プログラム合成対象言語 Pro5Lang のための行動価値関数圧縮アルゴリズム

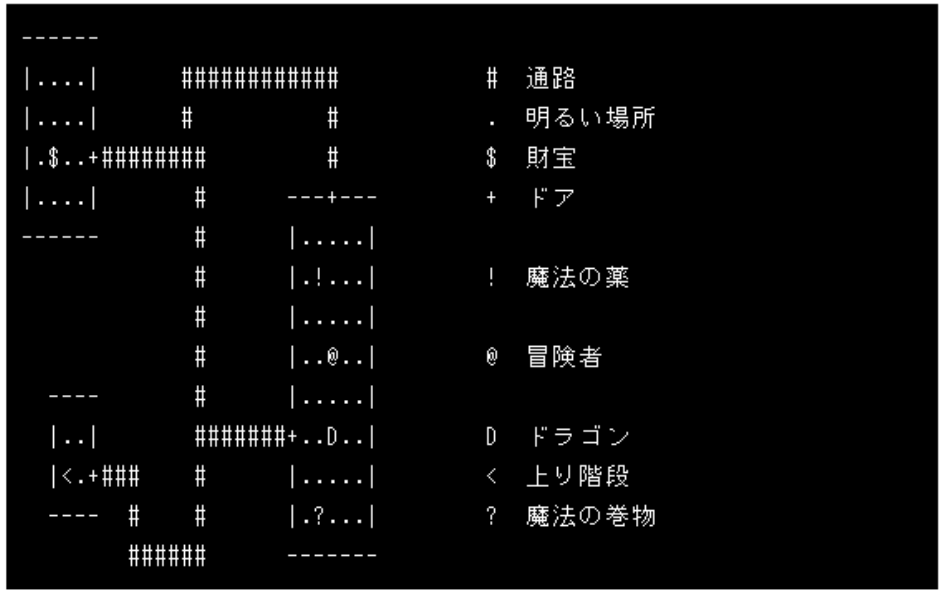
第22回 人工知能学会 汎用人工知能研究会(SIG-AGI)

一杉裕志, 中田秀基, 高橋直人, 竹内泉（産総研）, 佐野崇（東洋大）

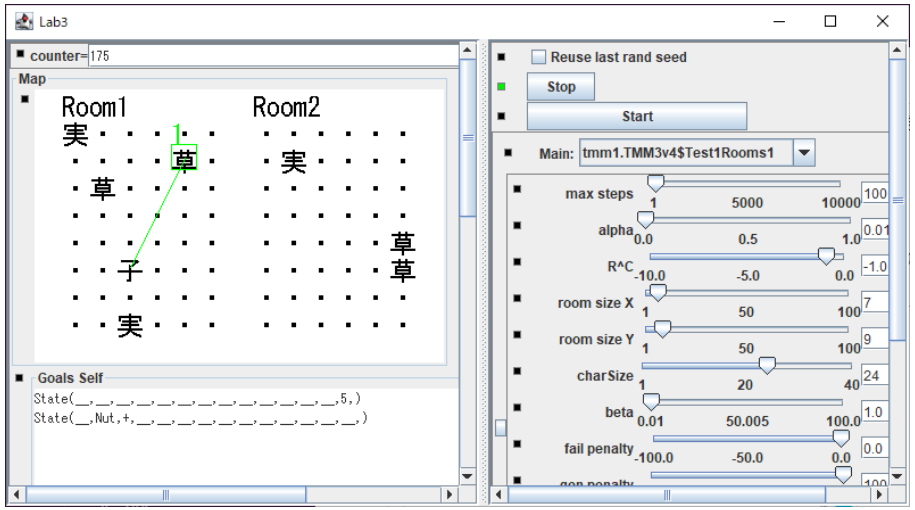
2022-11-22

私の研究の中期的目標

- ローグライクゲームのような**実世界を極力単純化した環境で**脳型AGIのデモを動かす
- 中核技術
  - プログラム合成対象言語 Pro5Lang
    [一杉+ 第20回 汎用人工知能研究会, 2022]
  - 再帰的強化学習 RGoal
    [Ichisugi+ 2019]
  - 大脳皮質モデル BESOM
    [Ichisugi 2007]



