人工知能学会研究会資料
SIG-AGI-015-02

脳の自律的プログラム合成機構のモデルに向けて：２層ベイジアンネットによる記号処理命令の獲得・実行機構

Towards a model of autonomous program synthesis mechanism in the brain: Acquisition and execution mechanism of symbol processing instructions by two-layer Bayesian network

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] 東洋大学 情報連携学部情報連携学科
[2] Faculty of Information Networking for Innovation And Design, Toyo University

Abstract: In this paper, we first describe the concept of the architecture of an intelligent agent that acquires knowledge from experience. Next, as one of the important elemental technologies of the architecture, we propose a method for pattern acquisition and pattern matching using two-layered Bayesian networks. This method uses a specially designed conditional probability model.

概要

本稿ではまず経験から知識を獲得する知的エージェントのアーキテクチャの構想を述べる。次にそのアーキテクチャの重要な要素技術の１つとして、２層ベイジアンネットを用いてパターン獲得・パターンマッチを行う手法を提案する。本手法は特別に設計された条件付確率モデルを用いる。

1　はじめに

強化学習を用いたプログラム合成は汎用人工知能実現に向けた有望なアプローチである [2][4][8]。我々は神経科学的知見をヒントにして、ヒトの脳のプログラム合成機構モデルの構築を進めている。これまでに、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 RGoal [9][12] を提案し、それを用いて記号推論が行えることを示した [11]。記号推論を行うためには、正しい**推論規則**の集合をエージェントが**知識**として持っている必要がある。そこで我々は推論規則の集合があらかじめ与えられているとき、個々の推論規則の正しさを近似的に**価値**（報酬期待値）に帰着させて強化学習により学習する手法を提案した [13]。推

*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

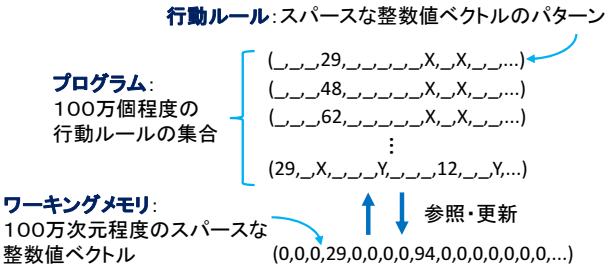

図 1: エージェントが獲得するプログラムの概念図。プログラムは整数値ベクトルのパターンの集合として表現される。

論規則の集合をプログラムとみなすならば、この機構は強化学習によりプログラムの性能（精度と実行効率）の最適化を行う機構であると言える。

しかし真に自律的に知識獲得する知的エージェントは、知識が与えられていなくても、みずからの経験から知識を獲得できなければならない。本稿では、そのための重要な要素技術として、整数値ベクトルに対するパターン獲得・パターンマッチを機械学習技術を用いて行う手法を提案する。パターンマッチやその拡張である単一化は記号ＡＩの基本となる技術である。提案手法は記号ＡＩと統計的機械学習とを融合し、双方の欠点を補完するＡＩ技術の実現につながるものである。

提案手法の実現方針は以前 [10] 示したものと同じだが、本稿では実際にベイジアンネットを実装し、動作

確認した結果を示す。また、実装することで初めて明らかになった本手法の特徴について報告する。

本稿は以下のような構成になっている。まず 2 章でプログラムの合成機構・実行機構の構想について述べた後、3 章で提案手法、4 章で実行例について述べる。5 章で関連研究について述べ、最後に 6 章でまとめと今後について述べる。

2 プログラム合成機構・実行機構の構想

2.1 プログラム合成機構

我々が仮定する知識獲得の原理の 1 つは強化学習である。環境中においてランダムに試行錯誤を繰り返し、報酬が得られた行動をより選択しやすくなるように学習を繰り返す。

しかし、汎用人工知能が動作すべき実世界においては、探索範囲が広すぎて、素朴な試行錯誤のみで有用な知識を獲得することは事実上不可能である。そこで以下のように 2 段階に分けて知識獲得させる。

1. 生得的な身体反射、他者の指示、模倣などによる行動の経験を知識獲得の素材とする。経験した行動とその結果を自己符号化器を用いた教師なし学習により圧縮・抽象化することで、汎用的な「知識」を獲得する。ただし獲得された知識は必ずしも有用なものとは限らない。

2. 得られた知識を用いて環境の中で行動し、強化学習の機構により知識の価値を学習する。この機構によりエージェントにとって有用な知識のみが高い確率で利用されるようになる。

