人工知能学会研究会資料
SIG-AGI-014-02

# 予測モデルと評価モデル（PEM: Prediction and Evaluation Models）に基づく汎用人工知能の実現方式

## Proposal of Artificial General Intelligence based on Prediction and Evaluation Models (PEM)

小池 賢一 [1]

KOIKE Kenichi 1[1]

**Abstract:** We proposed a method to realize Artificial General Intelligence in Ref. [1], and showed that an artificial intelligence that simulates the motion of an object from a video can be automatically generated. Then, useful information was obtained. However, it was necessary to repeat 13 simulations to obtain good result, and the success rate of the simulation was 42%. In this paper, we apply Monte Carlo tree search used in games such as shogi when evaluating predicted alternatives, and show that it is also effective in Artificial General Intelligence.

# 1 はじめに

## 1.1 背景と本研究の内容

文献[1]において予測モデルと評価モデルに基づいて汎用人工知能を実現する方式を提案した．本稿ではこの方式をペム（PEM : Prediction and Evaluation Models）方式と呼ぶことにする．文献[1]ではさらにこの方式の実装例として動画からオブジェクトの動きをシミュレートする方法を示し人工知能を自動生成できることを示した．そして，与えた動画より評価値の高いオブジェクトの動きを求めることで有用な情報を得られることを示した．ただし，その結果を得るまでに 13 回のシミュレーションを行う必要があり，それ以外のシミュレーションではオブジェクトが逆走してしまうなどの失敗例も多く発生するため，成功率は 42%であった．そこで，本研究では選択枝を評価するときに将棋などのゲームで利用されているモンテカルロ木探索を適用して，成功率を改善することを示し，モンテカルロ木探索が汎用人工知能においても有効であることを示す．

## 1.2 シミュレーションが必要な問題

深層学習により解かれる問題を大きく分類すると，深層学習のみで解ける問題と深層学習とシミュレーションを組み合わせなければ解けない問題に分けることができると考えられる．深層学習のみで解ける問題の例としては以下がある．

- ✓ パターン認識
- ✓ 音声認識

一方，深層学習とシミュレーションを組み合わせないと解けない問題の例としては以下がある．

- ✓ 将棋の指し手の決定
- ✓ 車の運転

これら二種類の問題を比較すると，将棋を指す人工知能は車の運転などの他の問題には全く流用できないことから，シミュレーションを必要とする問題を解決する汎用人工知能を実現する方がより困難と考えられる．そこで，文献[1]ではシミュレータを必要とする問題を解決する汎用人工知能を提案した．

## 1.3 基本アイデア

深層学習とシミュレータを組み合わせる場合，シミュレータが対象の問題専用であるため，結果として作成される人工知能も専用人工知能になる．そのため汎用人工知能を実現するためには専用シミュレータを自動生成しなければならないということになる．そこで，文献[1]では人が目で見た情報のみからシミュレートを行うのと同様に，動画のみから自動的にシミュレータを生成する方式を提案した．

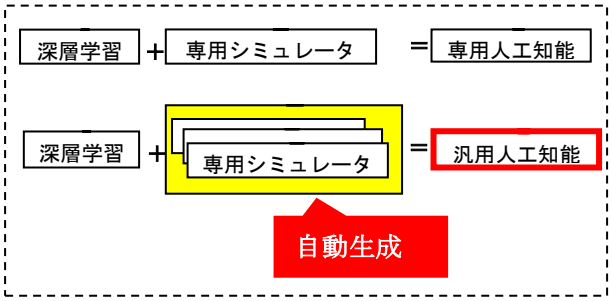


図 1 基本アイデア

### 1.4 人によるシミュレーションにおいて必要とするデータ量と学習時間

人がシミュレーションを行う場合を観察すると，最も手本が少ない場合は数回の手本を見るだけでシミュレーションを開始できる場合がある．このことから汎用人工知能の場合も数回の手本のデータからシミュレータを自動生成できる方式でなければならないと考えられる．また，人は手本見た直後からシミュレーションを開始できることから，汎用人工知能もそのような問題では学習時間は短時間で実現できる方式になると考えられる．