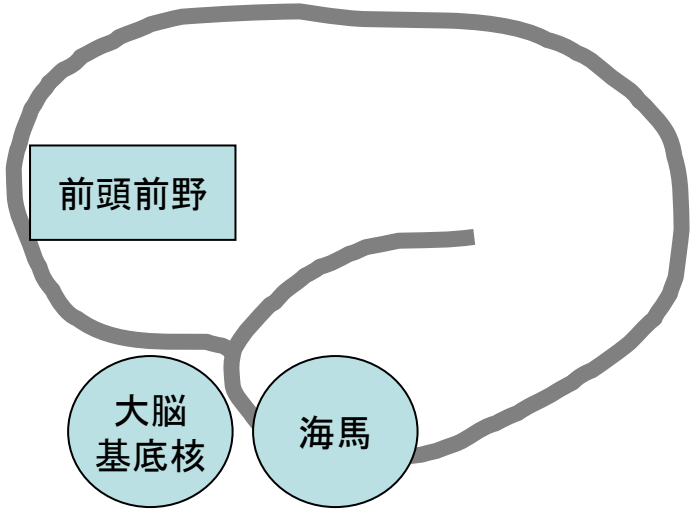
「ローグライクゲーム - Wikipedia」
https://ja.wikipedia.org/wiki/ローグライクゲーム



開発中のAGIエージェント実行環境

プログラム合成対象言語 Pro5Lang [一杉+ 第20回 汎用人工知能研究会，2022]

- 証明済み命題を確率的に処理するプログラミング言語：**論理型言語＋機械語**
  (Probabilistic Proven-Proposition Processing Programming Language)
- AGI エージェントの思考・行動を制御するプログラムを表現するためのプログラミング言語
- このプログラムを、AGI エージェントが**自分自身の経験から自律的に**獲得するアーキテクチャの実現を目指してる



プログラム獲得のためのさまざまな機構

- 自動応答機構：
  - 刺激に対して自動的に応答（目立つ物体に視線を向ける、など）
  - 自動応答だけが試行錯誤を生み出す原因となる。
- 経験評価・保持機構：
  - 行動結果の価値を計算し、結果を経験履歴として一定期間保持

今回の発表

- 帰納推論機構：
  - 経験履歴に基づいて、汎用性の高い行動ルールを獲得
- 演繹推論機構：
  - 汎用的な行動ルールや宣言的知識を組み合わせる
  - 模倣や対話を通じた知識獲得

Pro5Lang のプログラムのいろいろな表現のレベル

```
rule(w(), g1, call(g2));
rule(w(a(ThatO1, c(Now, Here, z, Knows, 0, 0),
        c(Now, Wh, Scissors, Exists, 0, 0))),
        g1,
        set(a(ThatO2, c(Now, Here, Alice, WantsToSayTo, z, 0),
    c(Now, Wh, Scissors, Exists, 0, 0))));
rule(w(a(ThatO2, c(Now, Here, Alice, WantsToSayTo, z, 0),
        c(Now, Wh, Scissors, Exists, 0, 0))), g1, SayIt);
rule(w(a(ThatO2, c(Now, Here, Alice, SaidTo, z, 0),
        c(Now, Wh, Scissors, Exists, 0, 0))), g1, Listen);
rule(w(a(ThatO2, c(Now, Here, __, SaidTo, Alice, 0),
        c(Now, x, Scissors, Exists, 0, 0))),
        g1,
        set(a(Now, x, Scissors, Exists, 0, 0)));
```

