The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

決定木とCross-Entropy法を用いた解釈可能な制御方策の学習

Learning Interpretable Control Policies with Decision Trees via the Cross-Entropy Method

田中　友紀子 *1*2　　平岡　拓也 *1*2　　鶴岡　慶雅 *2*3
Yukiko TANAKA　　Takuya HIRAOKA　　Yoshimasa TSURUOKA

*1NEC セキュリティ研究所
NEC Security Research Laboratories

*2産業技術総合研究所 人工知能研究センター
National Institute of Advanced Industrial Science and Technology, Artificial Intelligence Research Center

*3東京大学
The University of Tokyo

Learning interpretable policies for control problems is important for industrial requirements for safety and maintenance. A common approach to acquiring interpretable policies is to learn a decision tree that imitates a black-box (e.g., neural network-based) policy trained to maximize the expected reward in a given environment. However, such approximated decision tree policies are suboptimal in the sense that they do not necessarily maximize the expected reward. In this paper, we propose a method for learning a decision tree policy that directly maximizes the reward using the cross-entropy method. Our experimental results show that our method can acquire interpretable decision tree policies that perform better than baseline policies learned by the imitation approach.

1. はじめに

解釈性のある制御方策 (以降、方策と呼ぶ) を学習することは、産業応用における安全性やメンテンナンス容易性の観点で重要である [Lipton 18]。人間が方策を解釈することができれば、実際に方策に従って操作を実行する前に、操作の安全性を確かめることができる。また、操作の過程で問題が起きた場合でも、解釈性のある方策は人の手によって修正することができる。

深層強化学習が多くの制御タスクで人間を介さずに高い性能の方策を獲得することに成功した [Silver 17, Andrychowicz 18] 一方で、これらの方策はニューラルネットワークに基づいているため、一般に解釈性が低い。そのような問題に対して、方策の学習に決定木のような解釈性の高いモデルを使う方法が存在する。Sammut らは、熟練操作員の操作ログから、教師あり学習を用いて決定木に基づく方策を構築している [Sammut 92]。Verma らは、ニューラルネットワークでモデル化した方策を模倣するように解釈性のある方策を学習している [Verma 18]。Liu らは、Q-Learning で得られたブラックボックスの Q 関数を近似することで、決定木でモデル化された Q 関数を構築している [Liu 18]。これらの模倣に基づく手法における主要な問題点のひとつに、学習済みのブラックボックス方策や Q 関数を近似するように学習した決定木方策は、モデルの複雑度の差によって、元のブラックボックス方策や Q 関数よりも性能が低下することが挙げられる。他方、Hein らは解釈性のある木構造の方策を、ブラックボックス方策の模倣を介さずに、遺伝プログラミングを用い期待報酬を直接最大化して学習している [Hein 18] が、このアプローチでは方策パラメータを離散化して扱う必要があり、そのことが学習で得た方策の性能低下を引き起こす可能性がある。

本研究では、Cross-Entropy 法 (CEM) [Szita 06] を用い、環境から与えられる期待報酬を直接最大化することで、解釈性のある決定木方策を獲得する手法を提案する (図 1) 。本論文では、まず CEM により決定木方策が直接最適化できるよう、決定木方策のパラメータ表現を定義する。そして、我々の手法を評価するため、複数の制御タスクで実験をする。実験の結果より、ニューラルネットワークに基づく方策のふるまいを模倣することで決定木方策を構築するベースライン手法よりも我々の手法がよい性能となることを示す。さらに、我々の手法で獲得した方策について考察し、それらがどのような解釈性をもつかを示す。

