2i3-03

ロボカップサッカーシミュレーションの Keepaway における協調行動の学習

Learning of Cooperation Behavior in Keepaway of RoboCup Soccer Simulation

伊佐野勝人[*1] Shoto Isano
片上大輔[*1] Daisuke Katagami
新田克己[*1] Katsumi Nitta

[*1] 東京工業大学大学院総合理工学研究科
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology

When Reinforcement Learning is applied to RoboCup soccer simulation, the number of state-action pairs becomes too large, it takes too much time to settle the learning curve. Therefore we applied Reinforcement Learning to Keepaway which restricts states, actions, the area of the field and the purpose. In this study we analyzed the states which the task can't be continued by only the selectable actions, introduced dribble as new actions in the states and showed its usefulness by the experiments.

1. はじめに

ロボカップサッカーシミュレーション(以下 RCSS)とは, 自律動作する 11 人のエージェントをチームとして構成し, サッカーを通じてその機能を評価する場である. RCSS の試合では, チームワークを構築する方法の一つとして強化学習を用いることが考えられるが, 状態行動対が大きくなり過ぎるため, 学習が収束するのに時間がかかり, 多くの実験を行うには不都合である. そこで限られた領域内で Keepers というチームが Takers という相手チームにボールを取られないように回し続けることだけを目的とした Keepaway(図 1)というタスクを用いる. Keepaway は RCSS の試合に比べて環境が単純なため, 強化学習のような繰り返し試行する実験に適している. 実際, Keepers に強化学習を適用した実験では, ハンドコーディングよりもチームワークを構築する方法として有効的であることが証明されている[Stone 01, 荒井 06]. Keepaway は, ボールをキープするための行動の組み合わせやそれらを実行するタイミングを計るタスクとして最適であり, ここで得られた技術はそのまま RCSS の試合にも応用することができる.

しかし Stone, 荒井ら[Stone 01, 荒井 06]の行なった研究では, Keepers の行動選択肢がホールドボールと味方へのパスしかなく, これだけで Keepaway を行うには限界があり, また RCSS の試合へその技術を移行させることも難しいと考えられる.

そこで本論文では, Keepers の行動選択肢として, ホールドボールと味方へのパスだけではタスクが終了する状態, すなわち新たな行動を追加すべき状態の条件を設定し, それらの状態に新たな行動としてドリブルを追加し, その有効性を実験によって確かめる.

2. 問題設定

2.1 Keepaway

Keepaway ではボールをキープするチームを Keepers, カットするチームを Takers と呼ぶ. Keepers は Takers からボールを奪われないようにボールをキープすることを目的とし, Keepers が

連絡先:伊佐野勝人, 東京工業大学総合理工学研究科知能システム科学専攻, 神奈川県横浜市緑区長津田町 4259, isano@ntt.dis.titech.ac.jp

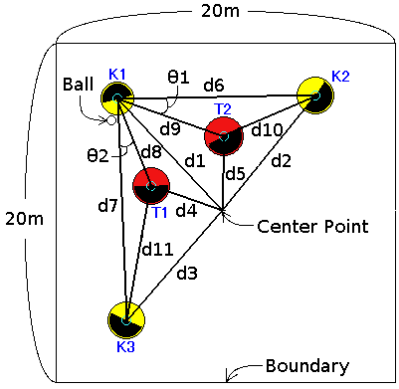

図 1 Keepaway の各名称と状態変数

Takers にボールを取られるか, ボールがエリアから出るとエピソードが終了する. 各エピソード開始時, Takers は全員左下のコーナーに, Keepers は残りの 3 つのコーナーにランダム配置され, Keepers が 3 人を超える場合はエリアの中央に配置される.