## 2 汎用人工知能の実現方式

最初に将棋の人工知能を生成する方法について検討する．将棋では以下の二つの手順で将棋を指す人工知能が開発されている．

A) 各駒の正しい動きを人がプログラムを作成してシミュレートする
B) 各局面の評価値（勝つ確率）を過去の棋譜から深層学習により作成する

この場合手順A)は人がプログラムを作成しているが，そのかわりに深層学習により予測モデルを過去の棋譜から作成することができる．作成したモデルを使用して図 2 の局面で飛車の移動先を予測すると移動できる場所の確率は数%から数十%など通常の確率になり，一方，移動できない場所の確率は 0.01%など極端に小さい値になると考えられる．つまり，極端に小さい確率の場所を指さないようにすることで人がプログラムを作成しなくても駒の正しい動きをシミュレートできる．

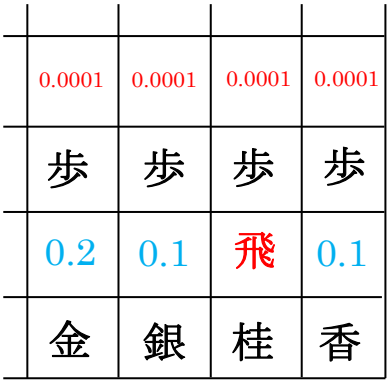


**図 2　将棋の予測モデルによる予測**

以上から将棋の場合は予測モデルと評価モデル（PEM）を作成することで将棋の人工知能を自動生成できることがわかる．

## 3 予測モデルと評価モデルの生成方法

将棋の人工知能を PEM 方式で自動生成すると以下の手順になる．

① 各局面における各駒の動きを予測する予測モデルを過去の棋譜から深層学習により作成する
② 各局面の評価値（勝つ確率）を求める評価モデルを過去の棋譜から深層学習により作成する

文献[1]ではこの手順を他の各種問題にも適用できるように拡張することで汎用人工知能を実現する方法を提案した．つまり，①で対象を「駒」に限定するのではなく，動画の中に存在するもの全てのオブジェクトを対象とし．さらに，オブジェクトの「動き」だけでなくオブジェクトの「形の変化」や「色の変化」などのあらゆる変化を予測モデルで学習する．②については，「盤面」だけでなく画面全体の中のオブジェクトの配置や状態に対する評価値を学習する．また，将棋の場合は途中の経過は評価値に影響しないが，一般の問題では，例えば「人の踊りの動画」などのようにオブジェクトの途中の変化の経過が評価値に影響するため途中の経過も含めて評価値を学習するようにする．以上を踏まえ①②の手順を以下のように拡張した．

(1) 動画に現れる各オブジェクトの正しい変化（動きを含む）をオブジェクト毎に動画から深層学習で学習し，予測用モデルを生成する
(2) 各オブジェクトの配置や状態について最初から最後までの変化と最後の状態に対して人が与えた評価値を深層学習により学習し，評価用モデルを生成する

なお，将棋の場合は大量の棋譜から学習を行う必要があるが，汎用人工知能では大量の場合のみでなく少数の動画からでも学習できる点が異なる．

## 4 シミュレーション実行手順

将棋の場合，作成した予測モデルと評価モデルを使用してシミュレーションを実行する手順は以下になる．

(a) 駒を初期状態か解きたい局面に配置する
(b) 予測モデルを使用して各駒の正しい動きを求める
(c) 評価モデルを使用して次の局面の中で最も評価値の高い局面を求める
(d) 玉が詰みの状態になるまで(b)(c)を繰り返す

手順(a)について，一般の問題では初期状態は複数

存在することを考慮する必要がある，またオブジェクトの位置は将棋のようにマスの中に限定されるとは限らない点も異なる．(b)については，将棋では一度に一つの駒しか動かせないが，一般の問題では複数のオブジェクトが同時に動くことができる. なお，将棋の場合は周りの駒に配置により駒の動ける位置が制限されるが，一般の問題では周り環境にも制限される場合がある．例えば，コースに沿って走る車はコースアウトすることはできない．そこで，ありえないオブジェクトの変化を防止するため制約条件としても動画を利用し，各オブジェクトがとりうる状態は動画に現れる範囲に制限する．例えばオブジェクトの動く範囲やスピードなどは動画に現れる範囲に制限する．(c)の評価については，(b)でのオブジェクトの動かし方の組み合わせで評価値が異なるため色々な組み合わせで評価を行う．(d)については，一般の問題では終了条件は異なるため，終了条件を人が与えるか，評価の高い各動画の最後のオブジェクトの配置や状態を調べてそれを終了条件とする．
以上を踏まえて実行手順を以下のように拡張する．