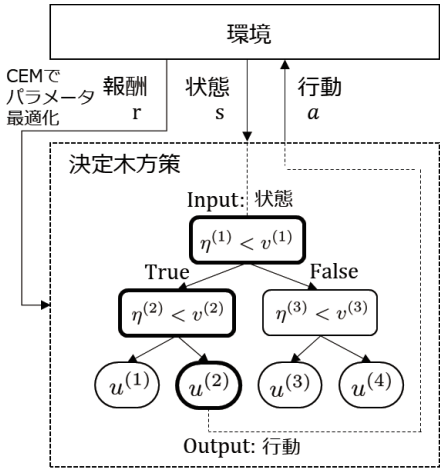

図 1: 決定木方策の学習

2. 検証準備

2.1 強化学習

強化学習の目標は、期待累積報酬を最大化するエージェントの方策を学習することである [Sutton 98] 。一般に、エー

連絡先: 田中　友紀子，NEC　セキュリティ研究所，y-tanaka@jz.jp.nec.com

ジェントが臨む問題は、マルコフ決定過程 (Markov Decision Processes; MDPs) で表される。MDPs では、各時間ステップにおいて、エージェントが環境から状態 $s_t \in S$ を観測し、環境に行動 $a_t \in A$ を返す。ここで、S は状態空間で、A は行動空間である。次に環境が次の s_{t+1} に遷移し、エージェントが報酬 $r_{t+1} \in \mathbb{R}$ を受け取る。エージェントは方策 π に従い、s_{t+1} における a_{t+1} を決定する。本問題の目的は、エピソード周期 T における期待累積報酬 $R(\pi^*) = E_{\pi^*}\left[\sum_{t=1}^{T} r_t\right]$ を最大化する最適な方策 π^* を獲得することである。なお、本論文では、feature 関数 $\phi(s)$ を用い、状態 s は F 次元ベクトル表現 $\left[\phi^{(1)}(s), ..., \phi^{(F)}(s)\right]$ に変換されると仮定する。

2.2 Cross-Entropy 法

本研究では、最適方策の獲得に CEM を用いる。具体的には、パラメータ $\boldsymbol{\theta} \in \mathbb{R}^d$ で方策 π をパラメータ化し（パラメタライズされた方策は π_θ と表記）、$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} R(\pi_{\boldsymbol{\theta}})$ となる最適パラメータ値 $\boldsymbol{\theta}^*$ をみつけるために CEM を使う。

CEM では、1) パラメータサンプル生成、2) パラメータサンプル評価、3) パラメータ生成確率分布更新、を繰り返すことで $\boldsymbol{\theta}^*$ を探索する。ステップ 1 では、b 個のパラメータサンプル $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_b$ が確率分布から生成される。本研究では、確率分布として、平均ベクトル $\boldsymbol{\mu}$ と対角共分散行列 $\boldsymbol{\Sigma}$ をもつ多変量ガウス分布 $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ を用いる。ここで、$\sigma^{(i,i)}$ を、$\boldsymbol{\Sigma}$ の i 番目の対角成分と定義する。ステップ 2 では、$\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_b$ がそれぞれに従う方策を実行することで得られる累積報酬 $R(\pi_{\boldsymbol{\theta}_1}), ..., R(\pi_{\boldsymbol{\theta}_b})$ に基づいて評価される。そして、高い累積報酬を得た上位 m 個のサンプルが $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_b$ から選択される。これらの選ばれたサンプル集合を M と定義する。ステップ 3 では、サンプル生成用の確率分布が M によって更新される。本研究では、前記多変量ガウス分布の平均ベクトルは次のように更新される。

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{\boldsymbol{\theta}' \in M} \boldsymbol{\theta}'. \tag{1}$$

また、$\boldsymbol{\Sigma}$ の i 番目の対角成分は次のように更新される [*1]。

$$\sigma^{(i,i)} = \frac{1}{m} \sum_{\boldsymbol{\theta}' \in M} (\theta^{(i)'} - \mu^{(i)})^2 + \epsilon, \tag{2}$$

ここで、$\theta^{(i)'}$ は $\boldsymbol{\theta}'$ の i 番目の要素であり、$\mu^{(i)}$ は $\boldsymbol{\mu}$ の i 番目の要素である。また、ϵ は学習早期における収束を防ぐための探索ノイズの定数である。このような手順を通し、CEM は $\boldsymbol{\theta}^*$ を探索する。

