3E3-06

# サッカーエージェントにおけるログファイルを利用した模倣学習

Imitative Learning from the Logs of a Soccer Simulator

山本 晃義[*1] 岡 夏樹[*1] 野田 五十樹[*2]
Akiyoshi Yamamoto Natsuki Oka Itsuki Noda

[*1]京都工芸繊維大学大学院 工芸科学研究科 電子情報工学専攻
Kyoto Institute of Technology

[*2]産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

In late years, multiagent systems in which agents achieve a task collaboratively attract attention. This study aims to achieve collaboration and learning in multiagent systems with RoboCup Soccer Simulator. However, the learning in multiagent systems has a problem with the way of getting supervised signals and difficulty for programmer to set criteria for evaluation. This study tries to acquire the skill in dribble by imitation learning from the logs of other teams.

## 1. はじめに

近年，エージェントが複数存在するマルチエージェントシステムに関する研究が注目を浴びており，複雑な問題や，そもそも単一のエージェントでは解決が不可能な問題を複数のエージェントが互いに協調することで解決する協調行動の実現は，工学的及び認知科学的観点から非常に興味深い話題である。

しかし，複数のエージェントが存在する環境におけるエージェントのすべての行動を設計することは非常に困難であり，学習によるエージェントの行動獲得手法の研究に対する期待が高まっている。

マルチエージェントシステムにおける学習を単一のエージェントの場合と同じように扱うと，状態数が組み合わせ的に増えることがしばしばある [1]。この状態空間の爆発問題を避けるために，扱う問題を分割し階層的に扱うことで状態数を減らす。またエージェントの学習手法には,「入力」に対する「出力」がセットとしてエージェントに渡され，エージェントは「入力」に対して出来る限り正しい出力を学習しようとする，教師付き学習があるが，エージェントの行動に対し，その正しい答えを全て教えてやる事は非常に困難である。そこで我々は優秀な他チームの行動をログファイルから学習する模倣学習を用いる。

本研究では，不完全知覚，情報の不確実さ，実時間処理など現実の問題に近い課題を多く含み，マルチエージェントシステムの標準問題として多くの研究がされている RoboCup Soccer Simulator 環境を用いて，ログファイルからサッカーの代表的な行動であるドリブルの模倣による獲得を試みた。

## 2. RoboCup Soccer Simulator

RoboCup は，ロボット工学と人工知能の融合，発展のために自律移動ロボットによるサッカーを題材として日本の研究者たちにより 1995 年に提唱されたものである [2]。RoboCup には 5 つのリーグがあり，その内の 1 つのであるシミュレーションリーグで使用され，本研究でも用いる RoboCup Soccer Simulator のログファイルについて説明する。

連絡先: 山本 晃義 京都工芸繊維大学大学院パターン情報処理研究室 e-mail:yamamoto@vox.dj.kitac.jp

### 2.1 ログファイルについて

Soccer Simulator[3] で扱われるログファイルは以下のように 2 種類のものがある。

- ".rcg"という拡張子のファイル
- ".rcl"という拡張子のファイル

これらのファイルはゲームが終了する際に作成される。

.rcg ファイルは選手・ボールの位置などの進行ログであり，サイクル数・ボールの位置や速度・各選手の情報 (位置や速度，スタミナなど) を含んでいて各サイクル毎の状況を表している。試合のリプレイなどに使用される。.rcg ファイルはバイナリファイルであるが，試合のログを再生するツール rcsslogplayer(version 9.3.3 以上) には，.rcg ファイルを XML ファイル形式に変換するツール (rcg2xml) がある。必要に応じて XML ファイル処理を行うことでログ解析などができる。

.rcl ファイルはサイクル数とその時サーバが各クライアントから受け付けたコマンドとそのパラメータなどの情報が含まれていて，各エージェントが判断した行為を表している。

それぞれのファイル内容を fig.1・fig.2 に示す。本研究では，この 2 種類のログファイルから優秀なエージェントの行動を切り出し，サッカーの基本的な行動であるドリブルを学習させる。