我々は 2 段階目に関する学習手法を以前提案した [13]。本稿で提案するのは 1 段階目に関する機構である。

記号ＡＩにおける知識には手続き的知識と宣言的知識がある。 2 段階目の機構は手続き的知識に対してのみ適用可能であるが、本稿で提案する手法は両方に適用可能である。

2.2 プログラム実行機構

これまでに我々が開発してきたプログラム実行機構 [10][11][13] について、この節で簡単に説明する。

実行機構はプロダクションシステムの基本構造を踏襲している [11]。エージェントは**ワーキングメモリ**の状態を参照して**行動ルール**を 1 つ選択し、ワーキングメモリの値の更新や環境に対する運動の出力を行う（図 1）。

行動ルールの選択は、ワーキングメモリの状態に対するパターンマッチにより行う。これにより、条件分岐が行える。行動ルールの集合は強化学習における行動価値関数を圧縮・抽象化したものである。1 つの行動ルールはフラットかつ（非常に長い）固定長の**パターン**と価値の組として表現される [10]。ここではパターンは整数値または変数からなるベクトルであると定義する。例えばパターン $(X, X, 3)$ や $(X, 2, 3)$ は整数値のベクトル $(2, 2, 3)$ と**マッチ**する。変数が取り得る値は整数値のみであり、 Prolog 言語が扱うような複雑なデータ構造は直接的には扱えない[1]。入力に対し複数のパターンがマッチする場合は、「最も特殊なパターン」が選択されるものとする。もし「最も特殊なパターン」が複数ある場合は価値の高いパターンほど高い確率で選択されるものとする。詳細は 3.3 節と 3.4 節で述べる。

1 章で述べた推論規則も、行動ルールの集合として実現される [11][13]。推論とは脳内のワーキングメモリの更新を行う脳内アクションであると考える。また、ワーキングメモリの一部の値がセンサー入力と運動出力を反映するようなアーキテクチャになっていれば、脳内アクションだけで環境とのインタラクションができることになる。（計算機アーキテクチャにおける Memory-mapped I/O と同じである。）

行動ルールの選択・実行・価値の学習は RGoal [9][12] の枠組みの中で行われる。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。

ワーキングメモリの更新を行う際、更新しようとする値が環境からのセンサー入力と矛盾したら更新が失敗する仕掛けにしておく [13]。この仕掛けにより、環境のモデルと食い違った推論規則の価値が下がり、知識が環境にグラウンディングすることになる。この機構は、推論や行動が期待した通りに行われているかどうかをチェックする実行モニタリングの役割も果たすと思われる。予想外の事態が起きたときにはそれまでの実行コンテキストを破棄してあらためて行動計画を立て直すことが可能になる。これはいわばプログラムの例外処理である。

行動ルールの集合はプログラムと見なすことができる。そのプログラムを表現する「プログラミング言語」は、この節で述べたように、メモリの参照・更新、条件分岐、 I/O 、サブルーチン呼び出し、例外処理の機構を有しており、トイプログラムではない実世界での情報処理への適用可能性を秘めている。また、行動ルールはフラットなベクトルであり、機械学習技術の適用が容易である。この特徴を用いて自律的にプログラム合成するアーキテクチャができれば、汎用人工知能実

[1] 複雑なデータ構造はエピソード記憶を用いて表現することを検討中である。エピソード記憶はヒープのように動的に確保可能な記憶領域として動作する。エピソードのワーキングメモリへの読み出しは、連想記憶の機構により行われる。

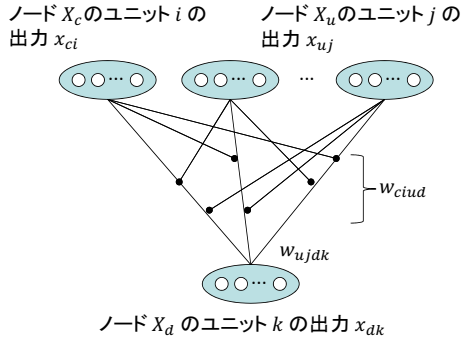

図 2: 本稿で用いる条件付確率モデルの内部構造。モデルのパラメタには 2 種類ある。1 つは他の親子ノード間の結合を制御するゲートの重み w_{ciud} 、もう 1 つは子ノードの事後確率を線形和で計算するための重み w_{ujdk} である。親ノードはゲート（黒丸で示されている）を制御することで、他の親ノードと子ノードの間の接続を切断することができる。