プロトタイピングのために
手で書いたプログラム

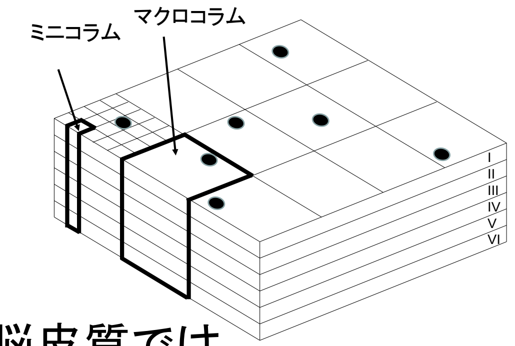
```
2:rule((__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,
__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
(Simple,Now,+,Scissors,Exists,0,0,0,0,0,0,0,0,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
Call(ThatO1,Now,Here,+,Knows,0,0,Now,Wh,Scissors,Exists,0,0,__,__,__,__,__,__,__,__,__
,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,)).q = 0.0
3:rule((ThatO1,Now,Here,z,Knows,0,0,Now,Wh,Scissors,Exists,0,0,__,__,__,__,__,__,__,__
,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,),
(Simple,Now,+,Scissors,Exists,0,0,0,0,0,0,0,0,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
Set(ThatO2,Now,Here,Alice,WantsToSayTo,z,0,Now,Wh,Scissors,Exists,0,0,=,=,=,=,=,=,=,=,
=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,)).q = 0.0
4:rule((ThatO2,Now,Here,Alice,WantsToSayTo,z,0,Now,Wh,Scissors,Exists,0,0,__,__,__,__,
__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__
,__,__,__,__,__,__,),
(Simple,Now,+,Scissors,Exists,0,0,0,0,0,0,0,0,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
SayIt).q = 0.0
5:rule((ThatO2,Now,Here,Alice,SaidTo,z,0,Now,Wh,Scissors,Exists,0,0,__,__,__,__,__,__,
__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__
,__,__,__,__,),
(Simple,Now,+,Scissors,Exists,0,0,0,0,0,0,0,0,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
Listen).q = 0.0
6:rule((ThatO2,Now,Here,__,SaidTo,Alice,0,Now,x,Scissors,Exists,0,0,__,__,__,__,__,__,
__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__
,__,__,__,__,),
(Simple,Now,+,Scissors,Exists,0,0,0,0,0,0,0,0,__,__,__,__,__,__,__,__,__,__,__,__,__,_
_,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,__,),
Set(Simple,Now,x,Scissors,Exists,0,0,0,0,0,0,0,0,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=
,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,=,)).q = 0.0
```

記号の配列の集合にエンコード

今回扱うのはこのレベルの表現

(0, 0, 29, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0, 0, 0, …)
(0, 0, 0, 0, 84, 27, 0, 0, 0, 0, 0, 0, 0, 0, 31, 0, …)
(0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, …)
(0, 87, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 78, …)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0, 0, 0, 0, …)
(0, 50, 0, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …)
…

将来はスパースな整数ベクトル表現
（NNやベイジアンネットで処理）



大脳皮質では
活動するミニコラムの集合で
符号化されると想定

経験履歴の圧縮で何ができる？

エージェントの経験履歴：
「リンゴをアリスに手渡した結果、今アリスはリンゴを持っている」
「みかんをボブに手渡した結果、今ボブはみかんを持っている」
・・・

経験履歴を圧縮して得られる行動ルール：
「Y に X を持たせるためには、 X を Y に手渡せばよい」

（行動ルールは汎用性が高いのでいろいろな行動に応用できる）

この行動ルールを応用した行動：
チャーリーに鍵を持たせるためには、チャーリーに鍵を手渡せばよい

行動価値関数のテーブル圧縮の例

[一杉+ 第10回 汎用人工知能研究会, 2018]

X \ Y	0	1	2	3	4
0	2.0	1.0	1.0	3.0	1.0
1	1.0	2.0	1.0	3.0	1.0
2	1.0	1.0	2.0	3.0	1.0
3	1.0	1.0	1.0	4.0	1.0
4	1.0	1.0	1.0	3.0	2.0

パターン	値
(3,3)	4.0
(X,3)	3.0
(X,X)	2.0
(X,Y)	1.0

サイズ 5 x 5 = 25 のテーブルが4個のルールに圧縮

Pro5Lang のプログラム
＝ 再帰的強化学習 RGoal の行動価値関数 Q(s,g,a) を圧縮表現した行動ルール集合

特殊な２層ベイジアンネット(BESOM)を用いた圧縮

[一杉+ 第15回 汎用人工知能研究会，2020]

条件付確率モデル

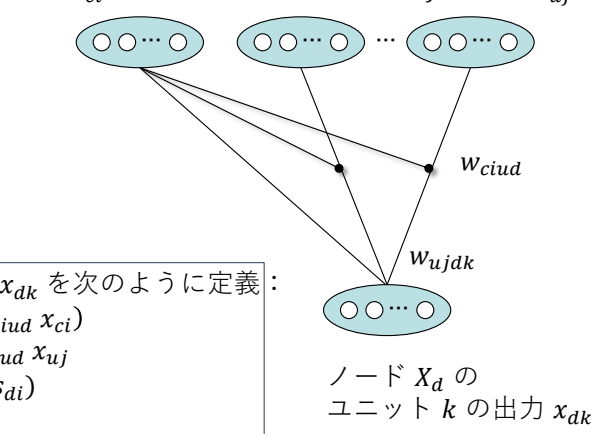
$P(X_d = d_k | X_u = u_j, \dots) = x_{dk}$

