連続的なシミュレーション環境での エージェントの学習と意思決定の実装について

奈良女子大学理学部 情報科学科 4 回生
新出研究室 林　果穂

概 要

自律エージェント及びロボットに関する研究において環境は、離散的な状態空間を用いた仮想世界上のものとして取り扱ってきた。しかし、行為が実世界で成功するためには、離散的な環境を想定するだけでは不十分であり、連続した状態空間上で環境の多様な変化に柔軟に対応できることを要する。

濱田・久妻による研究では例題としてカヌーレーシングを選び、連続世界に基づくシミュレーションを作成した。この研究では、エージェントは自律的に行為を選択することができなかった。そこで本研究では、エージェントが漕ぎ方を学習し、最適な行動を選択する機能を追加した。

1　はじめに

BDI エージェントとは、信念 (B)、願望 (D)、意図 (I) という 3 つの心的状態パラメータをもつことで人間の思考をモデル化し、熟考しながら自律的に行動選択を行うものである。これを用いることにより、エージェントは矛盾のない一貫した行動をとることが可能となる。

我々は、自律型エージェントの実装を目指すため、BDI エージェントを用いた研究を行ってきた [1]。従来、BDI エージェントを用いた研究において環境は、離散的な状態空間を用いた仮想世界上のものとして取り扱ってきた。そのため、エージェントの行動は、例えばグリッドワールド上の決められたマスを動くといった制限された行動に限られていた。しかし、実世界では、決められたマス目のみ上での行動では対応できない。連続した状態空間上で、環境の多様な変化に合理かつ柔軟に対応する必要がある。

従来研究 [2] [3] では、実世界におけるエージェント構築に向けて、カヌーレーシングをテストベッドとしたシミュレーション環境の作成を行った。これにより、エージェントはあらかじめ記述されたプラン通りに行為を実行し川を下ることが可能になった。しかし、エージェントが実世界で課題を遂行するためには、決められたプランに従って行動するだけでは、目標達成には不十分である。実世界は常に動的に変化し、かつ極めて複雑なシステムであるため、プランを選択して行動を決定するのみならず、反射的な行動スキルを獲得して利用することも必要となる。また、エージェントが反射的行動を獲得するためには、環境を知覚しなければならない。よって、本研究では従来のシミュレーション環境を、実世界のエージェント構築に向けてより有用なものとするため、エージェントの知覚と、エージェントがその知覚を用いて反射的行動を獲得するための学習機構の追加を行った。

本研究は、奈良女子大学情報科学科 4 回生亀村との共同研究である。本論文では、カヌーシミュレーションの環境設計と強化学習について述べる。行為の選択については [4] に述べるので本論文では割愛する。

2　カヌーシミュレーション実装方針

2.1　問題点とその解決

従来研究では、エージェントはあらかじめ記述されたプラン通りにしか行為を行うことができず、環境を知覚することもできなかった。エージェントが学習を行う機能もなかった。エージェントは川を下るまで行為を実行し続け、その途中でエージェントが岸に衝突した場合は、エージェントを同じ y 座標上の近くの岸に移動させ、再度その地点から出発するようになっていた。そこで本研究では、エージェントが環境を知覚し、その情報をもとに行為を学習し、選択できるような機能の追加を行う。そして、エージェントが岸に衝突しないようにカヌーを操り、川を下る能力を身につけられることを示す。

2.2　簡略化

本研究のシミュレーション環境は、実世界でのエージェント構築に向けての検討のためのものではあるが、実世界は複雑であるため、その全てを最初からシミュレータに盛り込むことは困難である。そこで実世界の川下りに対する簡略化を行ったモデルを用いる。以下に、本研究で行った簡略化について述べる。