現に向けた大きなブレークスルーとなるだろう。本稿で提案するパターン獲得の機構は、その重要な要素技術になる。詳細は次の章で説明する。

2.3 想定する脳との対応

アーキテクチャ設計のヒントとするために想定した、脳の部位との対応関係について簡単に述べる。行動ルールの集合は背外側前頭前野 (DLPFC) の長期記憶として記憶され、ワーキングメモリの値は前頭極・感覚連合野のニューロン発火で表現されると想定している。将来はワーキングメモリの一部として海馬の機能も取り入れる。パターンマッチは大脳皮質が確率伝搬法により効率的に行う。再帰呼び出し可能な強化学習は DLPFC と大脳基底核のループ構造が実現する。

1 つの行動ルールはヒトの DLPFC の 1 つのミニコラムに対応し、ワーキングメモリの 1 つの要素は前頭極・感覚連合野の 1 つのマクロコラムに対応する。ヒトの大脳皮質のマクロコラム数・ミニコラム数の概数からおおまかに推定すると、ヒトが保持する行動ルールの数は１００万程度、1 つのワーキングメモリの要素数は１００万程度、1 つの要素が取り得る値の数は１００個程度となる（図 1）。ワーキングメモリの要素はほとんどが 0、パターンの要素はほとんどがワイルドカード（4.2 節参照）であると想定している。

3 提案手法

3.1 条件付確率モデル

この節では学習に用いるベイジアンネットの条件付確率モデルについて説明する[2]。

まずノード（確率変数）の値の one hot vector 表現を以下のよう定義する。$s+1$ 個の離散値 $0, 1, 2, \cdots, s$ を取り得るノード X の値 i は、i 番目の要素のみが 1 で他は 0 である長さ $s+1$ のベクトルで表現する。例えばノード X が取り得る値が $0, 1, 2, 3$ のとき、値 2 を表す one hot vector 表現は $(0, 0, 1, 0)$ である。

親ノードの集合を $\{X_u | u \in U\}$ 、子ノードを $X_d, d \in D$ とする。x_{uj}, x_{dk} はそれぞれノード X_u, X_d の one hot vector 表現における j, k 番目の要素とする。以下、条件付確率モデルを神経回路に見立てて、x_{uj} を「ノード X_u のユニット j の出力」のように表現する[3]。

モデルのパラメタには 2 種類ある（図 2）。1 つはある親ノード $X_c, c \in U$ のユニット i が別の親ノード X_u と子ノード X_d の間の結合を制御するゲートの重み w_{ciud} である。もう 1 つは親ノード X_u のユニット j の出力の線形和によって子ノード X_d のユニット k の出力 x_{dk} を計算するための重み w_{ujdk} である。親ノードの値が与えられたとき子ノードが $X_d = k$ になる条件付確率は下記のように、 x_{dk} で与えられるものとする。

$$P(X_d = k | \,\mathrm{pa}(X_d)) = x_{dk} \tag{1}$$

ただし $\mathrm{pa}(X_d)$ は与えられた親ノードの値の組とする。

$U \neq \phi$ すなわちノード X_d が 1 つ以上親ノードを持つとき、$k \neq 0$ に対して x_{dk} は以下の式で計算される。

$$\begin{aligned} g_{ud} &= \Pi_{c \in U} \Pi_{i \neq 0} (1 - w_{ciud}\, x_{ci}) \\ s_{dk} &= \Sigma_{u \in U} \Sigma_{j \neq 0}\, w_{ujdk}\, g_{ud}\, x_{uj} \\ x_{dk} &= s_{dk} / \max(1,\, \Sigma_{i \neq 0}\, s_{di}) \end{aligned} \tag{2}$$

g_{ud} はゲートが開いている度合いを表し、s_{dk} は正規化前の値、x_{dk} 正規化後の最終的な X_d の出力である。

$U = \phi$ （親ノードの数が 0）のときは $x_{dk}, k \neq 0$ は以下の式で計算される。

$$x_{dk} = \alpha_d / \,\mathrm{sizeof}(X_d) \tag{3}$$

ただし α_d はノード X_d が非 0 の値を取る事前確率（これをノード X_d の**活性率**と呼ぶ）、$\mathrm{sizeof}(X_d)$ はノード

[2]我々の以前の研究では疑似ベイジアンネット QBC[5][7] を使ってプロトタイピングを行っていた。本稿の条件付確率モデルは定性的に以前提案したものと同じだが、真のベイジアンネットとしての扱いがより簡単になるように定義しなおした。

[3]我々は大脳皮質の計算論的モデル構築を目指しているが、この条件付確率計算をする神経回路がそのまま大脳皮質の中に存在すると主張するわけではない。存在すると想定するものはこの条件付確率モデルのもとで確率伝搬法を実行するためのメッセージ計算をする神経回路である。

