人工知能学会研究会資料
SIG-AGI-014-09

推論規則の価値を階層型強化学習 RGoal を用いて学習する手法の提案

Preliminary Study of a Method for Learning Inference Rules Using Hierarchical Reinforcement Learning RGoal

一杉裕志 [1*]　中田秀基 [1]　高橋直人 [1]　佐野崇 [2]
Yuuji Ichisugi[1]　Hidemoto Nakada[1]　Naoto Takahashi [1]　Takashi Sano[2]

[1] 産業技術総合研究所 人工知能研究センター
[1] National Institute of Advanced Industrial Science and Technology (AIST), AIRC
[2] 成蹊大学 理工学部 情報科学科
[2] Department of Computer and Information Science, Faculty of Science and Technology, Seikei University

Abstract: We propose a method to obtain correct inference rules from experience for an intelligent agent running in an environment. For each inference rule, the proposed method substitutes "efficiency to reach the correct answer" for "correctness." This proposed method learns the inference rules using a hierarchical reinforcement learning method called RGoal. The whole architecture is biologically plausible. We believe the proposed method will be a basic principle of autonomous knowledge acquisition for artificial general intelligence.

概要

環境の中で動く知的エージェントが正しい推論規則を経験から獲得するための手法を提案する。提案手法は、推論規則の「正しさ」を「正解に到達するまでのコストの低さ」で代用し、それを階層型強化学習 RGoal によって学習する。全体のアーキテクチャは神経科学的にも妥当である。提案手法は汎用人工知能の自律的な知識獲得の基本原理の 1 つになると考えている。

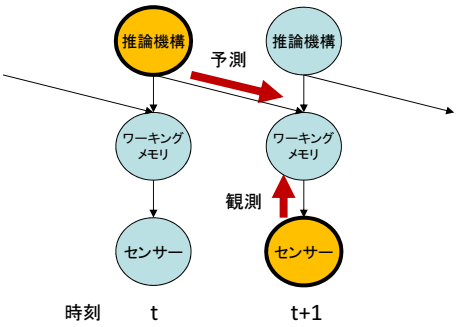

図 1: アーキテクチャを確率的生成モデルの形で表現したもの。ワーキングメモリは環境の状態を表現している。推論機構はワーキングメモリの現在の状態から次の状態を予測・推論する。

1　はじめに

ヒトはおそらく記号的な推論を行う機構を持っている。推論に用いる知識は決して完全なものではないが、経験に基づいてできるだけ正しいものになるように無意識のうちに修正を行っているように思われる。例えば幼児が次のような推論を行ったとしよう。
「きのう戸棚にチョコレートがあった。ということは、きょうもあるはず。」
そして、もし戸棚の中にチョコレートがなければ次のように考えるかもしれない。
「あれ？チョコレートがない。そういえば、おにいちゃんが今朝食べたと言っていた。食べたからなくなったのか … 。」
この経験により「きのうチョコレートがあったらきょうもチョコレートがある」という推論は常に正しいとは限らない、と学習することができるだろう。このように経験から推論規則の正しさを学習するヒトの振る舞いを、機械で再現することは可能だろうか。

記号ＡＩと統計的機械学習の統合は重要な未解決問題である。我々はヒト前頭前野周辺の計算論的モデルの構築を目指しており、それに向けて再帰的にサブルーチン呼び出しが可能な階層型強化学習アーキテクチャ

*連絡先：産業技術総合研究所
茨城県つくば市梅園 1 − 1 − 1 中央第 1
E-mail: y-ichisugi@aist.go.jp

RGoal を提案している [7][8][10]。我々はこれまでに、RGoal にわずかな拡張を加えることで、記号推論の一例である定理証明ができることを示した [9]。本稿はその方法をより具体化し、環境の中で動く知的エージェントが正しい推論規則を経験から獲得するための手法を提案する。提案手法は、推論規則の「正しさ」を「正解に到達するまでのコストの低さ」で代用し、それを強化学習によって学習する。学習が進むにつれてエージェントが行う推論はより妥当に、より効率的になる。推論規則を RGoal のサブルーチンの形で実現することにより、マルチタスク環境での学習の加速や、未経験のタスクへの対処が可能になる。

提案手法が前提とする知的エージェントのアーキテクチャを単純化して確率的生成モデルの形で表現したものを図 1 に示す。推論機構はワーキングメモリの値を参照・更新することができるが、センサーからの入力と矛盾する状態には更新できないものとする。この制約が、推論機構を環境にグラウンディングさせる。

なお、ヒトが持つ知識には宣言的知識と手続き的知識があるが、本稿が扱うのは「記号推論を行うための手続き的知識」である。

本論文は以下のような構成になっている。まず 2 章で提案手法が前提とする RGoal アーキテクチャとテーブル圧縮手法について説明する。その後 3 章で提案手法、4 章で実装の詳細、5 章で実行例について述べ、6 章で議論を行う。7 章で関連研究について述べ、最後に 8 章でまとめと今後について述べる。

2 階層型強化学習 RGoal

RGoal は再帰的なサブルーチン呼び出しが可能な階層型強化学習アーキテクチャである。RGoal では、サブルーチンは「任意の状態からある 1 つの状態（**サブゴール**）に向かう方策」と定義される。エージェントは過去の経験からの学習結果を用いて、合理的にサブゴールを設定し、それに向けて行動する。また、サブルーチンの組み合わせで未経験のタスクを解くことも可能にする。