3. 決定木方策の学習

この章では、決定木方策の学習手法について述べる。始めに決定木方策のパラメータ表現について説明し、次に CEM を用いたパラメータ更新手順について示す。

3.1 パラメータ表現

この節では、決定木方策のパラメータ表現について定義する。本研究では、図 2 の右側に示すような二分木構造をもつ決定木を方策の表現として用いることとする。この方策において、非終端ノードは条件を表し、葉ノードは選択される行動を表す。また、ノード間のリンクは条件の評価結果を表す。この

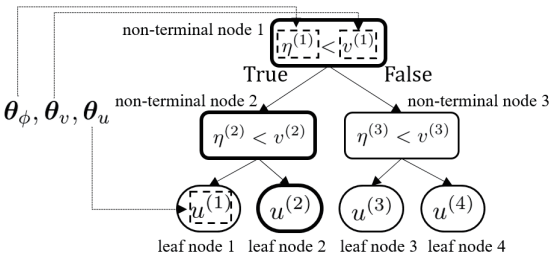

図 2: 決定木方策のパラメータ化

方策は、ある状態が与えられたとき、木の根ノードから葉ノードまでを辿ってひとつの行動を選択する。ここで、n を非終端ノードのインデックス番号として表記し、非終端ノードの数を l 、決定木の最大の深さを D と表記する [*2]。

我々は、図 2 の左側に示すように、$\boldsymbol{\theta}_\phi$、$\boldsymbol{\theta}_v$、$\boldsymbol{\theta}_u$ で決定木方策をパラメータ化する。$\boldsymbol{\theta}_\phi$ は各非終端ノードで評価される $\phi(s)$ の要素を決定するために用いられる。$\boldsymbol{\theta}_\phi$ は $(2^D - 1) \times F$ 行列として表現される:

$$\boldsymbol{\theta}_\phi = \begin{bmatrix} \theta_\phi^{(1,1)} & \cdots & \theta_\phi^{(1,f)} & \cdots & \theta_\phi^{(1,F)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_\phi^{(n,1)} & \cdots & \theta_\phi^{(n,f)} & \cdots & \theta_\phi^{(n,F)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_\phi^{(2^D-1,1)} & \cdots & \theta_\phi^{(2^D-1,f)} & \cdots & \theta_\phi^{(2^D-1,F)} \end{bmatrix}, \tag{3}$$

ここで、$\theta_\phi^{(n,f)}$ は n 番目の非終端ノードでの評価における状態ベクトル $\phi(s)$ の f 番目の要素に対応する値である。n 番目の非終端ノードにおいて、評価に用いられる状態ベクトルの要素 $\eta^{(n)}$ は、$\boldsymbol{\theta}_\phi$ を用いて次のように決定される: 始めに、$\boldsymbol{\theta}_\phi$ の n 番目の行にある全ての要素を取り出す (これらの要素を $\{\theta_\phi^{(n,1)}, \cdots \theta_\phi^{(n,F)}\}$ とする) 。次に、$\{\theta_\phi^{(n,1)}, \cdots \theta_\phi^{(n,F)}\}$ の中で最も値が高い要素を選択し、対応する feature の要素を $\eta^{(n)}$ として用いる。例えば、$\{\theta_\phi^{(n,1)}, \cdots \theta_\phi^{(n,F)}\}$ から $\theta_\phi^{(n,f')}$ が選択された場合、$\eta^{(n)}$ は $\eta^{(n)} := \phi^{(f')}$ と決定される。

$\boldsymbol{\theta}_v$ は各非終端ノードにおける閾値を決定するために用いられる。$\boldsymbol{\theta}_v$ は $(2^D - 1)$ 次元ベクトルで表現される:

$$\boldsymbol{\theta}_v = \left[\theta_v^{(1)}, \cdots, \theta_v^{(n)}, \cdots, \theta_v^{(2^D-1)}\right], \tag{4}$$