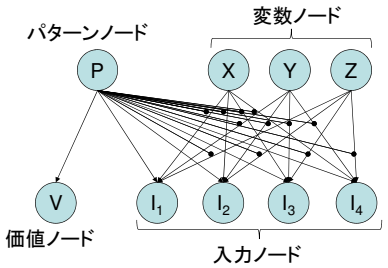

図 3: パターンマッチを行うネットワークの構造。下の層の価値ノードに観測値 1 、入力ノードに入力ベクトルを与えると、マッチするパターンと変数割り当ての結果が上の層に推論される。

X_d が取り得る非 0 の値の個数で $X_d \in \{0, 1, 2, \cdots, s\}$ のとき $\mathrm{sizeof}(X_d) = s$ である。

$k = 0$ に対しては、x_{dk} は以下の式で計算される。

$$x_{d0} \quad = \quad 1 - \Sigma_{k \neq 0}\, x_{dk} \tag{4}$$

2 層構造のネットワークにおいて上の層と下の層のノード数をそれぞれ n とすると、パラメタ数は全体で $O(n^3)$ となる。これは、一般のベイジアンネットのパラメタ数 $O(2^n)$ よりもはるかに少なく、全結合のニューラルネット $O(n^2)$ よりも大きいオーダーである。実際には重みはスパースであり非 0 のパラメタ数は非常に少なくてすむ。

以下の節では、この条件付確率モデルを用いて記号ＡＩの重要な要素技術の 1 つであるパターンマッチが自然に実現可能であることを示す。なお、我々はパターンマッチの実現のみを目標として条件付確率モデルを設計しているわけではないことを強調しておく。我々は大脳皮質のすべての領野に共通の計算論的モデルの構築を目標としている。将来的には本節で述べた条件付確率モデルを用いて、視覚野のモデルや注意のモデルなど含む多くの計算論的モデルの再現を目指している。

3.2 パターンマッチを行うネットワークの構造

図 3 のような 2 層のベイジアンネットを用いてパターンマッチを行う。このネットワークでは各ノードの役割が決められている。上の層は 1 つの**パターンノード**と複数の**変数ノード**から構成され、下の層は 1 つの**価値ノード**と複数の**入力ノード**から構成される。パターンノード P はマッチするパターンのＩＤを表現するノードで、P が取り得る値が $0, 1, \cdots, s$ ならばネットワークは s 個のパターンを保持できる。子ノードの条件付確率は、前節で説明したモデルにより計算される。ゲート制御のための重み w_{ciud} は、パターンノードが変数ノードと入力ノードとの間の結合を適切に制御するように設定される。また、結合された変数ノードと入力ノードはすべて同じ値を生成するように重み w_{ujdk} が設定される。パターンの中の定数の部分はパターンノードが重み w_{ujdk} を用いて直接生成する。価値ノードについてはこの節では説明せず 3.4 節で述べる。

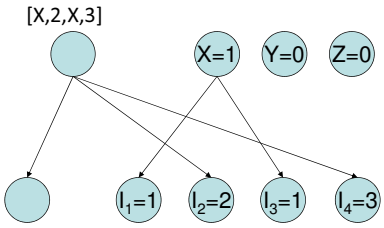

図 4: パターンマッチを行うベイジアンネットの状態の例。上の層に値が与えられれば下の層の値を、下の層に値が与えられれば上の層を値を推論することができる。

上の層に値が与えられたときに入力ノードの値が生成される過程を具体例で説明する（図 4）。まず、パターンノードの値がパターン $(X, 2, X, 3)$ のＩＤを表現しており、変数ノードの値には $X = 1, Y = 0, Z = 0$ が与えられたとする。パターンノードは変数ノード X が入力ノード I_1 と I_3 の値を生成し Y, Z は何も生成しないように結合を制御する。また、パターンノードは I_2 に対し 2 を、I_4 に対し 3 を生成する。その結果生成された 4 つの入力ノードの値は $(1, 2, 1, 3)$ というベクトルを表現することになる。

逆にパターンマッチを行う際には、価値ノードには値 1 、入力ノードには入力ベクトルの各要素の値を観測値として与え、事後確率が最大となるような上の層の値の組を推論する。推論結果のパターンノードの値はマッチしたパターン、各変数ノードの値はパターンマッチの結果における変数割り当てを表現する。

このネットワークは自己符号化器の形をしているため、原理的には下の層に入力ベクトルを訓練データとして与えることで、パターンを教師なし学習により獲得可能である。ただし学習を安定化させるためには適切な事前分布の設定が必要となる。このネットワークは多層ニューラルネットワークとは違い、個々の重みの機能が明確である。したがって、入力データの性質や獲得したいパターンの性質に応じて重みの事前分布が設計しやすいという利点がある。

