかのピークが選択されていたと考えられる．これは探索の意味で好ましくない．提案手法では，両者の可能性を考慮しつつ，明らかに報酬の小さい行動は取らないように学習することができている．

4. おわりに

今回は単純な問題を用いて，提案手法によって，明示的に分布の形状を想定しなくても多峰分布が獲得できるかどうかを検証した．今後さらに，状態によって異なる分布が獲得できるかどうかも検証して行く．

強化学習において確率的方策を扱うことは，未経験の行動を実行して環境からの評価を得る，つまり探索を行うために重要である．本稿の手法により多様な探索を可能とすることで，効率的な学習が期待できる．またゲームなど，他のエージェントの存在を考慮する場合は，そもそも最適方策が一般に確率的である．このような場合に多様な確率的方策を獲得できることはさらに重要である．

本稿では，確率的方策を確率密度関数のパラメトリックな表現ではなく，既知の分布からの変数変換と捉えることで，強化学習の枠組みを提案したが，実際にどのような関数近似器を用いるかは重要な問題として残されている．今後，確率変数の変換の学習に適した関数近似器の構造について考える必要がある．

謝辞

本研究は，JST，CREST の支援を受けたものである．

参考文献

[1] C. J. C. H. Watkins, “Learning from Delayed Rewards,” Ph.D. dissertation, Cambridge Univ., Cambridge, England, 1989.

[2] R. J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. Machine Learning, vol. 8, no. 3, pp. 229–256, 1992.

[3] K. Kersting and K. Driessens, “ Non-Parametric Policy Gradients: A Unified Treatment of Propositional and Relational Domains,” in Pro. 25th Int. Conf. Machine Learning, Helsinki, Finland, 2008.

[4] H. van Hoof and J. Peters, “Learning of Non-Parametric Control Policies with High-Dimensional State Features,” in Proc. 18th Int. Conf. Aritificial Intelligence and Statistics, San Diego, CA, USA, 2015.