提案手法は、行動価値関数が [8] で述べた方法で圧縮表現されることを前提とする。この方法では行動価値関数のテーブルをパターンと値のペアからなる**行動ルール**の集合で表現し、パターンマッチにより行動を選択する。パターンの処理に多少コストはかかるものの、テーブルサイズを大幅に減らすことで、汎化性能を向上させることを意図している。このパターンマッチを行う回路はベイジアンネットで実現可能である。脳内の神経回路でもおそらく実現は容易であろう。実際、前頭前野外側部で動作順序をパターンで抽象化した動作カテゴリとして表現していることが示されている [4]。

3 提案手法

3.1 問題とその解決方針

記号的な推論を行うための規則（以下、**推論規則**と呼ぶ）を経験から獲得するにはいくつか本質的に難しい問題がある。

まず第一に、解の探索範囲が広すぎて、事前知識なしでの獲得は現実的に難しいという問題がある。生物が生き残るためには、むやみに解を探索するのではなく、生得的知識を使って探索範囲を絞る必要がある。

第二に、学習に必要な「正解」が必ずしも得られないという問題がある。推論の大きな目的の 1 つは、環境の見えない状態の推定である。例えば戸棚の中の状態を推論しようとするのは戸棚の中が見えないからである。もし推論結果の正解が見えているのなら、わざわざ推論をする必要がないし、逆に見えていないのなら推論の結果が正しいかどうかわからないので、正しい推論方法が学習できない、という矛盾がある。

我々はこの 2 つの問題を次のように解決する。

第一の問題（解の探索範囲が広すぎる）に対しては、学習を「**正しい推論規則の候補の獲得**」と「**推論規則の価値の学習**」の 2 段階に分けて知識を獲得するというシナリオを提案する。1 段階目の候補の獲得は、注意の機構を用いたヒューリスティックスや、言語を通じた獲得を想定している。この機構は未実装であるが、実現方針について 3.2 節で簡単に述べる。本稿では、理論的な妥当性の議論がしやすい 2 段階目を主に扱う。

第二の問題（正解が必ずしも得られない）に対しては、エージェントの振る舞いを**通常モード**と**推論訓練モード**に適宜切り替えることで解決する方法を提案する。この方法は、ヒトの脳が予測と観測の不一致を検出するという振る舞いにヒントを得ている。このモードの切り替えも未実装であるが、実装方針を 3.4 節で簡単に述べる。本稿では推論訓練モードについての詳細を述べる。推論訓練モードは上で述べた「推論規則の価値の学習」を行うモードである。

3.2 正しい推論規則の候補の獲得の方針

ヒトは少ない経験から様々な推論規則を獲得する。ただし、その推論規則が本当に正しいかどうかはすぐにはわからない。正しいかどうかは、長い時間をかけて実世界での経験をもとに検証する必要がある。したがって、少ない経験から獲得される推論規則は、あくまで「正しい推論規則の候補」にすぎない。

このような「必ずしも正しくなくてもよい推論規則の獲得」は、目立つ刺激や目立つ事象に注目することで容易に実現可能であろう。例えば「カラスを見た。そばにカメがいた。」という経験から「カラスがいたらそ

ばにカメもいる。」という推論規則が獲得できる。環境にもよるが、この推論規則はもちろん正しいとは限らない。なお、生物にとって「注意を引くべきものは何か」は、知識獲得の性能を決める重要な生得的知識である。工学的には、神経科学的知見や我々自身の日常的な経験を参考に設計することが可能であろう。

言語を通じた知識の獲得も、ヒトの知識獲得の重要な手段である。例えば「隣の家の太郎君が犬をかわいがっていたよ。」という発話を耳にしたら、その文の内容を一般化することで「太郎君は犬が好きである。」「隣の家の人はみんな犬が好きである。」「太郎君は動物が好きである。」といった様々な推論規則を獲得することができる。もちろんこれらも必ず正しいとは限らない。例えば太郎君は犬は好きでもすべての動物が好きとは限らない。

3.3 推論規則の価値の学習

提案手法では、推論規則の「正しさ」を「正解に到達するまでのコストの低さ」で代用し、コスト（価値）を強化学習によって学習する。具体的には、まず推論の正解を用意し、推論の結果が正解でない場合は推論を最初からやり直す、という動作を繰り返す。

例えば今「チョコレートがない」という命題が成り立っているならば、その命題をゴールにした推論（証明探索）を開始する。推論の結果「チョコレートがあるはず」という結果が得られたらまた最初から推論をやり直す。これを繰り返せば、今の状況で「チョコレートがあるはず」という結果を導くことに関与する状態行動対の価値が下がっていき、やがて「チョコレートがない」という正しい推論をするようになると期待できる[1] [2]。

迷路に例えると次のようになる。「正解が得られた状態」が迷路のゴールであり、「間違った推論結果が得られた状態」は迷路の壁にぶつかった状態である。もし壁にぶつかったら、エージェントはスタート地点に戻されるものとする。各状態の価値は、ゴールに到達するまでに必要な移動コストの期待値（方策オフ型で学習する場合は移動コストの最小値）である。スタート地点におけるゴールまでのコストの期待値を a 、行動 1 ステップごとのコストを -1 とすると、任意の地点において壁にぶつかるという状態行動対の価値は $a-1$ となる[3]。状態行動対の価値の学習が終われば、エージェントは価値の高い行動を greedy に選択することでもっとも小さい移動コストでゴールにたどり着ける。