ここで $\theta_v^{(n)}$ は n 番目の非終端ノードにおける閾値 $v^{(n)}$ を表す。$v^{(n)}$ は $v^{(n)} = (v_{max}^{(n)} - v_{min}^{(n)}) g(\theta_v^{(n)}) + v_{min}^{(n)}$ で算出される。ここで、$v_{max}^{(n)}$ と $v_{min}^{(n)}$ は、$v^{(n)}$ の上限値と下限値である。また、g はクリッピングを行うための関数であり、本研究では sigmoid 関数を用いる。

$\boldsymbol{\theta}_u$ は葉ノードにおける行動を決定するために用いられる。我々は、離散行動問題と連続行動問題それぞれのための $\boldsymbol{\theta}_u$ の表現を提案する。

離散行動問題を想定した場合、$\boldsymbol{\theta}_u$ は $2^D \times |A|$ 行列として

*1 ただし、ステップ 3 において、Σ のすべての対角成分は式 (2) で更新される

*2 つまり、非終端ノードの総数は $(2^D - 1)$ 、葉ノードの総数は 2^D である

表される:

$$\boldsymbol{\theta}_u = \begin{bmatrix} \theta_u^{(1,1)} & \cdots & \theta_u^{(1,k)} & \cdots & \theta_u^{(1,|A|)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_u^{(l,1)} & \cdots & \theta_u^{(l,k)} & \cdots & \theta_u^{(l,|A|)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_u^{(2^D,1)} & \cdots & \theta_u^{(2^D,k)} & \cdots & \theta_u^{(2^D,|A|)} \end{bmatrix}, \quad (5)$$

ここで $\theta_u^{(l,k)}$ は l 番目の葉ノードにおける k 番目の行動候補 $a^{(k)} \in A$ に対応する値を表す。方策実行時において、$\{\theta_u^{(l,1)}, \cdots \theta_u^{(l,|A|)}\}$ のうち最も値の大きい要素に対応する行動候補が l 番目の葉ノードにおける行動 $u^{(l)}$ として選ばれる。

連続行動問題を想定した場合、$\boldsymbol{\theta}_u$ は 2^D 次元ベクトルで表される:

$$\boldsymbol{\theta}_u = \left[\theta_u^{(1)}, \cdots, \theta_u^{(l)}, \cdots, \theta_u^{(2^D)}\right], \quad (6)$$

ここで $\theta_u^{(l)}$ は l 番目の葉ノードにおける行動 $u^{(l)} \in \mathbb{R}$ に対応する値を表す。$u^{(l)}$ は $u^{(l)} = \alpha h(\theta_u^{(l)})$ で算出される。ここで関数 h はクリッピングを行うための関数であり、本研究では tanh 関数を用いる。また、 α は $u^{(l)}$ を適切な範囲に収めるための係数である。

3.2 パラメータ更新

前節で説明した決定木方策のパラメータ (すなわち、$\boldsymbol{\theta}_\phi$、$\boldsymbol{\theta}_v$、$\boldsymbol{\theta}_u$) を報酬を最大化するように更新する。パラメータの更新手順を、Algorithm 1 に示す。ここでは、 $\boldsymbol{\theta}_\phi$、$\boldsymbol{\theta}_v$、$\boldsymbol{\theta}_u$ のすべての要素を結合して $\boldsymbol{\theta}$ として扱い、CEM を適用する。

Algorithm 1 Cross-Entropy 法による決定木方策の学習

Require: initial mean vector $\boldsymbol{\mu}_0$, initial diagonal covariance matrix $\boldsymbol{\Sigma}_0$, number of elite m, constant exploration noise ϵ

1: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_0$
2: $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma}_0$
3: **for** each update **do**
4: Generate b parameter samples $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_b$ from $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
5: Evaluate each parameter samples using $R(\pi_{\boldsymbol{\theta}_1}), ..., R(\pi_{\boldsymbol{\theta}_b})$
6: Select m parameter samples with the highest evaluations.
7: Update $\boldsymbol{\mu}$ by Equation (1)
8: Update all diagonal elements in $\boldsymbol{\Sigma}$ by Equation (2)
9: **end for**