(A) 各オブジェクトを幾つかある初期状態の一つかあるいは解きたい場面に配置する

(B) 各オブジェクトの正しい変化の候補を予測用モデルで二通り以上生成する，ただし，オブジェクトの動く範囲やその他の属性（速さなど）は動画に現れる範囲に制限する

(C) 各オブジェクトを同時に変化させて全ての組み合わせにより次の場面を生成し，評価用モデルを使用して最も評価値の高い状態を選択する

(D) 各オブジェクトが終了条件の状態になるまで(B)(C)を繰り返し，終了時点の評価値が低く且つ初期状態から始めた場合は(A)に戻り別の初期状態から開始し，一方解きたい場面から開始した場合は終了する．

## 5 予測モデルと評価モデルの生成の実装例

汎用人工知能はオブジェクトの全ての属性の変化をシミュレートする必要がるが, 今回は PEM 方式の有効性を示すために，特に重要な属性であるオブジェクトの動きをシミュレートする例を示す．動画として，図 2 のように白板上に手書きしたコースを棒の先に付けた車の絵を時計周りに動かすところを撮影して使用した．一本の動画は 5 秒程度の長さであり学習用に 10 本の動画を用意した. 今回はより短い時間で周回する動き方を求めるため，コースを一周する時間が短い動画に最も高い評価値を与えた．動きを予測するモデルを作成するため 3 章の手順(1)を以下のように詳細化する．

(1-1) 動画に現れるオブジェクトの位置を求める

(1-2) 直前の数フレームのオブジェクトの位置から次の位置を予測するように学習する．

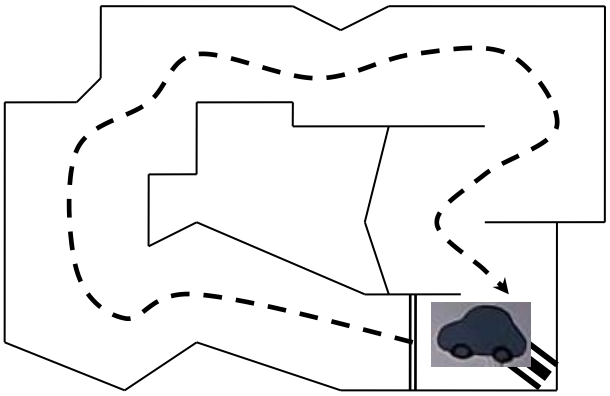

**図 3　動画の内容**

手順(1-1)については，動画の各フレームの車の位置を深層学習や OpenCV などのライブラリを使用して求める．今回は OpenCV を使用し動画の背景を消した後 connectedComponentsWithStats 関数により車の位置を求めた. 図 4 に 10 回の動画から車の絵の経路を求めた結果を示す.

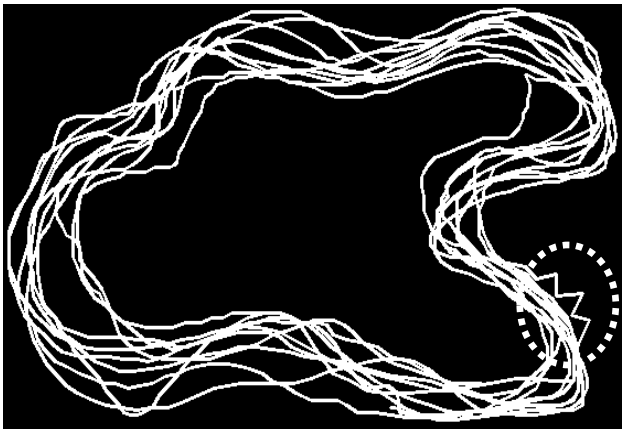

**図 4　車の絵の移動経路**

図 4 の右下の点線で囲った折れ線状の部分は車の絵の位置を求めたときのノイズであるが，多少のノイズは問題ないと考え今回はそのままデータとして使用した．手順(1-2)については，深層学習を使用して直前の 5 フレームの経路から次の車の位置を予測する．入出力データとしては各フレームでの速度と方向を与える．表 1 にデータの内容を示す.