なお、前に述べたように、学習するには推論の正解が得られていなければならないという問題がある。その解決方法は次の節で述べる。

3.4 通常モードと推論訓練モード

エージェントは通常モードと推論訓練モードの 2 つの状態を持つ。

通常モードでは、エージェントは通常の強化学習と同様に、自分が持つ行動価値関数から導出される方策に従って行動を行う。この間、行動価値関数は通常の強化学習と同じ方法で学習される。しかし、これだけでは間違った推論を行う状態行動対の価値が下がることはない。（詳しい理由は 6.1 節で述べる。）

通常モードの実行中に予測と観測事実との矛盾が検出されたら、推論訓練モードに移行する。推論訓練モードでは、前節で述べた方法で学習を行う。学習に必要な「正解」は、たった今、予想に反して観測したその命題そのもの用いる。例えば「きのうチョコレートがあったのに今チョコレートがない」ということが予想外の事実であれば「チョコレートがない」という命題をゴールにして推論を繰り返す。そしてある程度学習が進んだところで通常モードに戻る。

4 推論訓練モードの実装

4.1 環境モデル

本稿では環境のモデルを、環境が取り得る状態の集合として表現する。また、ある瞬間の環境の状態は、その時に成り立っている述語の集合で表現する。本稿では、述語とは述語名 P と 1 個の引数 a を用いて P(a) と表現される命題であると定義する。例えば環境 E が状態 $\{P(1), Q(2)\}$ と状態 $\{P(2), Q(1)\}$ のどちらかを取り得るなら、

$$E = \{\{P(1), Q(2)\}, \{P(2), Q(1)\}\}$$

と表現する。

時刻・場所が異なる複数のイベントを同時に表現するには、例えば自然言語の形式意味論で使われる neo-Davidsonian 形式 [1] を用いる方法が考えられるが、そのような拡張は将来の課題とする。

[1] なお、ここで正解に到達するために必要な推論規則は 3.2 節で述べた方法ですでに獲得済みだと暗に仮定している。もし必要な推論規則を持ち合わせていなければ、いくら推論を繰り返しても正解にたどり着けない。それは、ゴールにいたる経路のない迷路を解くようなものである。知的エージェントを実環境で頑健に動くものにするためには、適宜あきらめて推論訓練モードを抜ける機構も用意する必要があるだろう。

[2] 実際には観測の方が間違っている場合もあり得る。おそらく生物は、予測と一致するまで観測の方も繰り返すだろう。それにより間違った観測方法の価値が下がり、予測通りの正しい状態が観測できるようになるかもしれない。

[3] 不正解なら負の報酬を与えてエピソードを終了するというやり方もあり得るが、定性的には同じであろう。

4.2 タスク

エージェントにはエピソードごとに推論すべき命題がゴールとして与えられる。エージェントは、環境において成り立っているゴール以外の命題からゴールの命題を証明しようとする。

環境が前節で例に用いた E である場合、エージェントは環境において成り立っている命題 $P(a)$ を観測し、その引数の値を使って $Q(b)$ を推論することができる。例えば下記の 2 つは、この環境において正しい推論をする推論規則の例である。

「 P(1) ならば Q(2) である。」
「 P(2) ならば Q(1) である。」

一方下記の 2 つは、この環境において正しくない推論規則の例である。

「 P(1) ならば Q(1) である。」
「 P(2) ならば Q(2) である。」

今回の実装では、エージェントには推論規則の集合が、後述の行動ルール集合という形であらかじめ与えられる。推論エピソードを繰り返すことで、正しくない推論規則の価値が下がり、正しい推論規則だけが選択されるようになることが、学習の目的である。

4.3 ワーキングメモリ

エージェントは環境の状態の観測やワーキングメモリの値の参照・更新を繰り返すことで、ゴールに向けた推論を進めていく。

ワーキングメモリの値は、述語の集合である。以前提案した定理証明機構 [9] と同様に、述語の集合は、実装上はフラットな固定長のベクトルに符号化される。(この表現方法の神経科学的妥当性については 6.2 節で述べる。)

ワーキングメモリの値の参照は 4.5 節で述べる行動ルールの選択の際に行われ、更新は次節で述べる Set 命令の実行によって行われる。環境の状態の観測もこの Set 命令で行われる。

4.4 行動

エージェントが取り得る行動は、Call(m), Fail, Set(m) のいずれかとする。今回の実装では環境に対する行動出力はなく、いずれの行動もエージェントの内部状態のみを変更する。

Call(m) はサブルーチン呼び出しである。現在のサブゴール g をスタックに積んで、m を新しいサブゴールとする。その後の実行でワーキングメモリの値が g になれば、サブゴールはスタックから取り出された値に再設定される。

Fail は状態のリセットである。現在のスタックの内容を破棄し、ワーキングメモリとサブゴールを初期値にリセットする。

Set(m) は環境の状態の観測またはワーキングメモリの値の更新である。提案手法ではワーキングメモリの値に含まれる命題が、環境において見えている状態と矛盾してはいけないと仮定している。(ワーキングメモリの値は、環境において成り立っているとエージェントが信じている述語集合の部分集合となる。) そのため、Set 命令の振る舞いは複雑である。