3.3 最も特殊なパターンの選択

図 4 のネットワークは入力にマッチする複数パターンの中で「最も特殊なパターン」を選択するという機能を持っている。ここで「最も特殊なパターン」とは、用いる変数の種類が最も少ないパターンと定義する。例え

ば値 $(1,2,1,3)$ はパターン $(X,2,X,3)$ と $(X,2,Y,3)$ の両方にマッチするが、用いる変数の種類は前者が 1 種類、後者が 2 種類なので前者の方がより特殊なパターンである。

提案するネットワークがこのような振る舞いをするのは変数ノードの事前分布のおかげである。ここで変数ノードの活性率 α_d は比較的小さい値であると仮定する。つまり、変数ノードは値 0 を取りやすいものとする。図 4 で示したようにパターン $(X,2,X,3)$ が選択された場合は値が 0 になる変数ノードは Y,Z の 2 つだが、パターン $(X,2,Y,3)$ が選択された場合は Z のみである。値が 0 になる変数ノードの数が多いほど事後確率は大きな値になるので、前者のパターンが選択されることになる。

3.4 最も価値の高いパターンの選択

図 4 のネットワークは、入力にマッチする「最も特殊なパターン」が複数ある場合、その中で最も価値が高いパターンを選択する。

その機能を実現するために、パターンノードと価値ノードの間の結合の重みを以下のように設定する。パターン p の価値を $r_p \in [0,1]$ とすると、パターンノードの値 $P=p$ に対する価値ノードの値 $V=1$ の条件付確率を

$$P(V=1|P=p)=r_p \tag{5}$$

となるように結合の重み w_{ujdk} を設定すればよい。ベイジアンネットにおけるネットワーク全体の同時確率は個々のノードの条件付確率の積で定義されるので、このように重みを設定することで、同時確率は選択されたパターンの価値に比例する値を持つことになる[4]。

4 実行例

4.1 実装

提案手法を Julia 言語を用いて実装して動作確認した。学習には機械学習ライブラリ Flux.jl を用いた[5]。

実験するにあたり、まずパターンを記述する DSL (Domain-Specific Language) を Julia 言語上で設計・実装した。提案手法ではパラメタの役割が決まっているため、パターンが与えられれば理想的なパラメタの値を容易に決めることができる。したがって DSL によるパターン集合の記述から 2 層ベイジアンネットを機械的に生成することができる。

```
matchResults(:[
    [X, 1, 2]=>1
    [X, X, 2]=>1
    [X, Y, 2]=>1
], [1,2,2])
```

図 5: Julia 言語上で実装されたデモプログラムの使用例。

図 5 は DSL を用いたデモプログラムの使用例である。関数 matchResults は 2 つの引数を取る。1 つはパターン集合の DSL による定義でもう 1 つはそれにマッチさせる値である。記号 => の右側はパターンの価値である。この関数は、上の層のノードのすべての値の組み合わせに対して、入力との同時確率を計算し、それをソートした結果を出力する。(最も同時確率の高いパターンが入力に対する事後確率を最大にするパターン、すなわち選択されるべきパターンである。) 次の節では提案手法を用いたパターンマッチの例を紹介する。すべての例において、変数ノードは X, Y の 2 つで、活性率は 0.1 、各ノードが取り得る値の集合は $\{0,1,2,3\}$ としている。

4.2 パターンマッチの実行例

図 5 の実行結果のうち、同時確率が 0 でない値の組み合わせは下記の 1 つのみである。(ノード P の値は実際は整数だが対応するパターンを文字列で表示している。JP は同時確率の値である。)

```
P="[X, Y, 2] => 1", X=1, Y=2 : JP=0.000037
```

値 $[1,2,2]$ にマッチするパターンは $[X,Y,2]$ のみなので正しい結果になっている。また、同じネットワークに $[1,1,2]$ を入力した場合は、同時確率が 0 でない値の組み合わせは以下のもののみとなる。

```
P="[X, 1, 2] => 1", X=1, Y=0 : JP=0.001000
P="[X, X, 2] => 1", X=1, Y=0 : JP=0.001000
P="[X, 1, 2] => 1", X=1, Y=1 : JP=0.000037
P="[X, 1, 2] => 1", X=1, Y=2 : JP=0.000037
P="[X, 1, 2] => 1", X=1, Y=3 : JP=0.000037
P="[X, X, 2] => 1", X=1, Y=1 : JP=0.000037
P="[X, X, 2] => 1", X=1, Y=2 : JP=0.000037
P="[X, X, 2] => 1", X=1, Y=3 : JP=0.000037
P="[X, Y, 2] => 1", X=1, Y=1 : JP=0.000037
```