- エージェントが使う基本行為を、何もせず川の流れに任せる、前進する、後進する、右に進む、左に進む、の５種類とする。
- エージェントは信念として自身の位置の情報をもつ。川の流れなどに関する信念の保持は省く。
- 連続世界で川の流れをそのまま表現すると学習を行うには無限の状態を持たなくてはならないため、持つべき情報が無限になる。そこで、川を一定区間毎に区切り、それぞれの区切り毎に流れの向きと速さの情報、学習結果をもつこととする。なお今回は 8×6 マスに区切ったが、これは変更可能である。精度よく学習できるように問題の規模によって決めなければならない。

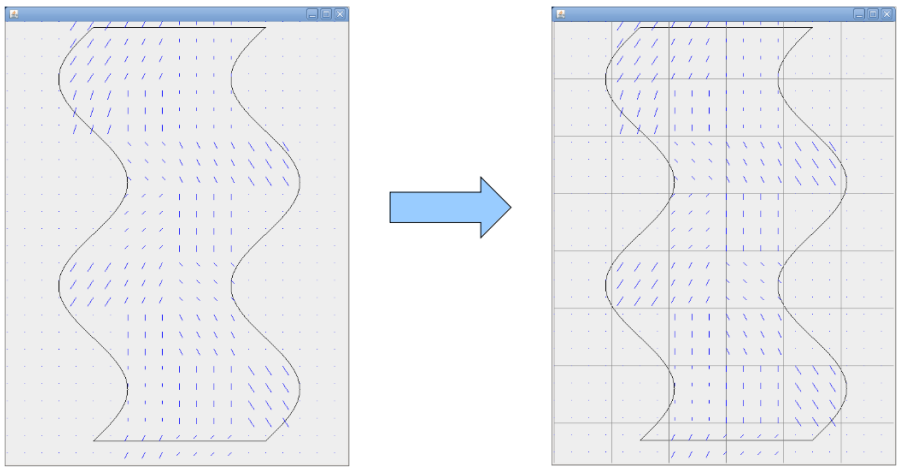

図 1: 川をマス目に区切る

3　プログラム構造

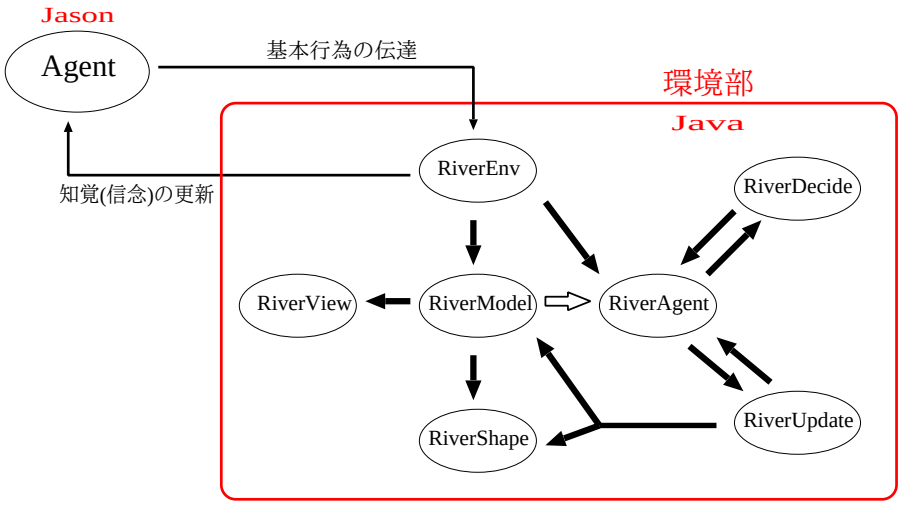

図 2: プログラムの構造

作成において、エージェントの記述には BDI エージェント構築のためのプラットホームである Jason[5] を、環境の設定には Java を用いた。プログラム構造は図 2 のようになっている。図の赤枠内は環境部であり、この部分の設計について以下に述べる。