Set の引数 m は述語の集合を表すパターンだが、m に含まれるすべての述語を、環境において成り立っている同じ述語名を持つ述語とパターンマッチさせた結果に置き換えたものが、新しいワーキングメモリの値となる。例えば環境において述語 $P(1)$ が観測可能であり、m がパターン $P(\_)$ (_ は任意の値とマッチするワイルドカード) であれば、ワーキングメモリは $P(1)$ となる[4]。しかしもし m が $P(2)$ であったならば、上で述べた行動 Fail と同じ動作をする。(これを「実行が fail する」ということにする。) もし Set の対象となる述語 P が観測不可能であれば Set は無条件に成功する。例えば m が $P(2)$ であればワーキングメモリも $P(2)$ となる。

4.5 行動ルール

状態行動対をパターンを用いて抽象化したものを行動ルールと呼ぶ。1 つの推論規則は 1 つ以上の行動ルールを用いて実現される。各行動ルールは価値を持っており、強化学習によって学習される。

本稿では行動ルールを rule(s,g,a) という記法で記述する。ただし s は現在のワーキングメモリの値、g はサブゴール、a は行動を表す。

行動選択は、以前提案した手法 [9] と同様に、以下のように行う。まず、現在のワーキングメモリとサブゴールにパターンマッチ可能なパターン s, g を持つ行動ルール rule(s, g, a) を 1 つ選択する。その時の変数の値の割り当てをパターン a に適用した結果を、次に実行すべき行動とする。例えば現在のワーキングメモリの値が P(1)、サブゴールが Q(_) で、それらとパターンマッチ可能な行動ルール

rule(P(X), Q(_), Set(R(X)))

が選択されたとする。このとき変数の値の割り当ては X=1 となるため、次に実行すべき行動は Set(R(1)) となる。

選択可能なルールが複数ある場合は「最も特殊なパターン」を持つルールを選択する。選択可能な「最も特殊なパターン」を持つルールが複数ある場合、今回の

[4] 正確にはワーキングメモリの値は述語の集合 $\{P(1)\}$ だが混乱がない限り省略して $P(1)$ と書く。

実装では softmax 関数で得られる確率分布を用いて、価値の高いルールほど高い確率で選択する。

4.6 実行が fail した場合の学習則

実行が fail した場合の価値の学習則は以下のように導出できる。まず、拡張状態行動空間 [7][10] 上での Sarsa による学習則は次のようになる。

$$Q_G((s,g),a) \leftarrow Q_G((s,g),a) \\ +\alpha(r + Q_G((s',g'),a') - Q_G((s,g),a)) \quad (1)$$

RGoal ではスタックの中身が $g_1, g_2, \cdots, g_n$ であるとき、エージェントは $s \rightarrow g \rightarrow g_1 \rightarrow g_2 \rightarrow \cdots \rightarrow g_n$ という経路でゴール g_n まで移動するはずなので下記の式が成り立つ。

$$Q_{g_n}((s,g),a) = Q(s,g,a) + V(g,stack) \quad (2)$$

ただし、状態 s, サブゴール g におけるエージェントの方策を $\pi((s,g),a)$ とすると

$$\begin{aligned} V(g,stack) &\equiv V_{g_1}(g) + V_{g_2}(g_1) + \cdots + V_{g_n}(g_{n-1}) \\ V_g(s) &\equiv \Sigma_a \pi((s,g),a) Q(s,g,a) \end{aligned} \quad (3)$$

である。なお、実行が fail したらスタックは破棄されるが、スタックが空の時は $V(g,stack) = 0$ となる。

式 (1)(2) より、 fail を引き起こしたルールの価値の学習則は以下のようになる。

$$Q(s,g,a) \leftarrow Q(s,g,a) \\ +\alpha(r + Q(s',g',a') - Q(s,g,a) - V(g,stack)) \quad (4)$$

となる。

なお、fail したことに特別なペナルティを与えたい場合、負の値を持つ定数 R^F を報酬として与えればよい。

本章で述べた推論訓練モードのアルゴリズムの疑似コードを図 2 に示す[5]。

5 実行例

5.1 実行例 1：間違った推論規則の価値が下がる例

学習により間違った推論をしなくなっていく例を示す。

環境モデルは $E = \{\{P(1),Q(2)\},\{P(2),Q(1)\}\}$ 、ゴールは $Q(a)$ （ただし a はエピソード開始時に成り立っている述語 Q の引数）とする。

[5]Java ソースコードを下記 URL で公開予定。
https://staff.aist.go.jp/y-ichisugi/besom/InfLearn20200220.zip

図 3 はエージェントに与えられる行動ルールの集合である。ルール 1 〜 4 は正しい推論を行うための行動ルール、ルール 5， 6 は間違った推論を行う行動ルールである。

学習は以下の設定のもとで行う。（このあとの実行例も同様である。）学習率は $\alpha = 0.01$、softmax 関数を用いた行動選択における逆温度は $\beta = 1$、 1 ステップあたりのコストは $R^C = -1$、fail したときのペナルティは $R^F = 0$ とした。環境の状態は、各エピソードごとに、集合 E から 1 つランダムに選んだ述語の集合が、エピソード実行中にずっと成り立っているとする。エピソード開始時のワーキングメモリは空の状態（どの述語も表現されていない状態）である。