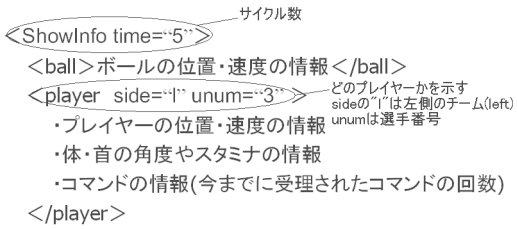


fig. 1: .rcg ファイル (xml ファイル変換後)

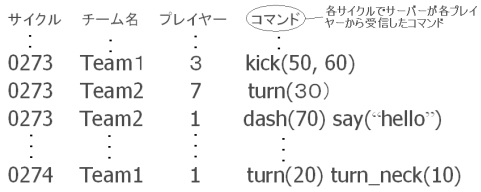


fig. 2: .rcl ファイル

## 3. 学習について

RoboCup Soccer Simulator においてサッカーの代表的な行動 (パス，シュートやドリブルなど) をエージェントに機械学習させる場合，通常，より精度の高いものを求めると以下のような情報が必要になってくる。(パス，シュート，ドリブルの場合)

- パス：敵の位置や距離，味方との距離，ボールの情報
- シュート：キーパや敵の位置，ゴールまでの距離や方向，ボールの情報
- ドリブル：敵の位置や距離，ボールの情報

このような場合，複数のエージェントが各々独自に判断して行動を取る RoboCup サッカーではエージェントが得られる情報は絶えず変化し入力が多すぎてしまうことが多く，学習する場合，状態数が組み合わせ的に増えることになる。さらに複雑なマルチエージェントシステムにおいて，エージェントの行動に対し，正しい答えを全て教えることは非常に困難であり，パスやシュート，ドリブルなどの行動は，コマンド (kick,dash,turn) の組み合わせで設計されるもので，どのようにしてコマンドのパラメータに対して最適な教示を与えるかが問題である。

そこで本研究では階層型学習と模倣学習を行う。

### 3.1 階層型学習

階層型学習は Peter Stone が提案したもので，マルチエージェントシステムなどにおける複雑な問題を機械学習を用いて解決する場合，単一の学習でやるのではなく問題を階層的に分割し，各層に適した学習法を使うというものである [4]。扱う状態変数が異なる学習器を統合することで，結果として多数の状態変数を利用する一連のタスクを扱うことが可能となる。より抽象的な高レベルの層とより具体的な低レベルの層に分け，学習は各層独立して行われ，下から上へと段階的に行われる。下の層での出力は次の上の層へと送り込まれる。

この学習方法は階層的に問題を分割することで状態数を減らすことができるが，階層の分け方や各層に適応する学習法をどのように決めるかは，開発者が提案するものである。

今回提案する階層化は以下の通りである。

- **行動選択レベル**：敵・味方の情報から状況に応じて適切な行動 (パス，シュート，ドリブル) を選択する
- **基本動作レベル**：各行動を実現するために適切なコマンドとそのパラメータを組み合わせる

行動選択レベルでは，敵・味方の情報の状況を入力とし判断された行動を出力とする。基本動作レベルでは，味方やボールの情報 (シュートの場合キーパーの位置やゴールまでの距離) を入力とし適切なコマンドとそのパラメータを出力とする。

基本動作レベルで敵情報を含まない理由は，上位層の行動選択レベルで敵情報を扱って行動選択を適切に判断しているとすれば，パスやドリブルなど行動そのものを学習する際には敵情報を含める必要がないと考えられるからである。これによって扱う問題が簡略化され，状態数も減らすことが可能である。

最終的に目指すシステムとしては，あるエージェントがボールを持った時を想定し，エージェントが状況に応じて行動を選択し，選択された行動を適切なコマンドとそのパラメータで実現するというものである。(fig.3)