3.1　クラス設計

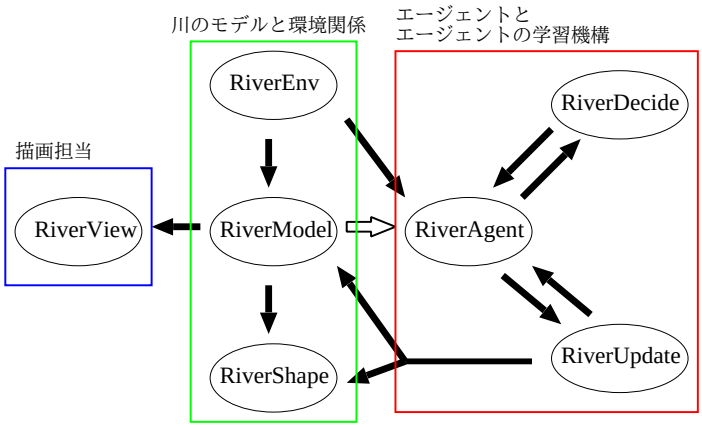

図 3: クラス設計

クラス設計は図 3 のようになっている。黒い矢印は作業依頼、白い矢印は一方が他方を持つ関係を示している。赤い四角で囲まれた 3 つのクラスはエージェントとエージェントの学習機構、緑の四角で囲まれた 3 つのクラスは川のモデルと環境関係、青い四角で囲まれたクラスは描画担当である。

3.2　クラスの概要

RiverView は従来研究のプログラムを引き継ぎ、RiverEnv, RiverModel, RiverShape, RiverAgent は一部のメソッドを改良、追加した。RiverUpdate, RiverDecide は新しく作成したクラスである。ここでは各クラスの役割について述べる。

- RiverEnv
  Jason が提供する Environment クラスのサブクラスである。Jason とやりとりをするため、エージェントの基本行為の伝達に関するメソッドを持つ。また、Environment の executeAction をオーバーライドすることで自前の基本行為を実現している。基本行為を Jason 側から受け取り、RiverModel に伝達して処理する。

- RiverModel
  伝達された基本行為の処理を行うクラスである。基本行為を受け取ると、RiverAgent からエージェントの位置を取得し、RiverShape からその位置の川の流速を受け取って、エージェントの速度と流速の合力から次の地点を計算する。そして、新しいエージェントの位置を RiverAgent に渡し、RiverView へエージェントの描画を依頼する。最後に、RiverEnv に基本行為の実行結果を返す。

- RiverShape
  川の形状、流速、各地点の位置情報を持つクラスである。川の形状を返すメソッド、与えられた座標地点の川の流速を返すメソッド、エージェントの位置が川の中であるかどうかを判定するメソッドを持つ。

- RiverView
  川やエージェントの描画を担当するクラスである。エージェントの位置情報が更新される度、描画を行う。エージェントは図 4 のように三角形で表す。また、エージェントの向きを示すため赤い印をつけた。川の流速は青い線で書く。青い線の傾きは流速の向きを表し、長さが長いほど流れが速いことを表す。

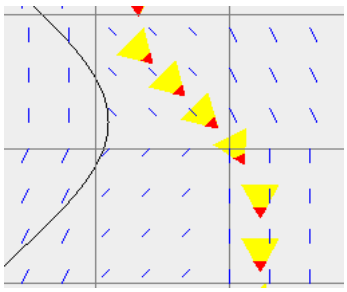

図 4: 描画

- RiverAgent
  エージェントの位置情報、学習結果をもつクラスである。

- RiverUpdate
  状態、行為及び知覚を受け取って漕ぎ方の学習を行うクラスである。学習中の基本行為の処理は RiverModel に依頼し、RiverShape から川の外に出てしまった、ゴー

ル位置に到達したなどの位置情報を取得する。学習結果は RiverAgent に渡す。学習方式については４章で詳しく述べる。

- RiverDecide
  RiverAgent からエージェントの位置と学習の結果を受け取って、最適な行為を選択するクラスである。行為選択の詳細については [4] で述べる。

3.3 クラスの変更点

ここでは従来研究のものから変更したクラスについて、その変更点について述べる。

- RiverEnv
  エージェントが信念として自身の位置の情報を持てるよう、信念の更新を行うメソッドを追加した。RiverModel から基本行為の実行結果を受け取り、エージェントが川の中にいるかを判定し、知覚の更新を行う。