この場合、3 つのパターンのすべてが入力にマッチするが、最も特殊なパターンである $[X,1,2]$ と $[X,X,2]$ がともに最も同時確率が高くなっている。

[4] 値域をより広く $r_p \in (-\infty,\infty)$ としたい場合は $P(V=1|P=p)=sigmoid(r_p)$ のように単調増加関数を用いて区間 $[0,1]$ に変換すればよいだろう。

[5]。Flux.jl は今回の実験のように学習モデルをスクラッチから構築するツールとして適している。目的関数を普通の Julia 言語で記述するだけでパラメタを最適化することができる。ただし、現在のバージョンでは目的関数の記述に使える言語機能に多少制限がある。

下記の例はそれぞれのパターンが異なる価値を持つ例である。

```
matchResults(:[
    [X, 1, 2]=>0.2
    [X, X, 2]=>0.1
    [X, Y, 2]=>1
], [1,1,2])
```

この場合、下記の様に最も特殊なパターンのうちもっとも価値の高いものが、最も同時確率が高い組み合わせとして選ばれる結果になる。

```
P="[X, 1, 2] => 0.2", X=1, Y=0 : JP=0.000200
P="[X, X, 2] => 0.1", X=1, Y=0 : JP=0.000100
P="[X, Y, 2] => 1", X=1, Y=1 : JP=0.000037
...
```

なお、提案手法では価値の差が大きいときに期待と異なる動作をすることがある。下記がその例である。

```
matchResults(:[
    [X, 1, 2]=>0.02
    [X, X, 2]=>0.01
    [X, Y, 2]=>1
], [1,1,2])
```

これを実行した結果は以下のようになる。

```
P="[X, Y, 2] => 1", X=1, Y=1 : JP=0.000037
P="[X, 1, 2] => 0.02", X=1, Y=0 : JP=0.000020
P="[X, X, 2] => 0.01", X=1, Y=0 : JP=0.000010
...
```

ここでは「最も特殊なパターン」ではない $[X, Y, 2]$ が選ばれている。ただしこのような逆転現象は、3.3 節で述べた変数ノードの活性率 α_d を小さくすることで起きにくくすることができるため、実際上の問題は小さいと思われる。

提案手法のもう 1 つの直感に反する振る舞いは、入力ノードの値 0 の扱いである。例えば下記の例は 0 を含むパターンと入力を与えている。

```
matchResults(:[
    [0, 1, 2]=>1
    [X, 1, 2]=>1
    [2, 1, 2]=>1
], [0,1,2])
```

これの実行結果は下記のようになってしまう。

```
P="[0, 1, 2] => 1", X=0, Y=0 : JP=0.027000
P="[X, 1, 2] => 1", X=0, Y=0 : JP=0.027000
...
```

つまり、この場合は最も特殊なパターンに思える $[0, 1, 2]$ が選択されるとは限らず、 $[X, 1, 2]$ も同程度に選択される可能性がある。パターン集合をプログラムと見なす場合、 “if V=0 then ... else ...” に相当する条件分岐がこの方法では書けないことを意味する。しかし、そのような条件分岐は別のやり方で実現することが可能である。DSL においてパターンの要素に “+” を書けば 0 以外の任意の値にマッチする。要素 “+” は、入力ノードに対し 0 以外の値を均等な確率で生成するような重みに変換される。

```
matchResults(:[
    [+, 1, 2]=>1
    [0, 1, 2]=>1
    [2, 1, 2]=>1
], [0,1,2])
```

これを実行すると 0 でない同時確率を持つパターンは $[0, 1, 2]$ のみとなり、0 と 0 以外との値で条件分岐することができるようになる。

DSL では、パターンの要素としてワイルドカード “_” も書けるようになっている。ワイルドカードは 0 も含む任意の値とマッチする。変換されたネットワークにおいては、対応する入力ノードのすべての値を等確率で生成するような重みが設定される。下記はワイルドカードの使用例である。

```
matchResults(:[
    [_, 1, 2]=>1
    [X, X, 2]=>1
    [_, _, 2]=>1
], [1,1,2])
```

これを実行した結果の上位 3 つは以下のようになり、3 つのいずれのパターンも入力にマッチしている（同時確率が 0 でない）ことがわかる。

```
P="[_, 1, 2] => 1", X=0, Y=0 : JP=0.006750
P="[_, _, 2] => 1", X=0, Y=0 : JP=0.001687
P="[X, X, 2] => 1", X=1, Y=0 : JP=0.001000
```