本研究では fig.3 に示す全体構成の内，基本動作レベルにおけるドリブルの学習を目指し，学習モデルはニューラルネットワークを使用する。

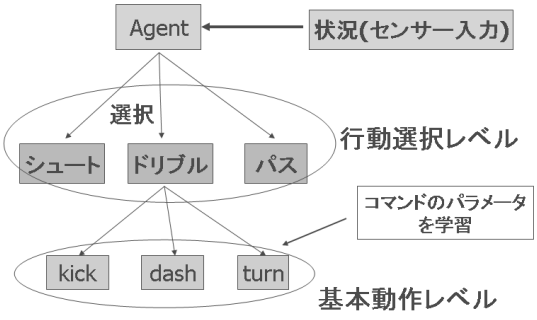


fig. 3: 目標とする階層型学習システム

### 3.2 模倣学習

模倣学習は，他のエージェントの行動の良い事例を集めて，それを模倣することで能力を改善しようというものである [5]。今回は他チームのドリブルを模倣するのだが，どのように事例を集めるかを説明する。2.1 でログファイルの説明を行ったが，.rcg ファイルはサイクル毎の状況を，.rcl ファイルはサイクル毎に各エージェントが実行したコマンド列を示している。.rcl ファイルから学習させたい行動 (コマンド列) を切り出し，それに対応する学習に必要な情報を .rcg ファイルから取得する。.rcg ファイルから得られた状況を入力とし，.rcl ファイルから得られたコマンドとそのパラメータを出力とし学習させる。

### 3.3 ドリブルの学習

ドリブルという行動は，ボールを蹴ってまたボールを追いかけるという行動であり，あるエージェントが kick コマンドを使用して (ボールを蹴る)，短いサイクルの中で，再び同エージェントが kick コマンドを使用するまでのコマンド列がドリブルの最小単位といえる。これらのコマンド列を .rcl ファイルから切り出す。

しかし，ドリブルはボールを蹴った後，状況が変わってしまうので時系列を扱う学習となり，学習モデルが複雑になる。そこでドリブルを以下のように分ける。

1. ボールへのアプローチ (turn,dash を駆使)
2. ボールを適度な強さで蹴る (kick を調整)

つまりドリブルという行動を (1) と (2) の二つの行動を合わせて 1 つと見なす。このように分けて学習を行えば，パラメータ学習は楽になるはずである。(1) では turn や dash の POWER パラメータのみを学習，(2) では kick パラメータのみを学習させる。それぞれの入出力は以下のようにする。

1. ボールへのアプローチ

   入力：ボールと選手との距離，選手の体の向きに対するボールとの角度 (fig.4)

   出力：turn or dash コマンド，そのコマンドのパラメータ

2. ボールを適度な強さで蹴る

   入力：ボールと選手との距離，選手の体の向きに対するボールとの角度 (fig.4)

   出力：kick コマンドの *Power* パラメータ

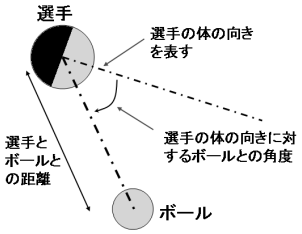


fig. 4: 状態表現

ドリブルの場合，どちらの方向に進むのかは敵情報が必要になってくるのだが，今回の場合は上位層の行動選択レベルでドリブルと判断した場合にボールをどちらの方向へ持っていくか (あるいは仮想的なゴールがどちらであるか) の情報を下位層に与えるつもりなので，今回の“ ボールを適度な強さで蹴る ”という学習において，kick コマンドの角度 (*Dir*) は出力としなかった。

## 4. 実験

“ ボールへのアプローチ ”と“ ボールを適度な強さで蹴る ”という行動を他チームの行動ログから必要な情報を切り出し, 3.3 で述べた入力・出力でニューラルネットワーク [6] を使用して学習させる。今回，ドリブルを模倣するチームは RoboCup-2003 の優勝チームである UvA Trilearn[7] である。このチームの試合ログからドリブルと思われる行動を切り出し，.rcl ファイルからコマンドとそのパラメータ，.rcg ファイルからそれに対応する状況を切り出す。十分な事例が集まったら各学習モデルで学習を行う (fig.6，fig.5)。