2.2 従来研究

Stone, 荒井ら[Stone 01, 荒井 06]の研究では, 図 1 のように 20m×20m のエリアで 3 対 2 の Keepaway を行なっている. ここで Keepers のうちボールを持ったエージェントを K1, それ以外の Keepers は K1 から近い順に K2, K3, Takers は K1 から近い順に T1, T2 と呼ばれ, 強化学習は K1 のみに適用される. K1 は 11 個の距離変数(d1～d11)と 2 個の角度変数($\theta 1$, $\theta 2$), 計 13 個の状態変数を観測し, その組み合わせから 3 種類の行動, ホールドボール, K2 へのパス, K3 へのパスの価値を計算し, ε -greedy によって選択された行動を実行する. 状態の近似関数には一次元タイルコーディングが用いられている. 報酬について, Stone ら[Stone 01]はステップごとに一定のプラスの報酬を与え, 荒井ら[荒井 06]はエピソード終了時に, その原因となるエージェントを明確にしてマイナスの報酬配分を行なっている. その結果, 荒井らの報酬設計が優れていることが実験から証明されている.

3. ドリブルの導入

3.1 従来研究の解析

Stone, 荒井ら[Stone 01, 荒井 06]の研究について, 表 1 のように各状態における行動の評価を単純化した例をあげて説明を行う. 縦軸の State1～3 は 13 個の状態変数を組み合わせた状態で, HoldBall, PassK2, PassK3 は各状態における 3 種類の行動選択肢であるホールドボール, K2 へのパス, K3 へのパスにそれぞれ対応する. また数値は行動評価であり, 各状態でそれぞれの行動を選択した結果, 5Step 以内にエピソードが終了しなかった回数から終了した回数を引いたものとする.

State1 ではどの 3 つの行動も評価がプラスなので, どれを選択しても 5Step 以内にエピソードが終了することはないが, 最も成功回数の多い PassK3 を選択するのが最善である. また State2 では PassK2 と PassK3 の評価がマイナスであるが, 評価がプラスである HoldBall を選択すれば 5Step 以内にエピソードが終了することはない. 問題となるのが State3 で, どれを選択しても 5Step 以内に必ず失敗することになり, このような状態が新たな行動を追加すべき状態であると言える.

表 1 各状態における行動の評価と新たな行動の追加

	HoldBall	PassK2	PassK3
State1	+100	+200	+300
State2	+500	-200	-100
State3	-200	-300	-400
…	…	…	…

では State3 のような, 新たな行動を追加すべき状態とは具体的にはどのような状態か. まず HoldBall を実行すると 5Step 以内にボールを取られるため, 敵が接近している状態と考えられる. さらに PassK2 と PassK3 を実行すれば失敗することから, 味方へのパスコース上に敵が存在するとも考えられる. 図 1 の状態変数を用いて考えるならば, K1 と T1 との距離変数 d8 が短く, $\theta 1$ と $\theta 2$ の角度変数が非常に小さい場合と言える. 以上から,

1. 敵が接近している
2. 味方 2 人へのパスコースがふさがれている

という条件に当てはまる状態が, 新たな行動を追加すべき状態であると言える.

3.2 ドリブルの導入

新たな行動を追加すべき状態では, パスコースをふさがれた状態で敵が接近していることから, この敵の動きを交わす必要があり, ドリブルが効果的であると考えられる. ドリブルの種類としては, 向かってくる敵を除ける方向へ動く必要である. エリアの境界線付近などへドリブルをして行動空間が狭まってしまうことを考えると, ドリブルは K1 からエリアの中心方向に対する法線に沿って行うことが適切であると考えられる. また敵を交わした後はパスコースが開き, 強化学習データからパスを実行する可能性が高い. そのためパスコースが見つかり次第, いつでもボールを制御してパスが出せるように, ドリブルは敵を軽く交わす程度に小さく行うべきである. さらにドリブル方向の先には, 敵が存在しないことはもちろん, 味方も存在しないことが望ましい. これはドリブルした先に味方が存在すれば, パスをする相手もその味方になる可能性が高く, 敵が接近した状態でパスを出すため, パスを出した相手も敵が接近している可能性が高いと考えられるからである. 以上から, 新たな行動を追加すべき状態の条件に