- RiverModel
  川の流れの処理と、基本行為の処理が別々のメソッドで行われていたのを統合した。また、学習中は RiverView に描画依頼を行わないようにした。

- RiverShape
  エージェントが岸にぶつかった時に軌道修正するメソッドを、ぶつかったことを知らせるメソッドに変更した。エージェントがゴール位置に到達したことを知らせるメソッドを追加した。また、学習前と学習後の違いが明確になるように、一部川の流れを変更した。

- RiverAgent
  学習結果を持つための配列を用意した。RiverUpdate に任意の回数の学習を依頼するメソッド、RiverDecide に学習結果から最適な行為の選択を依頼するメソッドを追加した。

4　強化学習

強化学習とは、『「どの選択肢を選ぶのが正解か」を教えてくれる存在はいないが、実際に試すことによって何らかの報酬が得られる』場合に、「試してみることによって各選択肢の価値を推定する」ことと「最も推定価値の高い選択肢を得る」ことのバランスを取ることによって、うまい行動を見つけ出そうという方式である。本研究では、状態遷移のある問題での強化学習でよく用いられる学習方式の 1 つである Q 学習を用いた。

4.1　問題設定

学習機能を実現するために、サンプルとしてエージェントにできるだけ速く川を下る行為を学習させる。ただし、途中で岸に衝突してしまうと失敗となる。エージェントがスタート地点を出発してから、岸にぶつかって止まる、またはゴール地点に到達するまでを 1 つの「エピソード」と呼ぶ。このとき、エージェントのいる位置によって 5 つの基本行為のうちどれをとればよいかが変わってくる。そこで、川を一定区間毎に区切ったマスを「状態」と捉え、そのマス毎に「どの基本行為を取ればよいか」を学習する。

ここで、ある状態でのある 1 つの行動に対する報酬を以下のように設定する。

- その行動で岸に衝突したら報酬 -100
- その行動でゴール地点についたら報酬 100
- それ以外なら行動 1 回あたり報酬 -1

行動 1 回ごとの報酬を -1 とするのは、エージェントができるだけ速くゴールに着く行動を学習できるようにするためである。その上で、各状態において「その状態からエピソード終了までの通算報酬」ができるだけ大きくなるように行動すればよい。

4.2　学習方式

エピソード終了までの通算報酬を最大化するにも、通算報酬の真の値はエピソード終了までわからない。そこで、「ある状態においてある行動を取ったときの、その行動からエピソード終了までの通算報酬」の平均値を推定して保持し、この値を学習によって次第に更新していくことにより、より良い行動を取れるようにする。この平均値をその状態 s でのその行動 a の「状態行動価値」と呼び、$Q(s,a)$ と書く。また、行動 a で直接得られた報酬を r とし、行動 a の結果として状態 s から状態 s' に移ったとする。状態 s' からエピソード終了までの通算報酬の和を「状態 s' において、状態行動価値の推定値が最もよい行動を a' としたときの、$Q(s',a')$ の現時点での推定値」を s' の状態価値と呼び、$Q(s')$ と書いて近似する。

$$
\begin{aligned}
Q(s,a)\text{ の新たな推定値} \quad = \quad & Q(s,a)\text{ の現在の推定値} \\
& + c \times (r + \alpha \times Q(s')\text{ の現在の推定値} - Q(s,a)\text{ の現在の推定値})
\end{aligned}
$$

この式によって、$Q(s,a)$ の推定値を更新していく。新たな推定値にこの式を使うものを Q 学習 [6] という。

ここで、c は学習係数であり、今回は 0.1 に設定した。α は 0 以上 1 未満の定数とし、割引率という。割引率の導入はエピソードが無限に続く場合にエピソード終了までの通算

報酬が無限大になってしまうことを抑える役割をする。また、エピソードが有限の場合でも、割引率の導入により、即時報酬を未来の報酬よりやや重視する効果を持たせることができる。