・ U, D はそれぞれ上の層・下の層のノードのインデックスの集合で $c, u \in U, d \in D$

・ x_{ci}, x_{uj}, x_{dk} はそれぞれ変数 X_c, X_u, X_d の値のone hot vector表現

・ w_{ciud} は、X_c のユニット i が X_u と X_d の間の結合を制御する重み

・ w_{ujdk} は、X_u のユニット j と、X_d のユニット k の間の重み



$u \in U, d \in D, k \neq 0$ に対して g_{ud}, s_{dk}, x_{dk} を次のように定義：

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

x_{d0} を次のように定義：

$$x_{d0} = 1 - \Sigma_{k \neq 0} x_{dk}$$

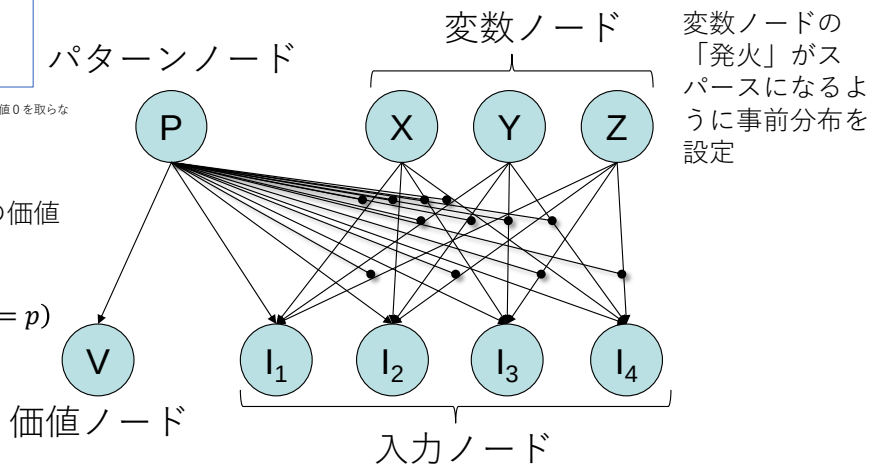
1

パターンマッチを行うベイジアンネット

[X, 1, 2]=>0.1
[X, X, 2]=>0.2
[1, X, 1]=>1
…

注：パターンノードは値０を取らない

パターン p の価値を r_p とすると
$P(V = 1 | P = p) = r_p$



変数ノードの「発火」がスパースになるように事前分布を設定

接続されている変数ノードと入力ノードは同じ値になるように重みを設定

1

勾配法を用いた教師あり学習で動作を確認

本研究

- 大脳皮質モデル BESOM は未完成で、現時点では大規模化できない（解決の方針は立っているが時間がかかりそう）
- そこでアドホックだが軽い圧縮アルゴリズムを提案
  - k-means 法に似た反復アルゴリズム
- なお、今回はワイルドカードは扱うが、変数は扱わない

k-means 法

- クラスタリングアルゴリズムの１つ
  - EM
  - 自己組織化マップ（大脳皮質の計算モデルの１つでもある）と類似
    - 違い：オンラインかバッチか、近傍学習があるかないか
- 下記の値を最適化

$$\arg\min_{V_1,\ldots,V_k} \sum_{i=1}^{n} \min_j \|x_i - V_j\|^2$$

  - データ x_i と最も近いクラスタの中心 V_j との距離の二乗の総和を最小化するようなクラスタ中心 $V_1,\ldots,V_k$ を求めるのが目的

参考：https://ja.wikipedia.org/wiki/K平均法

- 求まるのは局所解
  - 初期値を工夫すると性能が大きく改善： K-means++ 法

圧縮アルゴリズム

- k-means 法

参考：https://ja.wikipedia.org/wiki/K平均法

  - k 個のクラスタに各データをランダムに割り当てる。
  - 収束するまで下記を繰り返す。
    - 割り当てたデータをもとに各クラスタの中心を計算する。
    - 各データを最も近い中心を持つクラスタに割り当て直す。