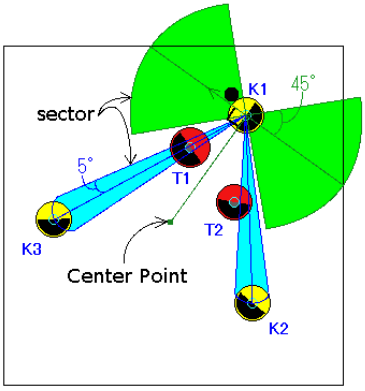

図 2 sector を用いたドリブル条件の判定

3. ドリブル方向に敵も味方もいない

という条件が加わった場合, エリアの中心方向に対する法線に沿った短いドリブルを実行する状態とする.

3.3 パラメータ設定

これまでに述べてきたことをまとめると, 新たな行動を追加すべき状態のいくつかは,「エリアの中心方向に対する法線に沿った短いドリブル」を追加することが効果的であると考えられ, その条件は,

1. 敵が接近している
2. 味方 2 人へのパスコースがふさがれている
3. ドリブル方向に敵も味方もいない

となる. ここではこれらの条件をさらに具体化するためのパラメータの設定を行う.

まず 1. について, 敵が接近している状態がどのような状態かをはっきりと定義できないため, この距離に関するパラメータだけは 3.0～7.0m の間で 1m ごとに変化させて, どの距離が接近している敵をドリブルで交わすのに最適であるかを実験的に確かめる.

次に 2. について, パスコースがふさがれているという状態を定義するために, librcsc [秋山 06]に実装されている sector という扇型のエリア判定関数を用いる(図 2). sector はターゲットポイントへの方向を中心に左右へ扇を広げることでエリアを限定し, 内部のオブジェクトの有無を判定できる. ここでは, K2 と K3 のポジションをターゲットポイントとした sector を 5° 広げることによって, その内部をパスコースと定義する.

3. についても sector を用いる. ここでは, エリアの中心方向に対する法線とエリアの境界線との交点をターゲットポイントとして, 左右へ 45° 広げた sector をドリブル方向と定義する. この内部に敵か味方がいる場合は, ドリブルを実行することができない.

またドリブルの距離については, 実行してから最大で 4Step 以内に再び制御できるように設定する. 4Step 分のドリブルは, Passer のスタミナにもよるが, 2.5m 前後のドリブルが可能で, これは敵を交わすのに十分な距離であると考えられる. 最大とする理由は, ドリブルによってボールがエリアから出ないように考慮したためである.

以上をまとめると

1. T1 が 3.0m 以下～7.0m 以下に存在する
2. K2, K3 までの sector5° の内部に Takers が存在する
3. ドリブルする方向を中心とした sector45° の内部に敵も味方も存在しない

という条件がそろったとき, エリアの中心方向の法線に沿って最大 4Step 以内で制御できるドリブルを実行する.

4. 実験

4.1 実験設定

本実験における強化学習は, 荒井ら[荒井 06]の研究結果を参考にするため, Sarsa(λ)を使用し, パラメータは α = 0.125, γ = 0.95, λ = 0.0, ε = 0.01 と設定する. また関数近似にタイルコーディングを, 報酬関数にエピソード終了の責任を明確にしたマイナスの報酬配分を使用する. ただし, 各エピソードを独立したものにするため, 各エピソードの開始時にすべてのエージェントのスタミナを回復させるようにし, チームプログラムの基本ライブラリには, 個人スキルが最も優れていると考えられる librcsc[秋山 06]を用いる. また本実験ではドリブルによる影響を観察するため, すべてのエージェントの配置を常に認識できるよう RoboCup サッカーシミュレータ[Noda 98]側で設定を行なった.