推論訓練モードにおける実行は次のように進む。いま、環境の状態が $\{P(1),Q(2)\}$ 、ゴールは $Q(2)$ とする。初期状態にマッチするのはルール 1 のみであり（ANY は任意の状態にマッチするものとする）、Call(P(_)) が実行され、サブゴールは P(_) になる。次にルール 4 が選択されて Set(P(_)) が実行され、環境の状態を観測した結果、ワーキングメモリの値は $P(1)$ に更新される。ここでサブゴール P(_) が達成されたので、サブゴールは Q(2) に戻る。次に選択可能なのはルール 2 と 5 であり、softmax 関数を使って価値に応じてどちらかが確率的に選択される。 2 が選択されればゴールが達成されエピソードは終了する。 5 が選択されれば Set 命令は fail し、ルール 1 から実行をやり直すことになる。

図 4 は、初期状態から正解に至るまでの平均実行ステップ数を 100 エピソードごとにプロットしたものである。学習が進むにつれ、ステップ数が短くなっている。最終的にはほぼ 4 ステップで推論が終わっているが、これは間違った推論をほとんどしなくなったことを意味する[6]。

図 3 の右側の数値は、1000 エピソード学習後の各行動ルールの価値である。ルール 5， 6 の価値が 2， 3 の価値よりもそれぞれ低いため、間違った行動ルールが選択されにくくなっている。

なお、ルール 5、6 の価値はスタート地点であるルール 1 の価値に -1 を足した値になるはずだが、この段階ではそこまで価値は下がっていない。これは、価値が下がるほどそのルールが選択されなくなり、価値の低下が停滞するためである。

1: **procedure** EPISODE(S, G)
2: $stack \leftarrow empty;\ s \leftarrow S;\ g \leftarrow G$
3: Choose a rule which matches (s, g) and get a from it.
4: **while** not ($stack$ is $empty$ and a is Return) **do**
5: # Take action.
6: **if** a is Return **then**
7: $s' \leftarrow s;\ g' \leftarrow stack$.pop(); $r \leftarrow 0$
8: **else if** a is Call(m) **then**
9: $stack$.push(g); $s' \leftarrow s;\ g' \leftarrow m;\ r \leftarrow R^C$
10: **else if** a is Set(m) **then**
11: **if** m' is the result of pattern matching between pattern m and the environment's state **then**
12: $s' \leftarrow m';\ g' \leftarrow g;\ r \leftarrow R^C$
13: **else**
14: # If pattern matching failed.
15: $stackV \leftarrow V(g, stack)$
16: $stack \leftarrow empty;\ s' \leftarrow S;\ g' \leftarrow G;\ r \leftarrow R^C + R^F$
17: **else if** a is Fail **then**
18: $stackV \leftarrow V(g, stack)$
19: $stack \leftarrow empty;\ s' \leftarrow S;\ g' \leftarrow G;\ r \leftarrow R^C + R^F$
20: **else**
21: Error
22: # Choose action.
23: **if** s' matches g' **then**
24: $a' \leftarrow$ Return
25: **else**
26: Choose a rule which matches (s', g') and get a' from it.
27: # Learn.
28: **if** a is Return **then**
29: # Do nothing.
30: **else if** last action failed **then**
31: $Q(s, g, a) \leftarrow Q(s, g, a) + \alpha(r + Q(s', g', a') - Q(s, g, a) - stackV)$
32: **else**
33: $Q(s, g, a) \leftarrow Q(s, g, a) + \alpha(r + Q(s', g', a') - Q(s, g, a) + V_g(g'))$
34: $s \leftarrow s';\ g \leftarrow g';\ a \leftarrow a'$

図 2: 推論訓練モードのアルゴリズムの疑似コード。以前提案した定理証明機構 [9] とほぼ変わらないが、環境を観測する機構と価値の学習機構を追加した。学習機構は fail の学習則を追加した点を除けば通常の RGoal[10]（スタックあり版）と同じである。R^C は 1 ステップあたりのコスト、R^F は fail したときのペナルティである。

1:	rule(ANY, Q(_), Call(P(_)))	-3.2281303
2:	rule(P(1), Q(_), Set(Q(2)))	-0.9949911
3:	rule(P(2), Q(_), Set(Q(1)))	-0.99138117
4:	rule(ANY, P(_), Set(P(_)))	-0.999997
5:	*rule(P(1), Q(_), Set(Q(1)))	-3.1717572
6:	*rule(P(2), Q(_), Set(Q(2)))	-2.9107559

図 3: 実行例 1 でエージェントに与えられる行動ルールの集合。与えられた環境のもとで必ずしも正しくないルールには星印 (*) を付けて示した。ルールの右側の数字は 1000 エピソード学習後の価値である。（以降の実行例も同様。）間違った選択肢 5,6 の価値は、正しい選択肢 2,3 と比較して低い価値になっている。

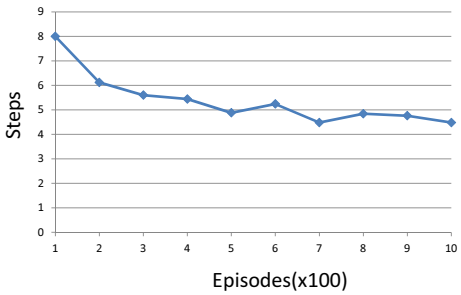

図 4: 実行例 1 におけるエピソード終了までの平均実行ステップ数の変化。学習が進むにつれ理想的な値 4 に近づいている。

5.2 実行例２：遠回りの推論が使われなくなる例

学習により推論が効率化される例を示す（図 5）。環境モデルは

$$E = \{\{P(1), Q(1), R(1)\}, \{P(2), Q(2), R(2)\}\}$$

であり、推論のゴールは $R(a)$ である。

推論の手順は 2 つ与えてあり、P(X) から R(X) を直接推論する方法（ルール 1 → 7 → 2 の順で実行が進む）と、まず P(X) から Q(X) を推論し、次に Q(X) から