ただし、ここで $[\_, \_, 2]$ の方が $[X, X, 2]$ より同時確率が高いことが、直感に反する動作になっている。パターン $[\_, \_, 2]$ は直感的には $[X, Y, 2]$ と等価に思えるが、そうであるならば $[X, X, 2]$ の方が変数の数が少ないのでより特殊なはずである。この振る舞いが本手法を用いるプログラム合成システムに何か悪影響を与えるかどうかは現時点では明らかではない。2.1 節で述べたように強化学習で各行動ルールの価値が決まる点がプログラム合成システムとしては本質的に重要であり、パターンマッチの細かい振る舞いが人間の直感に合っているかどうかはおそらく重要ではないと現時点では考えている。

```
(1,1,1,1,1)
(1,2,1,2,2)
(1,3,1,3,3)*
(1,0,1,0,0)
(2,1,1,2,1)
(2,2,2,2,2)*
(2,3,3,2,3)
(2,0,0,2,3)
(3,1,1,1,3)*
(3,2,2,2,3)
(3,3,3,3,3)
(3,0,0,0,3)*
```

図 6: 学習に用いた訓練データ。1 つのデータが持つ 5 つの要素は順に 1 つのパターンノード、1 つの変数ノード、3 つの入力ノードの値を表す。価値ノードの値は常に 1 なので省略している。1 2 個のデータのうち * をつけた 4 個を除いて学習しても正しい学習結果になる。

4.3 パラメタの学習

本節ではネットワークのパラメタが完全データで学習できることを示す。「完全データによる学習」とはベイジアンネットのパラメタ学習において、隠れ変数がなくすべての変数の観測値が与えられるような学習である。

訓練データは以下の方法で生成される人工データである。まず 4.1 節で述べた DSL を用いて小規模のパターンマッチのネットワークを構築する。そして、そのネットワークの上の層のすべての値の組に対する下の層の値を生成する。このようにして作った全ノードの値のリストを訓練データとする。ただし、パターンノードの値が 0 となるものは訓練データから除いておく。

学習する側のネットワークは、訓練データの生成に用いたネットワーク構造と全く同じものを用いる。パラメタの初期値はランダムな値とする。

モデルのパラメタ w_{ciud}, w_{ujdk} は区間 $[0,1]$ の値でなければならないが、勾配法で学習するとこの区間をはみ出すことがありやっかいである。そこで区間 $(-\infty,\infty)$ のパラメタ b_{ciud}, b_{ujdk} を導入しこれを学習の対象とする。値の変換はシグモイド関数 $w = 1/(1+e^{-b})$ で行う。

学習の目的関数は訓練データの負の対数同時確率の和であり、これを小さくする方向にパラメタを学習する。これはパラメタの最尤推定を行うことに相当する。

下記の 3 つのパターンを持つネットワークから生成した訓練データを図 6 に示す。

```
[1, X, X]=>1
[X, 2, X]=>1
[X, X, 3]=>1
```

変数ノードの数を 1 個、変数が取り得る値を $\{0,1,2,3\}$ として、Momentum 法（学習率 0.01 、速度減衰率 0.99）を用い 1000 エポック学習させたところ、安定して正しい入力ノードの確率分布を生成するネットワークに収束することが確認された。

さらに、図 6 の訓練データのうち * 印をつけたものを間引いて学習してもやはり安定して正しいネットワークに収束することが確認された。このことは、$[1,1,1]$ と $[1,2,2]$ という 2 つの経験のみからパターン $[1,X,X]$ で表される一般化された知識を獲得し、未経験の入力 $[1,3,3]$ にも対処可能になることを表している。この性質は、経験から汎用性の高い知識を自律的に獲得する知的エージェントの実現に役立つと考えられる。

4.4 隠れ変数がある場合の学習

知的エージェントが経験のみから知識を獲得する場合は、上の層の変数の値は与えられない。その状況を想定した学習についても予備的実験を行った。今回は、目的関数は隠れ変数を周辺化した対数尤度とし、勾配法で学習した[6]。しかし学習結果はあまり正しい分布を生成するネットワークとはならなかった。よりよい学習結果を得るための工夫として、勾配法ではなく EM アルゴリズムを用い、かつ E ステップにおいて隠れ変数の事後分布に関する正則化項を入れる方法が考えられる。同様の手法は変分オートエンコーダーや筆者らの以前の研究 [6] でも使われている。その実装は将来の課題とする。