4. 実験

4.1 タスク

今回、2 つの制御タスクにて手法を評価する。タスク設定のサマリを表 1 に示す。

CartPole: ポールが倒れないよう、台車をコントロールするタスクである。今回、OpenAI Gym [Brockman 16] の「CartPole-v0」環境を用いる。状態空間は連続で 4 次元、行動空間は 2 つの離散行動 (台車に加える力の方向) である。

Pendulum: トルクをコントロールし、振り子を倒立状態に維持するタスクである。今回、OpenAI Gym の「Pendulum-v0」環境を用いる。状態空間は連続で 3 次元、行動空間は 1 次元の連続値 (トルク) である。

表 1: タスク設定

タスク	F	行動
CartPole	4	離散、$\|A\| = 2$
Pendulum	3	連続、$a \in \mathbb{R}$

表 2: ベースライン手法と提案手法の平均累積報酬

タスク	D	ベースライン	提案
CartPole	2	13.2	**200.0**
(NN: 200.0)	3	24.4	**200.0**
	4	10.55	**200.0**
Pendulum	2	−1091.4	**−848.96**
(NN: −342.5)	3	−1331.1	**−290.98**
	4	−1219.94	**−542.57**

CartPole、Pendulum の両タスクにおいて、各エピソードの初期状態はランダムに初期化される。

4.2 実験設定

ベースライン手法としてブラックボックス方策のふるまいを模倣する決定木を用いる [Sammut 92, Verma 18, Liu 18] 。まず始めに、ニューラルネットワークに基づく方策 (ブラックボックス方策) を PPO [Schulman 17] を用いて学習する。次に、ニューラルネットワークに基づく方策を用いて 20 エピソード分の試行をし、訓練用のサンプル (状態と行動のペア) を集める。最後に、訓練用のサンプルを用いて、教師あり学習で決定木方策を構築する。なお、ニューラルネットワークは 64 ノードの隠れ層 2 層を用いる。また、PPO は 環境との相互作用を 10^6 回通して学習される。

また、提案手法として Algorithm 1 を用いて決定木を構築する。ハイパーパラメータには、$b = 20$、$m = 4$ 、$\epsilon = 0.0001$ を用いる。

5. 評価

表 2 に定量的な評価結果を示す。今回、20 エピソードの平均累積報酬に基づいて前記手法を評価した。ベースライン手法は、5 つの乱数シードを用いた試行のうち、学習終了時の方策の成績 (平均累積報酬) がもっともよい方策を用い、決定木を構築した。提案手法は、5 つの乱数シードを用いた試行のうち、学習時における方策の成績 (平均累積報酬) がもっともよい更新回の方策を評価に用いた。なお、表 2 に示す評価値は、学習時とは異なる新たな 20 エピソード分の試行を用いて算出したものである。表 2 内の D は決定木の最大の深さである。また、表 2 の各タスク名の下に「NN」として記載している手法は、PPO で学習したニューラルネットワークに基づく方策を用いたものである。表 2 の結果より、我々の手法はベースライン手法の成績を上回っていることがわかる。

決定木方策の解釈性についての定性的な評価をする。図 3 に、Pendulum タスクで学習した決定木方策の可視化例を示す。方策がある状態を環境から受け取った場合、図 3 に示されるような決定木方策の各非終端ノードでの条件分岐結果に従い、次の時間ステップで取るべき行動を選択する。非終端ノードでの条件文は、振り子の位置や速度のような物理量を用いた表現で構成されており、タスクを達成する方法 (どのように振り子を倒立した状態に維持するか) が人間からみて理解しやすい形式となっている。また、図 4 に CartPole タスクで学習した決定木方策の例を可視化した図を示す。図 4 では、2 番目の

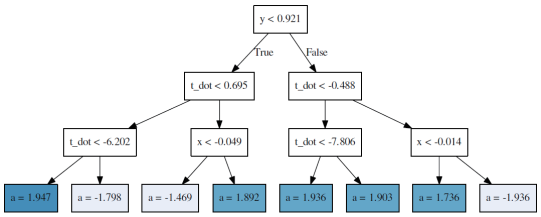

図 3: Pendulum タスクにおける Cross-Entropy 法を用いた決定木方策 $D = 3$.