1:	rule(ANY, R(_), Call(P(_)))	-2.9817963
2:	rule(P(X), R(_), Set(R(X)))	-0.9982739
3:	rule(ANY, R(_), Call(Q(_)))	-4.2641506
4:	rule(Q(X), R(_), Set(R(X)))	-0.9749899
5:	rule(ANY, Q(_), Call(P(_)))	-2.80906
6:	rule(P(X), Q(_), Set(Q(X)))	-0.9749899
7:	rule(ANY, P(_), Set(P(_)))	-0.9999568

図 5: 実行例 2 の行動ルール集合と学習後の価値

[6]学習では Return はコスト 0 として扱われているが、このグラフのステップ数には Return 命令の実行回数も含まれている。

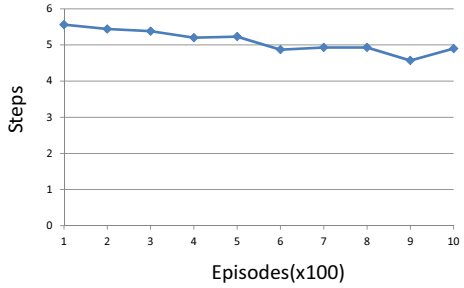

図 6: 実行例 2 におけるエピソード終了までの平均実行ステップ数の変化

R(X) を推論する方法（3 → 5 → 7 → 6 → 4 の順で実行が進む）のどちらかが確率的に選択される。どちらの方法でも正しいが、学習の結果、より短いステップで推論が終わるルール 1 の方がルール 3 よりも価値が高くなり、より高い頻度で選択されるようになる。図 4 は平均実行ステップ数の変化であり、学習が進むにつれ理想的な値 4 に近づいている。

なお、このような推論の最適化には推論の正解は必要ないため、推論訓練モードだけでなく通常モードでも学習が進むはずである。

5.3 実行例３：間違ったサブルーチンを呼び出す行動ルールが使われなくなる例

1:	rule(ANY, R(_), Call(P(_)))	-2.9989986
2:	rule(P(X), R(_), Set(R(X)))	-0.9999568
3:	rule(ANY, R(_), Call(Q(_)))	-5.0751033
4:	rule(Q(X), R(_), Set(R(X)))	-3.8932161
5:	rule(ANY, P(_), Set(P(_)))	-0.9999568
6:	*rule(ANY, Q(_), Set(Q(2)))	-0.98258275

図 7: 実行例 3 の行動ルール集合と学習後の価値

学習環境によっては間違った推論をするサブルーチンの価値が下がらない場合もあるが、その場合でもサ

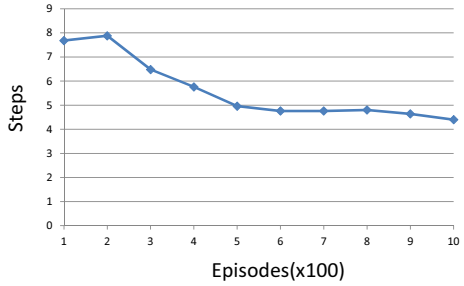

図 8: 実行例 3 におけるエピソード終了までの平均実行ステップ数の変化

ブルーチンを呼び出す側の価値が下がり、全体として推論が正しくなる例を示す（図 7）。

環境モデルは $E = \{\{P(1), Q(1), R(1)\}\}$ であり、推論のゴールは $R(a)$ である。また、命題 $Q(1)$ はエージェントから観測できないとする。

推論の手順は 2 つ与えてあり、P(X) から R(X) を推論する方法（ルール 1 → 5 → 2 の順で実行が進む）と Q(X) から R(X) を推論する方法（ 3 → 6 → 4 の順で実行が進む）のどちらかが確率的に選択される。ルール 6 は正しくない推論をするサブルーチンであるが命題 $Q(1)$ が観測不能なのでそこでは fail せず実行が進み、ルール 4 で Set(R(2)) を実行したところで fail して最初に戻る。

ルール 6 は間違った推論をするにも関わらず、この学習環境では正解を得る機会がないので価値が下がらない。しかしこのサブルーチンを呼び出す側のルール 3 の価値は下がるので、その結果、正しい推論だけが行われるようになる（図 8）。

5.4 実行例４：決めつけの方が価値が高い例

1:	rule(ANY, Q(_), Call(P(_)))	-2.7919052
2:	rule(P(1), Q(_), Set(Q(2)))	-0.8824299
3:	rule(P(2), Q(_), Set(Q(1)))	-0.9446731
4:	rule(ANY, P(_), Set(P(_)))	-0.9934953
5:	*rule(ANY, Q(_), Set(Q(1)))	-1.987416

図 9: 実行例 4 の行動ルール集合と学習後の価値

提案手法は推論規則の正しさを推論コストの低さで代用しているので、学習環境によって正しくない推論規則の価値が下がらずに残ってしまうことも起こり得る。この節ではその例を 1 つ示す（図 9）。

