CS 知的システム演習

強化学習による ライントレーサーのプログラム

松吉 俊

強化学習を利用する手順

(1) 状態を定義する
(2) 報酬関数を定義する
(3) Q 学習のアルゴリズムを適用する
(4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

強化学習を利用する手順

(1) 状態を定義する
(2) 報酬関数を定義する
(3) Q 学習のアルゴリズムを適用する
(4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

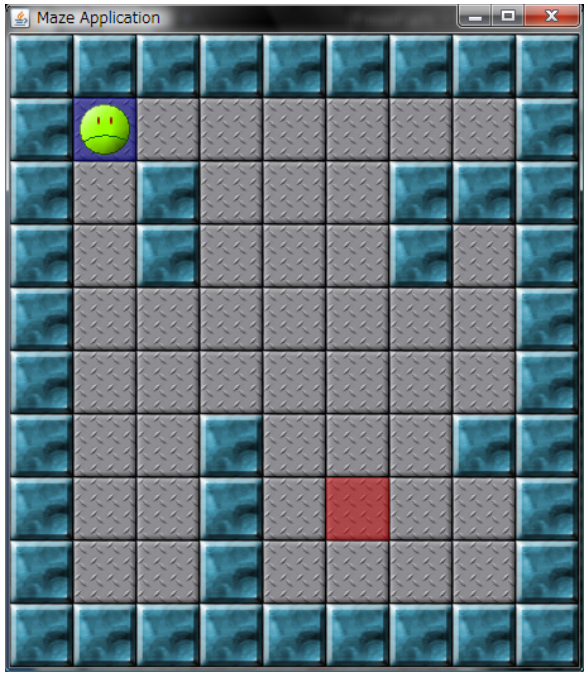
状態と行動を定義する

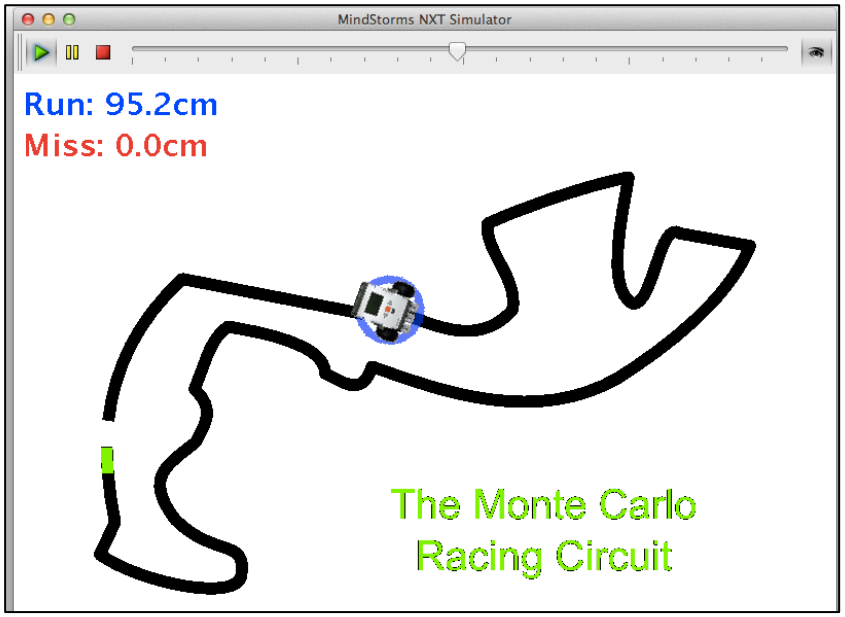
- 状態:
  - 光センサーの値の組み合わせを状態とすると良い
    - 値: WHITE or BLACK
    - センサーは3つあるので、2 x 2 x 2 = 8つの状態
- 行動:
  - 雛形プログラムの場合、次の2種類
    - 10度右回転＋1cm前進
    - 10度左回転＋1cm前進
  - 有効な行動を考案し、利用するとよい