学習は多数回のエピソードを繰り返し、1 つ 1 つのエピソードについて、毎回の学習を ε-greedy 法で選んで実行し、行動のたびに上記を実施して状態行動価値の推定値を更新していくことによって行う。

4.3 学習過程

今回は 5000 回エピソードを繰り返し、学習を行った。学習中、コンソール画面に通算報酬の平均を出力すると図 5 のようになった。

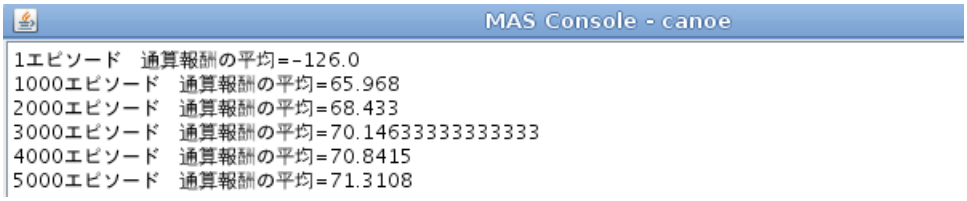

図 5: コンソール画面

通算報酬の平均が最初は負数だが、エピソードを繰り返していくと上昇し、70 程度に落ち着くことがわかる。平均が 70 程度より高くならないのは、行動選択に ε-greedy を使っており、岸にぶつかるような事態がときどき起きてしまうためである。

5　学習前後の実行結果

学習前と学習後の実行結果の比較のため、それぞれのコンソール画面を図 6,7 に示す。平均報酬と、学習後にエージェントが川を下るときにどの行為を使ったかが出力されている。

図 6: コンソール画面 (学習前)

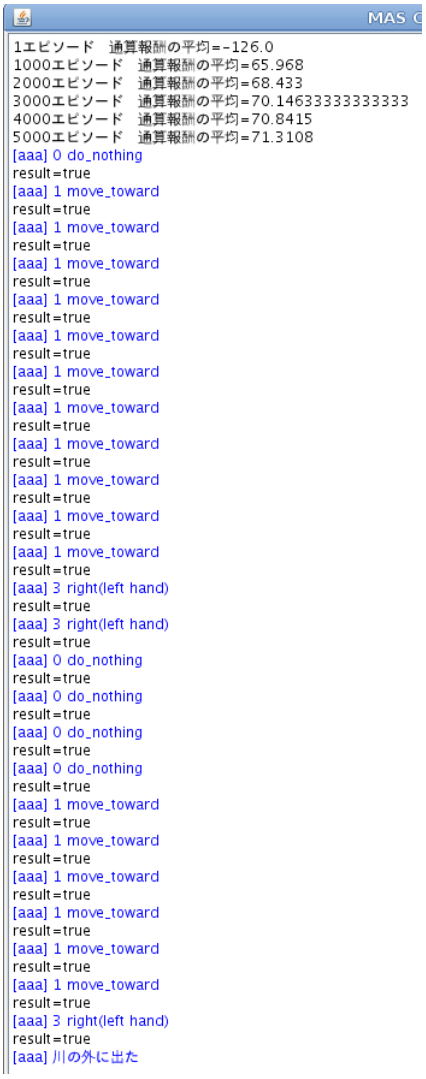

図 7: コンソール画面 (5000 回学習後)

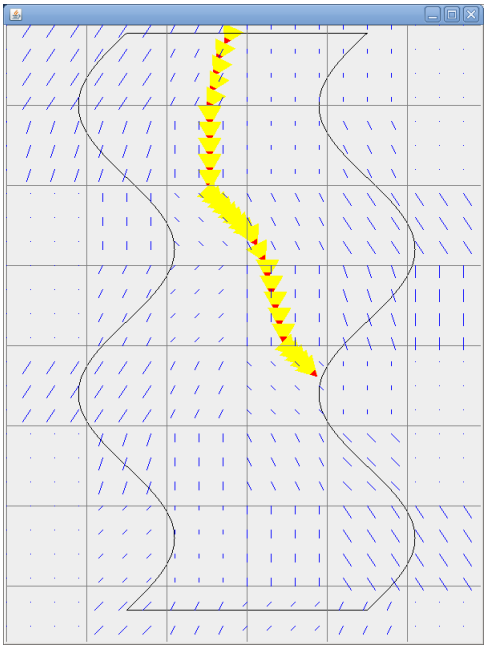

図 8: 学習前 (1 エピソード目)