### 4.1 実験結果

今回の実験では，ニューラルネットシミュレータ tlearn[8] を使用した。“ ボールを適度な強さで蹴る ”の教師信号は約 200，“ ボールへのアプローチ ”の教師信号は約 600 ほど集めて学習させた。学習率は 0.1，学習回数は 2000 回とした。その結果を fig.5 と fig.6 に示したネットワーク全体のエラー推移で fig.7 と fig.8 に示す (エラー推移は結合加重の初期値で多少変化するのだが，これらの図ではその一例を示している)。これ

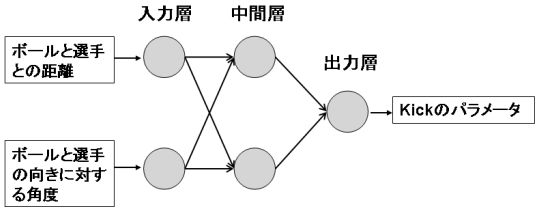


fig. 5: 学習モデル：ボールを適度な強さで蹴る

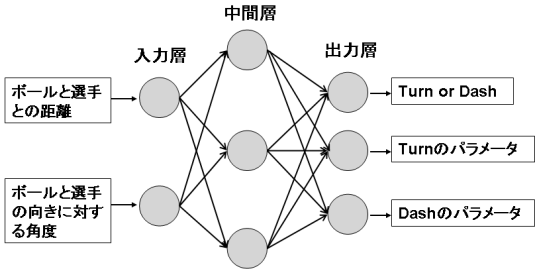


fig. 6: 学習モデル：ボールへのアプローチ

によって学習した結果が全体としてどれだけの誤差を含むかがわかる。

ここでのエラーは次のように与えられる。

$$\text{トータルエラー} \quad = \quad \sqrt{\frac{\sum(\vec{t_k} - \vec{o_k})^2}{k}}$$

$k$ : 学習パターンの個数

$\vec{t_k}$ : パターン$_k$についての教師信号ベクトル

$\vec{o_k}$ : パターン$_k$についての出力ベクトル

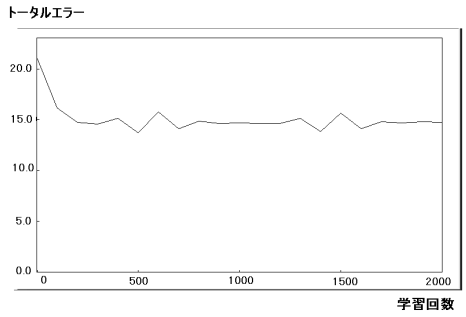


fig. 7: fig.5 のニューラルネットワークの学習曲線

### 4.2 実験結果の考察

4.1 の実験結果よりエラー推移から読み取れる誤差は，fig.7 と fig.8 を見てわかるように，“ ボールを適度な強さで蹴る ”場合と“ ボールへのアプローチ ”の場合，両方ともかなり大きいものだった。ドリブルをする際の kick コマンドのパラメータは，平均して 30〜40 くらいの大きさを取るのが普通である

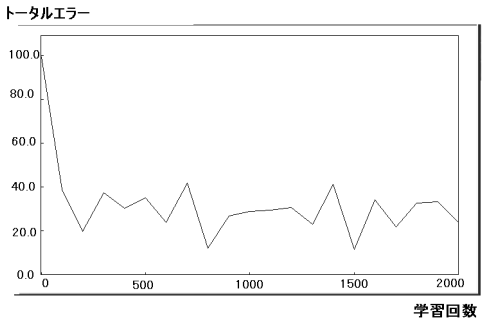


fig. 8: fig.6 のニューラルネットワークの学習曲線

が，fig.7 から読み取れるトータルエラーは約 14〜15 である。これはかなり大きなエラーであり，“ ボールへのアプローチ ”に関しても同様のことが言えて，dash コマンドのパラメータは平均して 90 くらいの大きさを取るが，トータルエラーは約 23 である。