**表 1 予測モデルの入出力データ**

| No | 入出力 | データ種別 | 単位 | データ数 |
|---|---|---|---|---|
| 1 | 入力 | 車の速度 | 正規化値 (0.0~1.0) | 5 |
| 2 | | 車の方向 | ラジアンの | 5 |

| | | | | |
|---|---|---|---|---|
| | | | 正規化値 | |
| 3 | 出力 | 車の速度 | 正規化値 (0.0~1.0) | 1 |
| 4 | | 車の方向 | ラジアンの正規化値 | 1 |

直前の経路が同じでも次の移動先が異なる場合がありうる．そこで，モデルを二つ以上用意して一つ目のモデルで予測できないデータを二つ目以降のモデルで予測する．モデルは以下の手順で作成する．

(1-2-1) 動画から取得した全てのデータでデータ分割用の予測モデルを作成する

(1-2-2) データ分割用モデルを使用して誤差の小さいデータと誤差の大きいデータに分ける

(1-2-3) 誤差の小さいデータで一つ目のモデルを作成し誤差の大きいデータで二つ目のモデルを作成する．誤差の大きいデータが多数残る場合はさらにモデルを増やす

今回は二つのモデルを作成した図 5 に一つ目のモデル用のデータの例を示す．

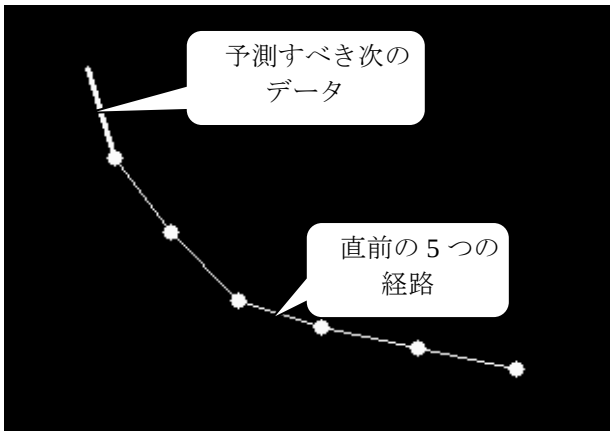


**図 5　一つ目のモデル用の学習データの例**

直前の 5 つの線分は車が直前に移動した経路であり，上の太い線分が予測するデータを示している．図 6 に二つ目のモデル用のデータの例を示す．

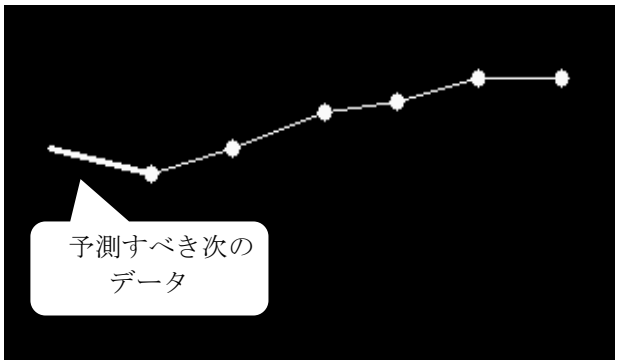


**図 6　二つ目のモデル用の学習データの例**

この例では予測すべきデータが上向きのデータになっているが，この方向は直前の 5 個のデータからは予測が難しいデータになっており，そのためにデータ分割用モデルでは誤差が大きくなったことが分かる．各フレームの動きからデータを作成し 10 本の動画から得られたデータ数はトータルで 1516 個であり，今回は一つ目のモデルの作成に 3/4 のデータを使用し，残りを二つ目のモデル作成に使用した．各モデルで使用したデータの内訳を表 2 に示す．

**表 2 予測モデル作成に使用したデータ数**

| No | モデル | データ用途 | データ数 |
|---|---|---|---|
| 1 | 一つ目のモデル | 学習用 | 909 |
| 2 | | 検証用 | 228 |
| 3 | 二つ目のモデル | 学習用 | 303 |
| 4 | | 検証用 | 76 |

予測に使用したモデルは 4 層の全結合ネットワークであり，各層は 600 個のノートを持つシンプルな構造をもつ．また 5 層目の関数として最小二乗誤差関数を使用した．図 7 にネットワーク構造を示す．