- **提案アルゴリズム**

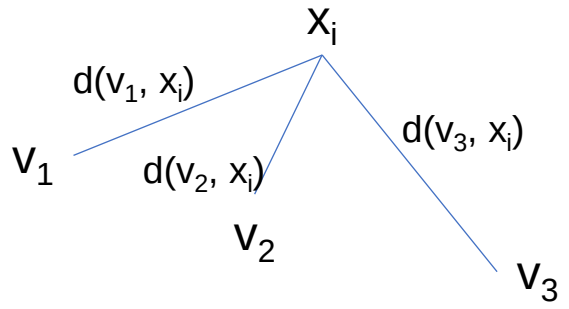
具体的にどう設計するかが問題

  - k 個の参照ベクトルを**初期化**する。
  - 収束するまで下記を繰り返す。
    - すべてのデータを、**最も近い**参照ベクトルに割り当てる。
    - 各参照ベクトルに割り当てられたデータの集合を１つの参照ベクトルに**圧縮**する。

今回実装した提案アルゴリズムのバリエーション

- **初期化方法**（２通り）
  - I1: データの中から重複しないようにランダムに１つずつ選択
  - I2: １つ目をデータからランダムに選んだあと、すでに選んだ参照ベクトルから最も遠いデータを順に選択 cf. k-means++法
- **「距離」の大小比較**（２通り）
  - D1: 参照ベクトルにマッチしない要素数、一般度、価値の二乗誤差の順に比較して判定
  - D2: 価値の二乗誤差、参照ベクトルにマッチしない要素数、一般度の順に比較して判定
- **圧縮方法**（１通り）
  - 割り当てたすべてのデータに共通な要素はそのまま、１つでも違うものがある要素はワイルドカードに　例：{(1,2),(1,2),(9,2)} -> (*,2)

距離の大小比較



- 参照ベクトル v とデータ x との間の距離
  d(v, x) =（マッチしない要素数, 一般度, 価値の二乗誤差）
  一般度：ワイルドカードを使ってマッチした要素数

例： ベクトル 価値
d((*, *):3 , (1, 2):2) = (0, 2, 1)
d((1, 2):3 , (1, 2):3) = (0, 0, 0)
d((0, 2):3 , (1, 2):3) = (1, 0, 0)

D1: マッチしない要素数、一般度、価値の二乗誤差の順に比較

(0, 0, 0) < (0, 2, 1) < (1, 0, 0)

D2: 価値の二乗誤差、マッチしない要素数、一般度の順に比較

(0, 0, 0) < (1, 0, 0) < (0, 2, 1)

人工データを用いて動作確認

手で与えた生成ルールから人工データを生成し、それを提案アルゴリズムで圧縮
生成ルール数 8, 参照ベクトル数 k = 16, 訓練データ数 k x 5 = 80

パターン　　価値

(100, *, *, *, *, *, *) :−3
(100, 1, *, *, *, *, *) :−2
(100, 1, 2, *, *, *, *) :−1
(100, 1, 2, 3, *, *, *) :0
(101, *, *, *, *, *, *) :−3
(101, *, *, *, 4, *, *) :−2
(101, *, *, *, 4, 5, *) :−1
(101, *, *, *, 4, 5, 6) :0

訓練データ生成

(101, 21, 58, 8, 4, 22, 68) :−2.0
(101, 23, 91, 70, 78, 20, 5) :−3.0
(101, 2, 32, 71, 4, 5, 57) :−1.0
(101, 80, 68, 71, 4, 5, 6) :0.0
(100, 1, 77, 40, 92, 21, 26) :−2.0
(100, 1, 31, 50, 8, 79, 50) :−2.0
(101, 92, 68, 77, 29, 48, 17) :−3.0
(101, 91, 2, 69, 4, 30, 73) :−2.0
(100, 24, 78, 31, 2, 56, 12) :−3.0
(100, 1, 2, 3, 49, 87, 89) :0.0
(100, 1, 2, 3, 30, 67, 96) :0.0
(100, 43, 75, 86, 96, 22, 49) :−3.0
(101, 60, 28, 93, 4, 5, 85) :−1.0
(100, 1, 2, 60, 57, 94, 0) :−1.0
(101, 24, 20, 39, 52, 36, 84) :−3.0
(100, 8, 52, 58, 17, 52, 95) :−3.0
(101, 51, 8, 79, 4, 69, 40) :−2.0
…