環境モデルとゴールは実行例 1 と同じであるが、環境の状態は $\{P(1), Q(2)\}$ である確率が 0.2 、$\{P(2), Q(1)\}$ である確率が 0.8 であるとする。この場合、素直に P(X) を観測してから Q(Y) を推論するよりも、まず Q(1) が成り立っていると決めつけた方が平均的に速く正解にたどり着ける。

ルール 5 は無条件に値 Q(1) を推論する正しくない行動ルールだが、学習の結果、正しい推論を行うルール 1 よりも高い価値を持っている。

これは推論訓練モードを使った価値の学習方法が不完全な場合があることを示す一例である。ただし、ペナルティ R^F に十分低い値を与えるとルールの 5 の価値は下がり、選択されなくなる。したがって実用上はほぼ問題ないと思われる。

5.5 実行例５：２つの命題から値を推論する例

これまでの例は 1 つの命題からもう 1 つの命題を推論する例だったが、２つの命題から推論することもできる（図 10）。図 10 では、 AND(P,Q) という記法を用いているが、これは命題 P と Q の両方が成り立っていることを表す。

環境モデルは

$$E = \{ \{P(1), Q(1), R(1)\}, \{P(1), Q(2), R(2)\}, \{P(2), Q(1), R(2)\}, \{P(2), Q(2), R(1)\} \} \quad (5)$$

であり、推論のゴールは $R(a)$ である。

実行はルール 1 → 9 → 2 → 3 → 1 0 → 4 と進んだ後、そのときのワーキングメモリの値に応じて 5 ～ 8 の中のいずれかが実行される。

ルール 4 はそれ単独で見ると常に正しいとは言えない（Q(Y) から P(X) ∧ Q(Y) を推論している）が、与えられた行動ルール集合のもとでは全体の推論は正しく動作する。これは、ルール 4 のゴール部分が呼び出し元の文脈の情報を持っているからである。ルール 4 が正しい推論をするかどうかは、呼び出し側が適切な文脈でルール 4 を呼び出すかどうかに依存する。

この例は、ゴールが必ずしも単独の述語ではない例になっている。Prolog 言語では個々のルールはホーン節の形に限られるが、この機構ではそのような制限はない。

5.6 実行例６：例外的な状況を処理するルールの例

単独では正しくないルールが存在していても、全体としては正しく動作する例をもう 1 つ示す（図 11）。環境モデルは

$$E = \{ \{P(1), Q(1)\}, \{P(2), Q(2)\}, \{P(3), Q(3)\}, \{P(4), Q(4)\}, \{P(5), Q(10)\} \} \quad (6)$$

であり、推論のゴールは $Q(a)$ である。

1:	rule(ANY, R(_), Call(P(_)))	-6.830637
2:	rule(P(X), R(_), Call(AND(P(X),Q(_))))	-4.8854413
3:	rule(P(X), AND(P(X),Q(_)), Call(Q(_)))	-2.9989986
4:	*rule(Q(Y), AND(P(X),Q(_)), Set(AND(P(X),Q(Y))))	-0.9999568
5:	rule(AND(P(1),Q(1)), R(_), Set(R(1)))	-0.90480024
6:	rule(AND(P(1),Q(2)), R(_), Set(R(2)))	-0.91729563
7:	rule(AND(P(2),Q(1)), R(_), Set(R(2)))	-0.935021
8:	rule(AND(P(2),Q(2)), R(_), Set(R(1)))	-0.9156164
9:	rule(ANY, P(_), Set(P(_)))	-0.9999568
10:	rule(ANY, Q(_), Set(Q(_)))	-0.9999568

図 10: 実行例 5 の行動ルール集合と学習後の価値

1:	rule(ANY, Q(_), Call(P(_)))	-2.8419564
2:	*rule(P(X), Q(_), Set(Q(X)))	-0.9996545
3:	rule(P(5), Q(_), Set(Q(10)))	-0.8751221
4:	rule(ANY, P(_), Set(P(_)))	-0.9999568

図 11: 実行例 6 の行動ルール集合と学習後の価値

ルール 2 は一般的な（デフォルトで使われる）推論規則、ルール 3 は例外的な状況で使われる推論規則である。環境で P(5) が成り立っているときルール 2 とルール 3 の両方がマッチするが、RGoal では「最も特殊なルール」が選択されるため、 ルール 3 が選択される。結果として、すべての環境の状態において、推論は正しく行われる。

6 議論

6.1 推論訓練モードの理論的妥当性

強化学習エージェントは能動的に動作するので、学習サンプルは偏ったものが得られる。推論機構に対しては、このことが最悪の方向に働く。エージェントは「正解がないときだけ推論する」という行動を取るので、推論機構の学習に役立つ学習サンプルは一切得られないことになる。推論訓練モードはこの学習サンプルの極端な偏りを補正する効果がある。推論機構の学習に役立ちそうな状況を検出した瞬間に推論訓練モードに切り替えることで、学習サンプルは少なくとも最悪の偏りからは脱することになる。

6.2 ワーキングメモリと大脳皮質

脳ではワーキングメモリは大脳皮質の連合野で表現されるものと想定している。より具体的には、大脳皮質は n 個のマクロコラムを使って n 引数 ($n >= 1$) の