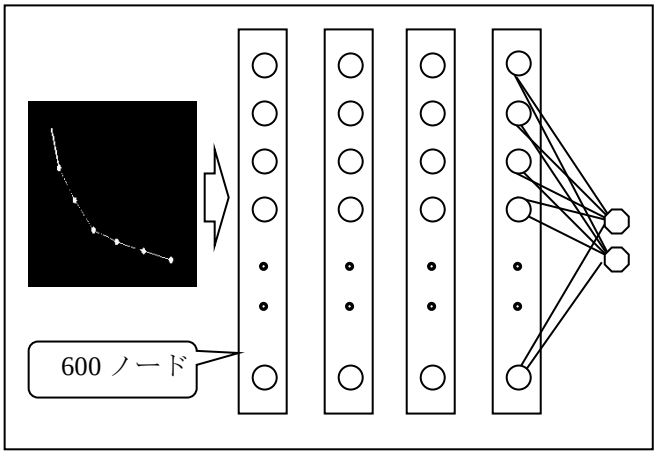


**図 7　予測モデルのネットワーク構造**

学習回数は 50 エポック実行し，データ分割用モデルで学習した場合のロス値は 0.15 であり，一つ目のモデルで学習した場合のロス値は 0.08 と改善し二つ目のモデルでのロス値は 0.38 となった．

次に評価用モデルを生成する方法を詳細化する．人が頭の中で学習する場合を考えると，座標の厳密な値の違いが気になることはあまりないことが多いことから，座標の値についてはある程度まるめた値で学習して問題ないと思われる．そこで動画の画像を縦 30 横 40 のグリッドに分割し，グリッド単位での変化を学習することにする．3 章の手順(2)を以下のように詳細化する．

(2-1) 動画を縦 30 横 40 のグリッドに分割し，オブジェクトの経路をグリッド上にプロットする

(2-2) 同じグリッドを何度も通過した場合は，一回通るごとにグリッドの値をインクリメントする

(2-3) 動画の途中の状態も学習データとして使用するために各途中フレームでのプロットも作成する

(2-4) グリッドの値を畳み込みニューラルネットワークの入力として正規化して評価値を学習する

図 8 に車の絵の経路をグリッド上にプロットした例を示す.

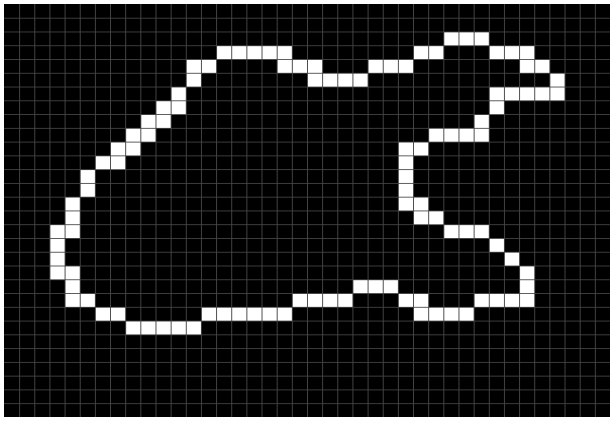

**図 8　オブジェクトの経路のプロット例**

手順(2-3) において動画の途中のフレームまでプロットを作成した例を図 9 に示す.

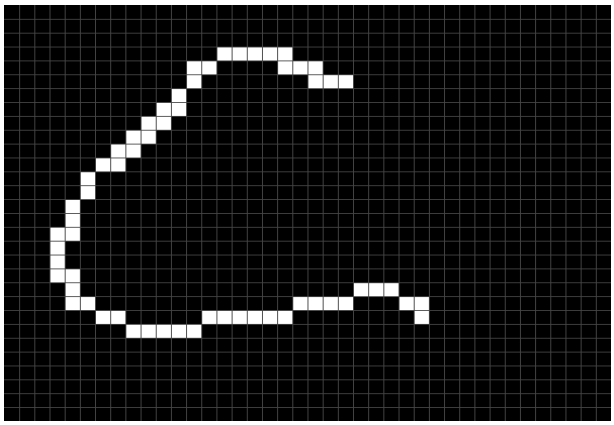

**図 9　動画の途中までプロットした例**

この途中の絵の評価値は動画の中間であるため図 8 の評価値が 1.0 の場合は評価値を 0.5 として学習する. 同様に動画の各フレームで作成した絵を全て学習データとして利用する. 10 本の動画から作成した画像データはトータルで 1504 個であり, このうち 1203 個を学習データとし 301 個を検証データとして利用した.

評価値の学習にはグリッドと同じサイズである縦 30 横 40 の値を入力として受け取る畳み込みニューラルネットを使用する. グリッドをチャネルに変換するときに 4 つの層に分割し, 図 10 ように一回のみ通過したグリッドは 1 層目のチャネルの値を 1 とし, 二回通過したグリッド 2 層目のチャネルの値を 1 とし, 四回以上通過したグリッド 4 層目の値を 1 とする. 評価値は正規化した 0.0～1.0 の値を使用する. 以下の図 10 にデータの正規化する様子を示す.