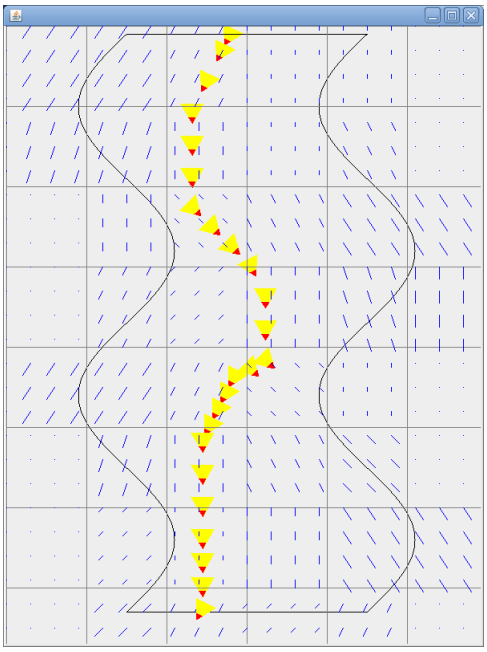

図 9: 5000 回学習後

図 8 は学習前、図 9 は学習後のエージェントが川を下る様子である。学習前はエージェントは途中で岸に衝突してしまうが、学習後は岸に衝突することなく、学習前より速く川を下ることができている。コンソール画面を比べても学習後の方が行動回数が少ないことがわかる。

6　まとめ

連続世界に基づいたシミュレータ環境に知覚と学習機構を追加した。これにより、エージェントは環境を知覚し、最適な行為の選択を学習できるようになった。また、学習中の川を下る様子は描画せず、学習後に学習結果を使ってエージェントが川を下る様子を描画するようにした。

現段階では、川の流れは一定で変化しない環境で、エージェントの位置情報と川の水流を知覚して学習を行っている。しかし、水流は知覚できるだけで、学習、行動選択で利用されているのは位置情報のみである。実際は時間変化に伴い水流も変化するので、位置情報をもとにした学習だけでは対応できない。状態が変化することも考慮した学習が必要である。

また、シミュレーション環境がカヌーレーシングという特定の例題にできるだけ依存しないように改善する必要もある。

さらに、学習によって得た行動を、BDI モデルでのプラン作成に利用できるようにすることが今後の課題である。

7　謝辞

本論文の執筆及び研究にあたり、丁寧かつ親切にご指導下さった指導教官の新出尚之准教授に深く感謝致します。また、新出研究室の皆様に感謝致します。ありがとうございました。

参考文献

[1] 高田司郎, 新出尚之, 濱砂幸裕, 波部斉, 藤田恵. アトラクター状態を用いた実世界における基本行為の学習について. 情報処理学会研究報告 2013-MPS-92, No. 24, pp. 1–6, 2013.

[2] 濱田百合. 連続世界におけるエージェントの学習と意思決定に向けて. 奈良女子大学情報科学科 2012 年度卒業論文, 2013.

[3] 久妻さゆり. 実世界におけるエージェント構築に向けたシミュレーション環境の作成. 奈良女子大学情報科学科 2012 年度卒業論文, 2013.

[4] 亀村美佳. エージェントの知覚と学習による行動決定の実現. 奈良女子大学情報科学科 2013 年度卒業論文, 2014.

[5] Rafeal H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons, 2007.

[6] 三上貞芳, 皆川雅章. 強化学習, pp. 159-161. 森北出版, 2000.