エージェントの
経験履歴を模した人工データ

圧縮

(100, *, *, *, *, *, *) :−3.0
(100, 1, *, *, *, *, *) :−2.0
(100, 1, 2, *, *, *, *) :−1.0
(100, 1, 2, 3, *, *, *) :0.0
(100, 1, 2, 3, 42, 97, 84) :0.0
(100, 8, 52, 58, 17, 52, 95) :−3.0
(101, *, *, *, *, *, *) :−3.0
(101, *, *, *, *, *, 29) :−3.0
(101, *, *, *, 4, *, *) :−2.0
(101, *, *, *, 4, 5, *) :−1.0
(101, *, *, *, 4, 5, 6) :0.0
(101, *, *, *, 29, *, *) :−3.0
(101, 0, 76, 91, 73, 99, 83) :−3.0
(101, 4, 12, 27, 98, 80, 82) :−3.0
(101, 24, 20, 39, 52, 36, 84) :−3.0
(101, 54, 30, 10, 4, 60, 37) :−2.0

もとの生成ルールと意味的に
同じものが得られれば圧縮成功

圧縮対象の訓練データの性質

- 要素の値は０～９９の整数値
- 生成ルールはほとんどがワイルドカード
- 価値はほぼ整数値
- 特殊な値になるほど価値が高くなる

性能の定量的評価

- テストデータに対して、圧縮後のテーブルを用いて得られる価値と、真の価値との二乗誤差で評価
  （理想的な圧縮なら誤差０）

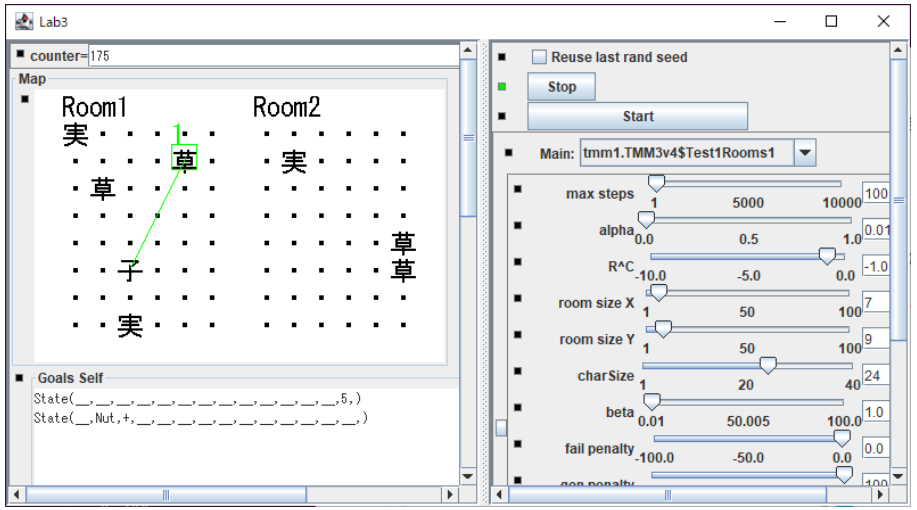
- 今回の実験においては、 I2 + D2 の組み合わせでほぼ理想的な圧縮結果
- 訓練誤差は、 I2 + D2 は初期化後１ステップでほぼ収束、他も２ステップでほぼ収束

表 1: 実験結果

初期化・距離	訓練誤差	汎化誤差
I1 + D1	0.12	0.15
I2 + D1	0.39	0.51
I1 + D2	0.20	0.24
I2 + D2	0.01	**0.01**

まとめと今後

- k-means 法に似た方法で行動価値関数圧縮がうまくいく見込みを得た
- 今後
  - 変数の導入（今回はワイルドカードのみを用いて圧縮）
    - {(1, 1, 1), (1, 2, 2), (1, 3, 3)} → (1, X, X)
  - 実データ（エージェントの経験リプレイバッファ）への適用
  - ローグライクな世界でエージェントが
    自律的に知識獲得するデモの実現を目指す



以上