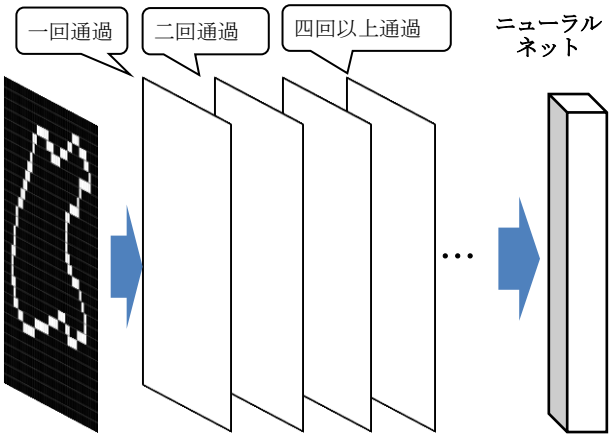


**図 10　入力データの正規化**

こなお, オブジェクトが複数の場合は, オブジェクトの数に応じてチャネルを増やす. 表 3 に評価用モデルの入出力データの内容を示す.

**表 3 評価用モデルの入出力データ**

| No | 入出力 | データ種別 | 値 | チャネル数 |
|---|---|---|---|---|
| 1 | 入力 | 一回通過した位置 | 通過した: 1<br>通過しない:0 | 1 |
| 2 | | 二回通過した位置 | 通過した: 1<br>通過しない:0 | 1 |
| 3 | | 三回通過した位置 | 通過した: 1<br>通過しない:0 | 1 |
| 4 | | 四回以上通過した位置 | 通過した: 1<br>通過しない:0 | 1 |
| 5 | 出力 | 評価値 | 正規化値<br>(0.0~1.0) | ― |

評価用モデルでは 2 次元のニューラルネットワークで特徴を抽出した後, 全結合のネットワークで評価値を計算する構成とし, 小規模なネットワーク構造を使用した. 表 4 に使用したネットワークの構造の詳細を示す.

**表 4 評価用モデルのネットワーク構造**

| 層 | 次元数 | フィルタサイズ | フィルタ数 | ノード数 | 活性化関数 |
|---|---|---|---|---|---|
| 1 | 2 次元 | 3×3 | 32 | ― | ReLU |
| 2 | 2 次元 | 3×3 | 32 | ― | ReLU |
| 3 | 2 次元 | 3×3 | 1 | ― | ReLU |
| 4 | 1 次元 | ― | ― | 500 | ― |
| 5 | 1 次元 | ― | ― | 1 | ― |

なお, ロス値の計算ではデータのノイズが学習結

果に大きな影響を与えないようにするため平均絶対誤差関数を使用した. 学習回数として 80 エポック実行したモデルの誤差は 0.04 であり，その際の学習曲線を図 11 に示す.

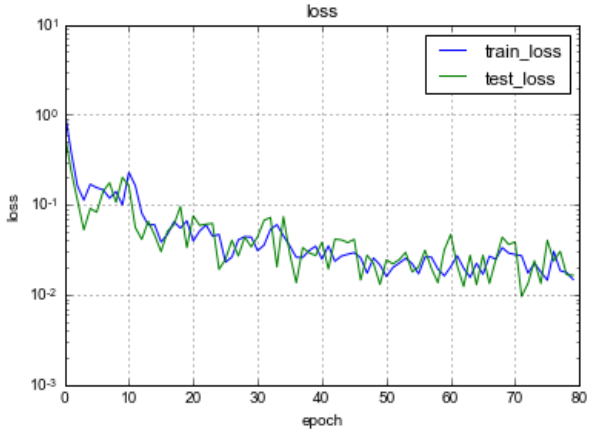


**図 11　評価用モデルの学習曲線**

## 6 シミュレーション実行の実装例

4 章の手順(A)のオブジェクトの初期化については，今回はコースの右下にあるスタートラインの位置に配置した．4 章の手順(B)のオブジェクトの次の位置の予測については二つの予測モデルを使用して一つずつ予測することで二通りの位置を予測した．そして，コースアウトすることがないように予測した位置を，動画でオブジェクトが通過した位置のなかで，予測した位置に最も近い位置に補正した．以下に詳細化した手順を示す.