4.2 各エージェントの動作

エージェントは大きく分けて, Keepers と Takers に分かれ, Keepers はさらに Passer と Receivers に分かれる.

Receivers と Takers の動作はハンドコーディングであり, Receivers は Passer を基準として Keepaway エリアの中心を内心とした正三角形の頂点となるように配置し, Passer に Takers が近付いたときは, Passer へパスコースを見つけさせるように移動する. Takers は 2 人ともボールを追いかける動作を実行し続ける.

Passer の動作は強化学習データのみと, 強化学習データと条件付きドリブルのハイブリッド方式の 2 種類に分かれる. ただし, ドリブルの条件がそろった場合は, 強化学習のデータより優先してドリブルが実行されることになる.

librcsc で実装されている Passer の選択できる行動は,

Body_HoldBall() : その場に留まりながら, 敵プレイヤに取られないようにボールを制御する.

Body_KickMultiStep(Ki) : 必要に応じて複数回のキックをプランニングして Ki にボールをパスする.

Body_Dribble() : 目的の位置へドリブルする. キック後, 設定した回数のダッシュを実行できるようにボールを蹴る.

の 3 種類となる.

4.3 結果

本実験では, K1 を強化学習させて Keepers のチームワークを構築した後, この Keepers を用いて基準とするデータを作成し, T1 との距離ごとに条件付きドリブル実験を行なった.

まず Keepers の強化学習の実験を, エピソード回数を 4500 に設定して行なった. この回数は強化学習が収束する平均を取ったものである. 図 3 は過去 100 エピソードの移動平均を取って, 強化学習が収束していく様子を表した図で, 横軸がエピソード回数, 縦軸がキープ時間となる. 約 1100 エピソードになったあたりで急にキープ時間が上昇するが, それ以後は徐々に上昇する. しかし 3000 エピソードを超えた辺りからはキープ時間の移動平均が荒くなり, そのままエピソードが終了する. 最終的には 4500 エピソードで 16.6 秒の移動平均キープ時間を記録していが, さらに学習させれば, よりキープ時間が延びるような曲線である.

次に強化学習データを元にした動作を実行するプログラムを RL(Reinforcement Learning), 強化学習データと条件付きドリブルのハイブリッド方式のプログラムを Hybrid と呼ぶことにし, RL と Hybrid をそれぞれ 5000 エピソード試行した実験結果をグラフ化して比較を行う. Hybrid は 3.3 節でも述べたとおり, T1 の

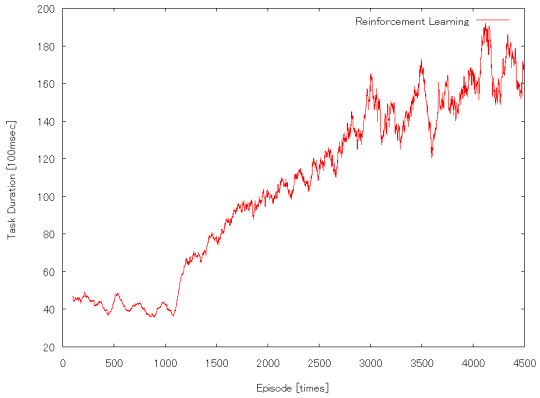

図 3 Keepers の強化学習の収束する様子

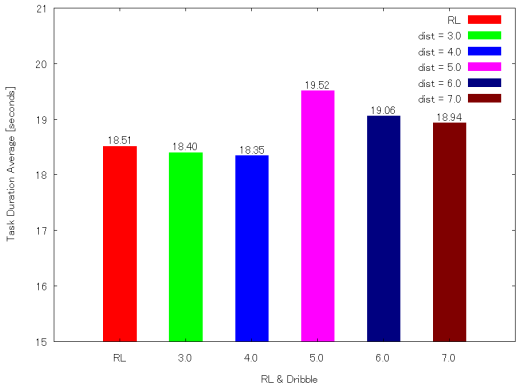

図 4 平均キープ時間の比較

距離条件を 3.0m 以下～7.0m 以下の間で変化させている.