できることとできないこと

センサー入力: (1) ラインに乗っているかどうか
(2) ゴール位置にあるかどうか

現在位置 (座標) の情報を取得できない





非マルコフ問題

一般に、エージェントが環境のすべてを観測することは
困難であり、部分的にしか観測できない

非マルコフ決定過程

タスクが非マルコフ決定過程だからといって
強化学習が適用できないわけではない

非マルコフ問題の解法

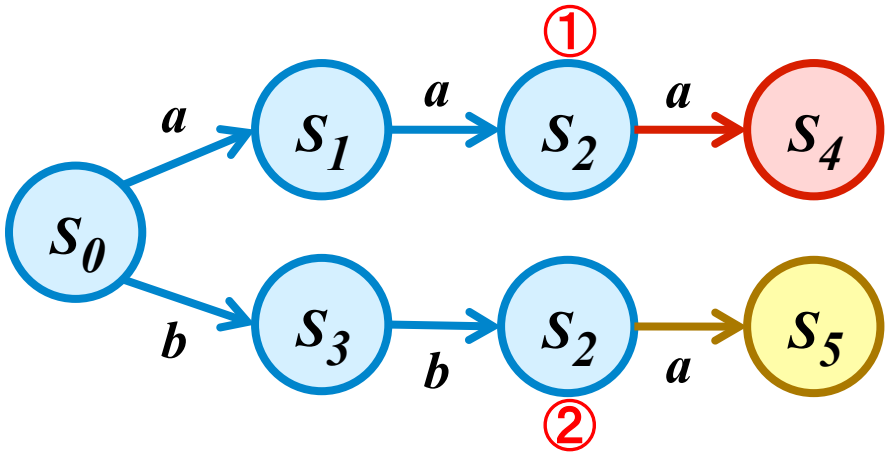
- 解法1： 気にしない

  観測の不完全性が小さければ、強化学習をそのまま適用しても、うまくいく場合がある

- 解法2： 有限長の過去の履歴を状態に追加する

  過去の状態と行動の履歴を現在の状態の一部として考える

例：2ステップ前までの履歴を考慮する場合



①の状態 s_2 には、$<(s_0, a), (s_1, a)>$ という履歴を状態の一部として追加

②の状態 s_2 には、$<(s_0, b), (s_3, b)>$ という履歴を状態の一部として追加

2つの状態 s_2 が区別可能になる

強化学習を利用する手順

(1) 状態を定義する
(2) 報酬関数を定義する
(3) Q 学習のアルゴリズムを適用する
(4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

報酬関数を定義する

- 報酬の例:
  - ゴール: プラスの値
  - ライン上: プラスの値
    - ゴールのみに報酬を与えると、
      ラインをトレースしてくれない
  - ライン外: マイナスの値

強化学習を利用する手順

(1) 状態を定義する
(2) 報酬関数を定義する
(3) Q 学習のアルゴリズムを適用する
(4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

Q 学習のアルゴリズム

(0) Qテーブル (状態数×行動数の2次元表) を作る

(1) 全ての状態 s と行動 a に対して、$Q(s,a)$ の値を 0 で初期化

(2) 現在の状態が s であるとき、ε-greedy 戦略で行動 a を選択し、それを実行する

(3) 遷移先の状態 s' を観測し、$Q(s,a)$ の値を次式で更新する

$$Q(s,a) \leftarrow Q(s,a) + \alpha[(r(s') + \gamma \max_{a'} Q(s',a')) - Q(s,a)]$$

(4) s' を s としてステップ **(2)** へ戻る

(2)-(4)を
サイクルさせる

MyRobot.java

```
int trials = 500;     // 強化学習の試行回数
int steps  = 100;     // 1試行あたりの最大ステップ数

for(int t=1; t <= trials; t++) {      // 試行回数だけ繰り返し

  /* ロボットを初期位置に戻す */

  for (int s=0; s < steps; s++) {     // ステップ数だけ繰り返し

    /* ε-Greedy 法により行動を選択 */

    /* 選択した行動を実行 (ロボットを移動する) */

    /* 新しい状態を観測＆報酬を得る */

    /* Q 値を更新 */

    /* もし時間差分誤差が十分小さくなれば終了 */

  }
}
```