(B-1) 各オブジェクトの位置の候補を予測用モデルで二通り生成する

(B-2) 動画の中でオブジェクトが通過した位置の中で最も近い位置に，二つの予測値を補正する

図 12 に予測した二つの値を示す.

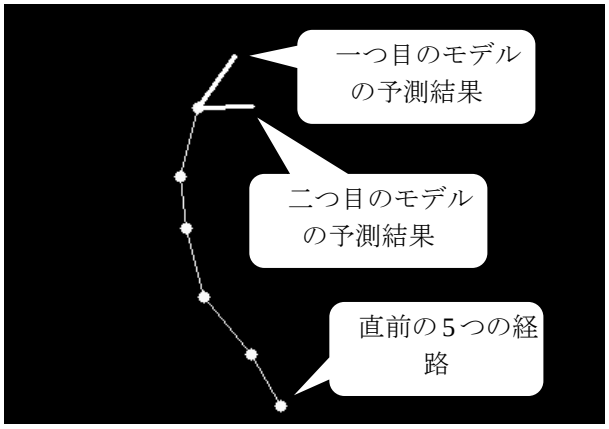


**図 12　車の次の位置の予測結果**

直前の 5 つの線分は車の絵が移動した経路を示しており，上の二つの太い線分が予測された位置を示している．　次に 4 章の手順(C)で二つの予測結果を評価するときの問題として，一回予測しただけではグリッド単位でみると絵としては同じ絵になってしまう問題がある．そこで，予測を 5 回再帰的に繰り返して予測経路を伸ばしてから評価を行うことにした．結果として経路の種類は 32 通りになり，それぞれについてグリッドのプロットを作成してそれらを評価モデルで評価してその平均値を元の予測の評価値とする. つまり 4 章の(C)を以下の様に詳細化する.

(C-1) 位置の予測を再帰的に 5 回繰り返し 32 通の経路を作成する

(C-2) 32 通りの経路それぞれグリッドにプロットして評価用の絵を作成する

(C-2) 評価用モデルを使用して 32 個の経路を評価してその平均値を元の予測位置の評価値とし，二つの予測の評価値が高い方を選択する

図 13 に予測を 5 回繰り返して 32 通りの経路を取得した状態を示す.

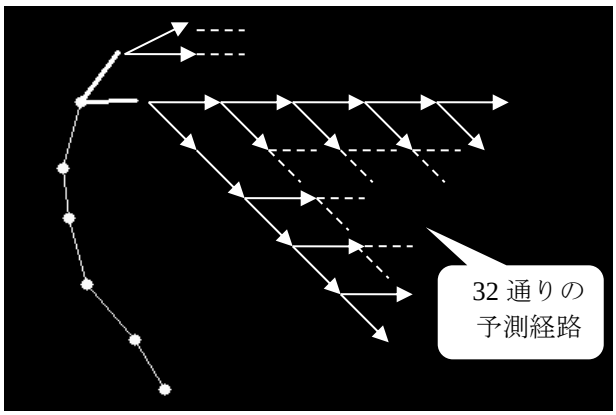


**図 13　予測を 5 回繰り返した状態**

シミュレーションを 13 回実行して最も短時間で周回する走行ルートが得られた結果を図 14 に示す.

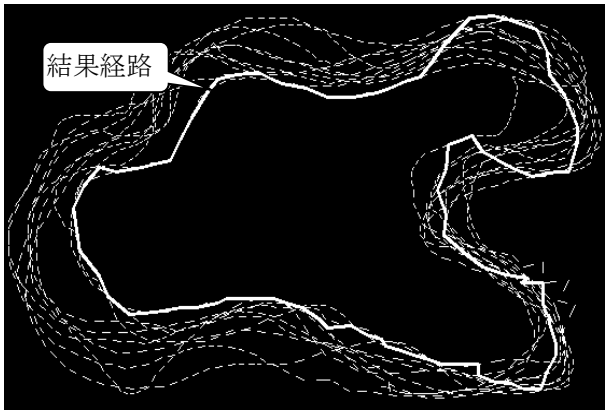


**図 14　シミュレーション実行結果**

図 14 では 10 本の動画の経路を点線で示し，シミュレーション結果を実践で示している．10 本の動画のなかで最も速いタイムは 4.7 秒であるのに対し，シ

ミュレーションの結果は 3.5 秒となり，動画で最も速い場合よりも良い結果が得られた．また，得られた走行ルートでは右下から中央上までは最も内側の最短の経路を通過しており，一方，右上のヘアピンカーブでは一旦外側に膨らむアウトインアウトのルートを通るコーナリングテクニックを使用していると解釈することができる．