図 4 は各プログラムの平均キープ時間を比較した図である. 横軸が RL と Hybrid を T1 との距離ごとに区分したもので, 縦軸が平均キープ時間である. 一番左の RL を基準とすると, Hybrid は T1 との距離が 5.0m になったときが最も高く, 6.0m, 7.0m になったときでも RL より高いキープ時間を記録している. 一方, 3.0m, 4.0m になったときは RL より低いキープ時間となっている. また最高キープ時間は 6.0m と 7.0m が 140 秒台をそれぞれ 2 回と 1 回, RL と 4.0m が 170 秒台をそれぞれ 1 回ずつ, 5.0m が 190 秒台を 1 回, 3.0m 台が 210 秒台を 2 回記録している.

図 5 はキープ時間ごとの発生回数を比較した図である. 横軸が RL と Hybrid を T1 の距離ごとに区分したもので, 縦軸が発生回数である. キープ時間は一桁台が最も多く, 次に 10 秒台, 20 秒台の順に割合が低くなっていくのがわかる. RL を含め, どの T1 との距離の Hybrid も 20 秒台までで約 4000 回のエピソードがあり, 実験エピソードの大半を占めている.

図 6 は図 5 の一桁台を拡大した図である. ここで図 4 の平均キープ時間と見比べると, 平均キープ時間の長い T1 との距離ほど一桁台の発生回数が低く, それぞれがシンクロしていることがわかる. 平均キープ時間が最も長い 5.0m が一桁台の発生回数が最も低い一方, 平均キープ時間の最も短い 4.0m が一桁台の発生回数が最も多い.

以上から, 3.0m が最高キープ時間の 210 秒台を 2 回も記録したにも関わらず, 全体として平均キープ時間最も低いことがわかった. また 5.0m における一桁台のキープ時間の割合が減少し, 全体として平均キープ時間が一番延びたこともわかった.

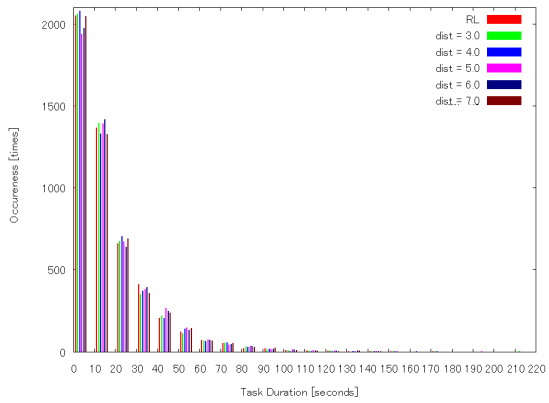

図 5 キープ時間ごとの発生回数の比較

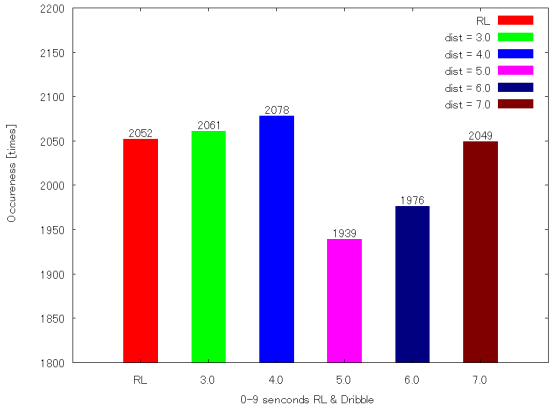

図 6 図 5 の一桁台の拡大図

4.4 考察

実験からドリブルで敵を交わすために最適な T1 との距離は 5.0m になったときという結果が出た. この距離は T1 がドリブルの距離条件を満たして, Passer がドリブルの意思を決定し, 身体の向きを変え, ドリブルを実行するまでに, T1 に追いつかれないための適切な距離であると考えられる. その点から 6.0m と 7.0m はドリブルを実行するには, 少し T1 との距離が離れているため, HoldBall を実行していれば Receivers がパスコースの見つかる位置へ移動するかもしれないため, ドリブルよりも HoldBall の行動価値が高かったと考えられる. 逆に 4.0m と 3.0m は T1 との距離が近すぎて, ドリブルを実行しても T1 が Passer を追いこんでボールを奪うことが可能な距離であると考えられる.