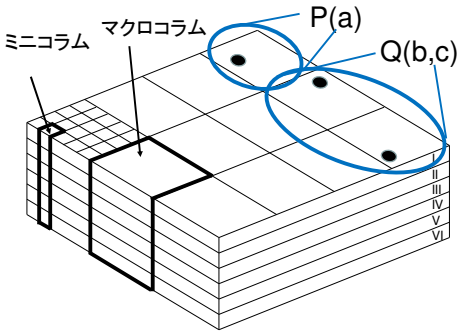

図 12: 大脳皮質のコラム構造と述語の表現

表 1: 大脳皮質とベイジアンネットと述語の対応

大脳皮質	ベイジアンネット	述語
マクロコラム	多値確率変数	述語の引数
ミニコラム	確率変数の値	述語の引数の値
ニューロン発火	事後確率	真偽値
矛盾検出	同時確率が 0	矛盾

述語 1 個を表現し、ミニコラムの活動が引数の値を表現すると想定する（図 12）。

表 1 は、大脳皮質とベイジアンネットを対応付ける BESOM モデル [5] を踏まえて、大脳皮質とベイジアンネットと述語の対応を表にまとめたものである。

提案手法では、予測と観測の不一致の検出機構を 2 つの目的で使っている。 1 つは通常モードから推論訓練モードへの切り替えタイミングの検出であり、もう 1 つは推論訓練モードにける間違った推論結果の検出である。大脳皮質がベイジアンネットであれば、この検出は同時確率が 0 かどうかを見ればよく、神経回路でも容易に実現可能ではないかと思われる。

7 関連研究

記号推論と強化学習と組み合わせる試みとして例えば RRL(Relational Reinforcement Learning)[2]、rlCoP[6]、Soar-RL[3] があり、いずれも本研究と同様に経験から

の知識獲得や推論の効率化を目標としている。提案手法は階層型強化学習を用いることによりマルチタスク環境での効率的学習を目指している点と生物学的妥当性を重視する点に特徴がある。

筆者らが提案した RGoal [7][10] は思考モードと呼ぶ、一種の演繹推論の機構を有している。本稿で述べた通常モードと推論訓練モードはこの思考モードとは直交した概念であり、おそらく組み合わせて使うことができる。

プロダクションシステムは記号ＡＩや認知アーキテクチャの実装方法として古くから使われている。エージェントはワーキングメモリの状態に応じて、与えられたルールの中から 1 つを選択し実行するということを繰り返す。本稿で述べたアーキテクチャはプロダクションシステムの基本構造を踏襲している。しかし提案手法ではワーキングメモリの値は自由に更新できず、環境の状態と矛盾してはならないという制約がある。これにより推論規則の意味を環境にグラウンディングさせるとともに、アーキテクチャの神経科学的妥当性を高めている。

8　まとめと今後

環境の中で動く知的エージェントが正しい推論規則を経験から獲得するための手法を提案した。提案手法は、推論規則の「正しさ」を「正解に到達するまでのコストの低さ」で代用し、それを階層型強化学習 RGoal によって学習する。提案手法は汎用人工知能の自律的な知識獲得の基本原理の 1 つになると考えている。

知的エージェントの行動（環境への働きかけ）と推論（脳内へ状態への働きかけ）が報酬期待値最大化という 1 つの目的関数に統合されるため、行動と推論の両方を合理的に組み合わせた振る舞いができるようになると期待できる。

提案手法は他者の心の状態の推定や言語理解、合目的的な発話計画などのモデルの実現につながる。それを発展させることで、会話を深く理解する対話システムへの応用も可能であろうと思われる。

謝辞

本研究に関して議論をしていただいた竹内泉氏に感謝いたします。

本研究は JSPS 科研費 JP18K11488, JP18K18117 の助成を受けたものです。

参考文献

[1] Parsons, T., Events in the semantics of English, MIT Press, 1990.

[2] Saso Dzeroski, Luc De Raedt, and Kurt Driessens, Relational reinforcement learning. Machine Learning, 43, pp.7–52, 2001.

[3] Shelley Nason, John E. Laird, Soar-RL: integrating reinforcement learning with Soar, Cognitive Systems Research 6, 5159, 2005.

[4] Keisetsu Shima, Masaki Isoda, Hajime Mushiake and Jun Tanji, Categorization of behavioural sequences in the prefrontal cortex, Nature, 445, 315-318, 2007.

[5] Yuuji ICHISUGI, The Cerebral Cortex Model that Self-Organizes Conditional Probability Tables and Executes Belief Propagation, In Proc. of IJCNN 2007, pp.1065–1070, Aug 2007.

[6] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, Mirek Olšák, Reinforcement Learning of Theorem Proving, In Proc. of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.

[7] 一杉裕志, 高橋直人, 中田秀基, 佐野崇, RGoal Architecture:再帰的にサブゴールを設定できる階層型強化学習アーキテクチャ, 第 9 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2018.

[8] 一杉裕志, 高橋直人, 中田秀基, 佐野崇, 単一化の機構を利用した階層型強化学習のテーブル圧縮手法の検討, 第 10 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2018.

[9] 一杉裕志, 中田秀基, 高橋直人, 佐野崇, 階層型強化学習 RGoal を用いた記号推論の実現手法の検討, 第 12 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2019.

[10] Yuuji Ichisugi, Naoto Takahashi, Hidemoto Nakada, Takashi Sano, Hierarchical Reinforcement Learning with Unlimited Recursive Subroutine Calls, In Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning, Lecture Notes in Computer Science, vol 11728, pp. 103–114, Springer, Cham, 2019.