## 7 モンテカルロ木探索の適用

6 章のシミュレーションの実行では以下の図 15 のように逆走して失敗してしまう場合が多い．

**図 15　シミュレーションの失敗例**

6 章のように 5 回繰り返す方法でこのような失敗例を除いた成功率は 42%となった．この理由はデータが少ないため評価モデルが十分に学習されていないためと考えられる．そこで，モンテカルロ木探索のように最後まで予測を繰り返して評価値に明確な差が出るような状態にしてから評価値を求める方法が有効と考えられる．しかし，一般の問題では終了状態になるまでの探索の深さが非常に長い場合もある．そこで，本研究では探索の深さを一定の深さに固定した．具体的には深さ 5 と 15 と 30 の 3 種類の場合について成功率を比較して検証した．以下の図 16 に探索の深さが 30 の場合の探索例を示す．

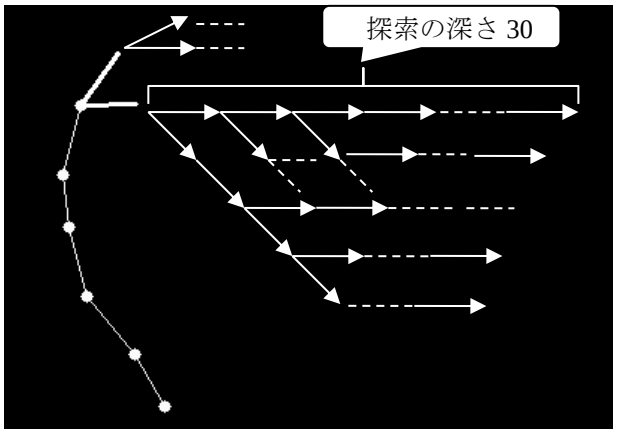


**図 16　探索の深さが 30 の場合**

結果を表 5 に示す．なお，本システムでは CPU として Core i5-2520M 2.5GHz 2.5GHz を使用し GPU は使用していない．

**表 5 モンテカルロ木探査の適用結果**

| No | 探索の深さ | 対策の回数 | シミュレーション実行時間 | 成功率 | 平均周回時間 |
|---|---|---|---|---|---|
| 1 | 5 | 32 回 | 151 分 | 42% | 3.9 秒 |
| 2 | 15 | 32 回 | 621 分 | 48% | 3.3 秒 |
| 3 | 30 | 8 回 | 301 分 | 58% | 3.3 秒 |

その結果, 探索の深さを 30 まで増やすに従い成功率が 58%まで改善し, さらに周回する時間の平均も 3.9 秒から 3.3 秒まで改善するという結果が得られた．結果として，汎用人工知能においてもモンテカルロ木探索が有効であることが確認できた．

## 7 今後の課題

今回はオブジェクトの数が一つであり，属性はオブジェクトの移動のみのシンプルな場合について検証した．今後以下の 3 つの拡張を行う必要がある．

・複数のオブジェクトへの対応
・他のオブジェクトから影響を受ける場合に対応
・移動以外の属性の変化への対応

これら 3 点に対応したモデルのネットワーク構造は, 複数の種類に分かれる可能性があるが，その場合は予測モデル用と評価モデル用にそれぞれ幾つかネットワーク構造を予め定義しておき，試しに全てのネットワークで学習を行い，最も誤差の小さいモデルを採用することで, ドメインの知識を不要にできる．

## 8 まとめ

本研究では汎用人工知能の実現方式である PEM 方式をオブジェクトが一つだけのシンプルな問題に対して適用した．そして，データとして 5 秒の動画 10 本という極端に少数のデータから作成したモデルに対してモンテカルロ木探索を適用して探索を深くするほどシミュレーション結果が改善することを示した. 今後は他のより複雑な問題にも PEM 方式が汎用人工知能の実現方式として有効であり，さらにモンテカルロ木探索が同様に有効であることを確認していく．

## 参考文献：

[1] 小池賢一: 動画から抽出した動きの情報に基づく汎用人工知能の実現方式, 電子情報通信学会,情報論的学習理論と機械学習研究会, IBISML2019-27 pp.61-66

[2] 松原仁編：コンピュータ囲碁―モンテカルロ法の理論と実践―，共立出版，pp.1-91(2012)