MyRobot.java

```
int trials = 500;     // 強化学習の試行回数
int steps  = 100;     // 1試行あたりの最大ステップ数

for(int t=1; t <= trials; t++) {      // 試行回数だけ繰り返し

  /* ロボットを初期位置に戻す */

  for

    /

    /* 選択した行動を実行（ロボットを移動する） */

    /* 新しい状態を観測＆報酬を得る */

    /* Q 値を更新 */

    /* もし時間差分誤差が十分小さくなれば終了 */

  }
}
```

強化学習により迷路を解くプログラムと同様!

クラス QLearning

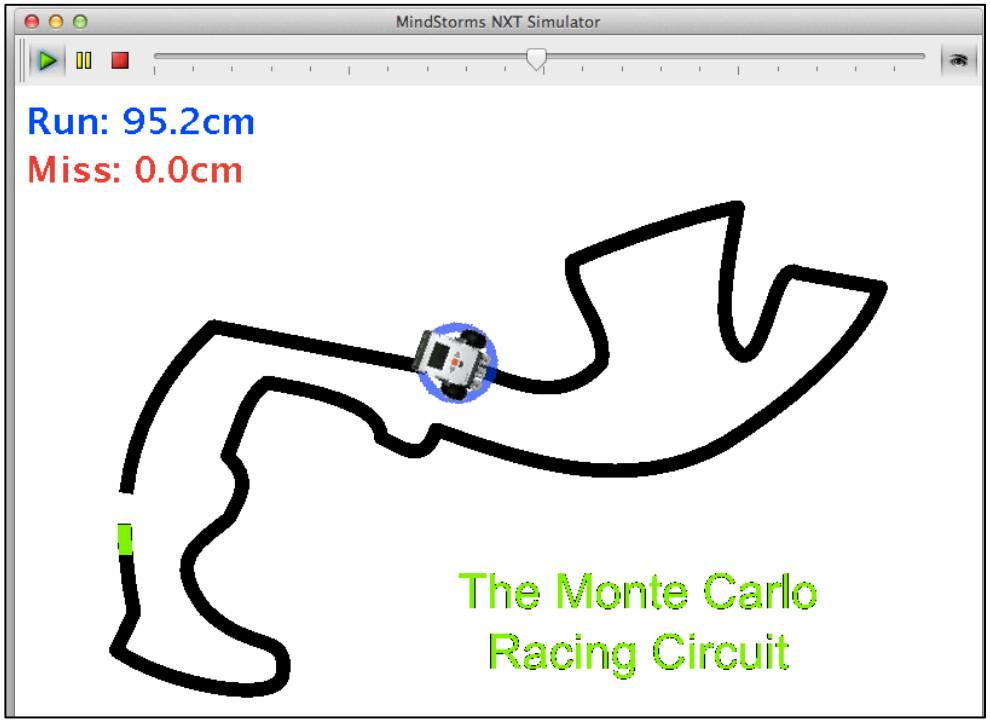
- 必要なメソッドは、強化学習により迷路を解くプログラムと同様
  - **この前作成したQLearning.javaが再利用できる!**
- 必要に応じて、Qテーブルを表示するメソッドも実装して利用する

強化学習を利用する手順

(1) 状態を定義する
(2) 報酬関数を定義する
(3) Q 学習のアルゴリズムを適用する
(4) 十分な学習を行ったのちに最適政策が得られる（はずな）ので、それに従って行動する
   - greedy法

演習5

- Q学習によるライントレーサーのプログラムを実装せよ



第4回の出席確認

- 実装できたところまでで良いので、
  演習5に対するプログラムをMoodle上で
  提出する
  - tar.gz形式のファイルを提出する
  - 最終評定には直接関係しません

- Special thanks:
  - 山本 泰生先生
  - 鍋島 英知先生