ここで再び, 表 1 を拡張した表 2 を例に出して考察を行う. T1 との距離が 5.0m の状態は, Dribble の評価がプラスとなる State3 と言える. しかし T1 との距離が 6.0m と 7.0m の状態は, ドリブルを追加すれば評価はプラスにはなるが, 繰り返し試行の結果からは HoldBall の方が行動の価値としての評価が高くなる State2 と言える. そのためこの状態は, 新たな行動を追加すべき状態の条件からは外れることになる. そして T1 との距離が 3.0m と 4.0m の状態は, ドリブルを追加しても T1 との距離が近すぎるために, 効果がまったくなく評価がマイナスになってしまう State4 と言える. このような状態では, 例えばスペースへパスのような行動を追加することでタスクが終了してしまう状態を回避できると考えられる.

表 2 各状態における行動の評価とドリブルの追加

	HoldBall	PassK2	PassK3	Dribble
State1	+100	+200	+300	
State2	+500	-200	-100	+100
State3	-200	-300	-400	+100
State4	-400	-200	-300	-100
…	…	…	…	…

以上より, Keepaway で HoldBall, PassK2, PassK3 の行動では乗り切れない状態に, 新たな行動としてエリアの中心方向の法線に沿って最大 4Step 以内で制御できるドリブルを追加する場合, T1 との距離は 5.0m 以下になったときが適切であることがわかった. また T1 との距離が 6.0m と 7.0m 以下の場合は, 新たな行動を追加すべき状態でなく, T1 との距離が 3.0m と 4.0m 以下の場合は, 少なくともエリアの中心方向の法線に沿って最大 4Step 以内で制御できるドリブルを追加すべき状態ではない.

5. おわりに

本研究では, 新たな行動を追加すべき状態の条件を設定し, その状態にエリアの中心方向に対する法線に沿った短いドリブルを追加する実験を行なった. その結果, キープ時間の一桁台の割合が減少し, 全体として平均キープ時間が上昇したことから, T1 との距離条件が 5.0m 以下になったときが, そのドリブルを追加するのに最適な状態であることがわかった. この実験によりエリアの中心方向に対する法線に沿った短いドリブルを追加すべき状態の条件がわかった.

しかし, これは新たな行動を追加すべき状態の一部分でしかなく, まだ多くの新たな行動を追加すべき状態が存在する. またこのような人手による状態の発見では, 全ての状態を新たな行動を追加すべき状態を把握することは時間がかかり過ぎてしまう. そのため表 1, 2 で示したような行動評価がプラスとマイナスで表示され, はっきりと新たな行動を追加すべき状態がわかるシステムを作る必要がある. また状態における行動の最適性を保つためにも, 既存の行動と新たに追加させた行動の価値を比較する必要もある.

参考文献

[Stone 01] Stone, P. and Sutton, R. S.: Reinforcement Learning toward RoboCup Soccer, in Proceedings of 18th International Conference on Machine Learning, 2001.

[荒井 06] 荒井幸代, 田中信行: マルチエージェント連続タスクにおける報酬設計の実験的考察, 人工知能学会論文誌, 2006.

[秋山 06] 秋山英久: ロボカップサッカーシミュレーション 2D リーグ必勝ガイド, 秀和システム, 2006

[Noda 98] Noda, I. Matsubara, H., Hiraki, K., and Frank, I.: Soccer server: A tool for research on multiagent systems, 1998