この原因の一つは，抽出した教師信号にあると思われる。今回，他チームのログファイルからある選手が kick コマンドを使用して再び kick コマンドを使用した区間をドリブルとして切り出していたが，その kick コマンドが本当にドリブルを行っている最中の kick なのかは確かではなかった。ボールをトラップする際にボールの勢いを消すための kick であったりする場合は，時にパラメータが 80 を超えたりすることもあった。

このように教師信号そのものに問題があると考えられるため，使用した教師信号における入力と出力との関係について分析・検討を行った。“ ボールと選手との距離 ”と“ kick コマンドパラメータ ”との相関図を調べてみる (fig.9)。この分布を見ると，“ ボールと選手との距離 ”と“ kick コマンドパラメータ ”との相関はそれほど強いものとは言えない。つまり“ ボールと選手との距離 ”は，kick コマンドのパラメータ学習をする入力としては不十分であると言える。より関係の強いと思われる入力を新たに見つける必要がある。またこの分布図を見ると分布の回帰曲線からの幅が約 60 ほどあり，回帰曲線から点線までの距離は約１５である (fig.9 の点線は回帰曲線より上方・下方の平均を表している)。結合加重の様子から“ ボールと選手との距離 ”だけが“ kick コマンドパラメータ ”に関わっていると言えないので断言はできないが，これがトータルエラー約 14〜15 となる原因であると考えられる。

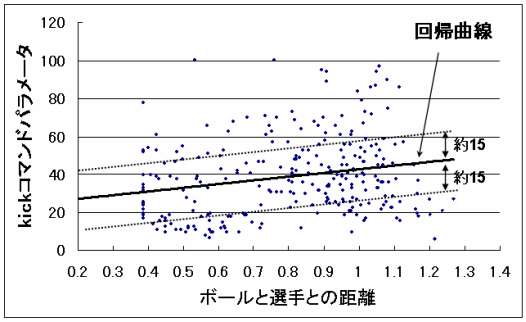


fig. 9:“ ボールと選手との距離 ”と“ kick コマンドパラメータ ”の相関図

同様に“ ボールへのアプローチ ”に関しては，dash コマンドの例はたくさん取れたが，turn コマンドに関してはほとんど例が取れなかった。その理由として，今回模倣を行った UvA チームの行動をログで見ると，turn による体の方向修正は kick コマンドを使用する前に行われていた。あらかじめドリブルする方向を定め，その方向に体を向けることで途中修正をすることなくそのまま dash コマンドを使い続けていたと思われる。ドリブルの途中，敵と出くわして方向転換する際に turn コマンドを使用する場面は見られたが，当初予想していた turn・dash コマンド列による“ ボールへのアプローチ ”ではなかった。

## 5. まとめ

本研究では，ログファイルから他チームの行動を模倣することでドリブルの獲得を試みたが，4.1 の実験結果から分かるように学習結果の誤差が十分には下がらなかった。推定される原因として，ログから得たドリブルと思われる教師信号が想定していたものと異なったことが考えられる。今回，UvA の行動ログからドリブルを模倣しようとしたが，どのように真似ていくか・どうやって必要な教示信号を集めるかが今後の課題となる。

## 参考文献

[1] 高玉 桂樹：マルチエージェント学習，pp.132/pp.142，コロナ社，2003

[2] RoboCup 公式ホームページ：
http://www.robocup.org/

[3] Soccer Server System：http://sserver.sourceforge.net/

[4] Peter Stone：Layered Learning in Multi-Agent Systems, CMU Computer Science Tech Report CMU-CS-98-187, 1998

[5] NODA, Itsuki：Hierarchical Hidden Markov Modeling for Team-play in Multiple Agents，Proc. of IEEE Conf. on System 2003，2003，

[6] 麻生 英樹：ニューラルネットワーク情報処理，pp.39/pp.54，産業図書，1991

[7] the official UvA Trilearn website
http://www.science.uva.nl/ jellekok/robocup/index.html

[8] Tlearn software：http://crl.ucsd.edu/innate/tlearn.html