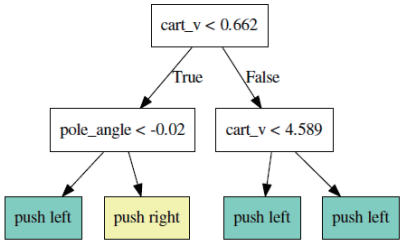

図 4: CartPole タスクにおける Cross-Entropy 法を用いた決定木方策 $D = 2$.

深さの右に位置する非終端ノードの条件「cart_v < 4.589」の分岐結果はいずれも「push left」となっている。このような結果は、図 5 のようなより簡潔な表現にすることができる。図 5 の例も、図 3 と同様に車の速度やポールの傾き角のような物理量を用いて方策が表現されており、人間からみて理解しやすい形で可視化されている。本研究では、Pendulum、CartPole の両タスクで図 3、図 5 のような比較的浅い木構造で学習結果を確認することができた。このような決定木方策は、数百のノードを含むニューラルネットワークに基づく方策と比較し、学習で得られた方策の確認に要する時間や労力が軽減される。

6. まとめ

今回、CEM を用いて報酬を直接最大化する、解釈性のある決定木方策の獲得手法を提案した。2 つの単純な制御タスクにおける実験結果から、ニューラルネットワークに基づく方策を模倣するベースライン手法より我々の手法がよい性能となることが分かった。将来的には、よい性能を出すためにより深い木構造が必要となるような、より複雑なタスクでの検証をすることが望ましい。深い決定木において、最適な方策パラメータを求めることは難しいことが予期される。今後は、解釈性を保ちつつ複雑なタスクに適用できる手法に拡張する必要がある。

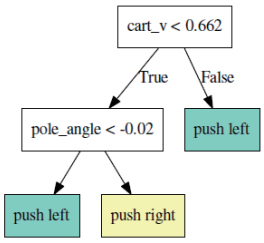

図 5: 簡潔な表現にした決定木方策

参考文献

[Andrychowicz 18] Andrychowicz, O. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W.: Learning Dexterous In-Hand Manipulation, Vol. arXiv:1808.00177, (2018)

[Brockman 16] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W.: OpenAI Gym, Vol. arXiv:1606.01540, (2016)

[Hein 18] Hein, D., Udluft, S., and Runkler, T. A.: Interpretable Policies for Reinforcement Learning by Genetic Programming, *Engineering Applications of Artificial Intelligence*, Vol. 76, pp. 158–169 (2018)

[Lipton 18] Lipton, Z. C.: The Mythos of Model Interpretability, *Queue*, Vol. 16, No. 3, pp. 30:31–30:57 (2018)

[Liu 18] Liu, G., Schulte, O., Zhu, W., and Li, Q.: Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 414–429, Springer (2018)

[Sammut 92] Sammut, C., Hurst, S., Kedzier, D., and Michie, D.: Learning to fly, in *Machine Learning Proceedings 1992*, pp. 385–393, Elsevier (1992)

[Schulman 17] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.: Proximal Policy Optimization Algorithms, Vol. arXiv:1707.06347, (2017)

[Silver 17] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of Go without human knowledge, *Nature*, Vol. 550, No. 7676, p. 354 (2017)

[Sutton 98] Sutton, R. S. and Barto, A. G.: *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA, USA, 1st edition (1998)

[Szita 06] Szita, I. and Lörincz, A.: Learning Tetris using the noisy cross-entropy method, *Neural computation*, Vol. 18, No. 12, pp. 2936–2941 (2006)

[Verma 18] Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S.: Programmatically Interpretable Reinforcement Learning, in *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *PMLR*, pp. 5045–5054, Stockholmsmssan, Stockholm Sweden (2018)