5 関連研究

Tensor Product Representations (TPR)[1] は記号処理に必要な変数束縛構造をニューラルネットワークで表現する手法である。TPR の最新の応用としては、自然言語で書かれた問題を Lisp プログラムなどの形式言語に変換するアーキテクチャとして TP-N2F [14] が提案されている。TPR はテンソルを用いて変数束縛の構造を表現する点が、本稿の提案手法と共通している。我々の手法は TPR よりもかなり制限が強く、パラメタは 3 階テンソル固定で、扱うデータ構造もフラットな数値のベクトルに制限されている。しかし我々の手法はベイジアンネットとして定式化されている点が TPR

[6] 周辺化の計算コストは変数の数に対して指数関数的に増大するが今回のような小規模ネットワークでは問題とならない。将来的には大規模化可能な確率伝搬法を開発し、それを用いて隠れ変数の値を近似推論する計画である。

とは異なり、推論・学習のアルゴリズムの選択肢の多さや他のモジュールとの結合が容易である点で有利になり得ると考えている。

RRL(Relational Reinforcement Learning)[3] は、強化学習と帰納論理プログラミングを組み合わせたシステムであり、タスクを試行錯誤によって繰り返し解きながら、行動価値関数を圧縮表現する論理決定木と呼ぶ構造を学習する。行動価値関数を記号表現を使って圧縮することで極めて強力な汎化能力を得ようとする点で、我々の研究と目標は同じである。我々が用いようとしているパターンマッチとサブルーチンを組み合わせた表現方法は、論理決定木よりは表現力が強いものになる。

強化学習を用いたプログラム合成による汎用人工知能の理論には AIXI[2] や UCAI[8]、動作するシステムとしては DNC (Differentiable Neural Computers)[4] がある。提案手法はベイジアンネットを用いることで視覚野や運動野などを表現する確率的モデルとも結合が容易である点に特徴がある。

6 まとめと今後

2 層ベイジアンネットを用いて整数値ベクトル対するパターン獲得・パターンマッチを行う手法を提案した。本手法は親子ノード間の結合を制御できるように設計された条件付確率モデルを用いる。勾配法による学習により、訓練データから一般化された知識が獲得される。この性質は、経験から汎用性の高い知識を自律的に獲得する知的エージェントの実現に役立つと考えられる。

今後の課題の 1 つにネットワークの大規模化がある。そのためには隠れ変数の値を効率的に推論できるような大規模化可能な確率伝搬アルゴリズムを開発する必要がある。

汎用人工知能実現に向けたプログラム合成システムという観点からは、2 章で述べたアーキテクチャにさらにいくつかの機能追加が必要である。動的記憶域のように動作する連想記憶機構、自然言語と知識の間の相互変換、高階述語を用いたメタプログラミングなどの実現方法についての考察を現在進めている。

同じ目的を持つ研究者を増やすことがもっとも優先度の高い課題である。そのために、2 章で構想を述べた知的エージェントのアーキテクチャの最小構成モデルを極力単純化した世界の中で動かし、脳にヒントを得た汎用人工知能の実現が十分に現実的であることを示したい。

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] Paul Smolensky, Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems, Artificial Intelligence, Vol 46, pp.159–216, 1990.

[2] Hutter, M., A theory of universal artificial intelligence based on algorithmic complexity, Technical Report: cs.AI/0004001, 2000.
http://arxiv.org/abs/cs.AI/0004001

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

[4] A. Graves, G. Wayne et al., Hybrid computing using a neural network with dynamic external memory, Nature 538, pp.471-476, 2016.

[5] 一杉裕志, 疑似ベイジアンネットを用いた認知モデルのプロトタイピング手法の提案, 第 4 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2016.

[6] Yuuji Ichisugi and Takashi Sano, Regularization Methods for the Restricted Bayesian Network BESOM, In Proc. of International Conference on Neural Information Processing (ICONIP2016) Part I, LNCS 9947, pp.290–299, 2016.

[7] Naoto Takahashi and Yuuji Ichisugi, Restricted Quasi Bayesian Networks as a Prototyping Tool for Computational Models of Individual Cortical Areas, In Proc. of Machine Learning Research (AMBN 2017), Proceedings of Machine Learning Research, Vol.73, pp.188–199, 2017.

[8] Susumu Katayama, Computable Variants of AIXI which are More Powerful than AIXItl, 2018.
https://arxiv.org/abs/1805.08592

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 一杉裕志, 中田秀基, 高橋直人, 佐野崇, 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, 第 14 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2020.

[14] Kezhen Chen, Qiuyuan Huang, Hamid Palangi, Paul Smolensky, Kenneth D. Forbus, Jianfeng Gao, Mapping Natural-language Problems to Formal-language Solutions Using Structured Neural Representations, In Proc. of ICML 2020, 2020.