CS 知的システム演習

学習結果の利用法

松吉 俊

実機で強化学習をするのか？

強化学習では多数の試行が必要

実機上で強化学習を実行することは困難

学習結果のみ、実機に移植する

強化学習の学習結果

行動

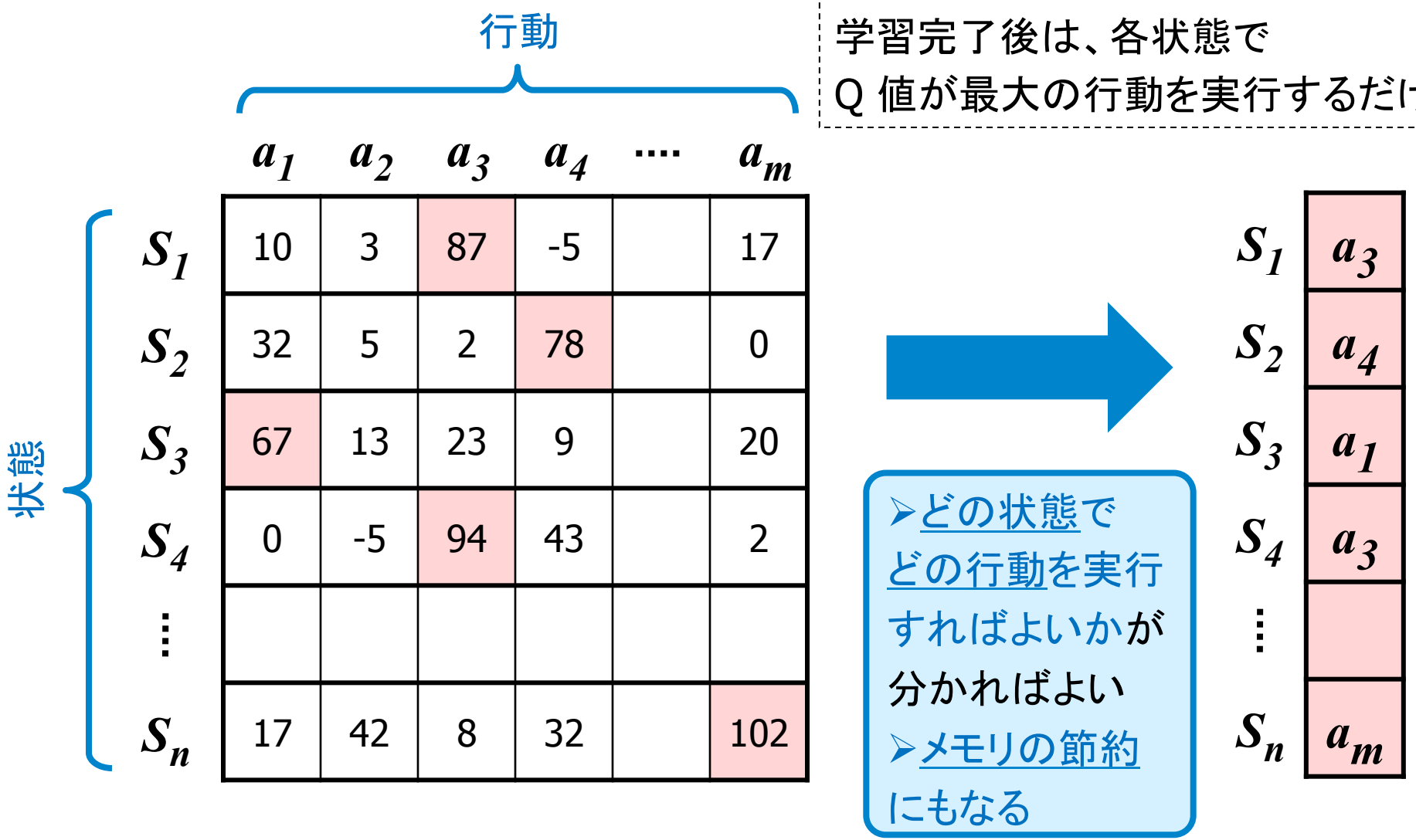
状態

	a_1	a_2	a_3	a_4	….	a_m
S_1	10	3	87	-5		17
S_2	32	5	2	78		0
S_3	67	13	23	9		20
S_4	0	-5	94	43		2
⋮						
S_n	17	42	8	32		102

学習完了後の
Q テーブルのみ
実機に移植すればよい

Qテーブル全体を保持する必要はない

学習完了後は、各状態で
Q 値が最大の行動を実行するだけ



MyRobotForNXT.javaを作る

1. 強化学習によるライントレーサーのプログラム `MyRobot.java`を完成させる
2. 学習完了後のQテーブルを画面に出力するように、このプログラムを改良する
3. シミュレーター上でプログラムを実行する
4. 実機NXT用の新しいクラス`MyRobotForNXT.java` を作成する
   - 2.で出力したQテーブル(からの抜粋)をここに移植する
5. シミュレーター上で実機NXT用のクラスの動作を確認する

```
% java Simulator MyRobotForNXT map1-rect.png
```

学習完了後のQテーブルを出力

- 学習が完了したら、すぐに出力してもよい
- しかしながら、うまく学習できたかどうかは学習結果による動作を確認しないと分からない

- 動作が確認でき、isOnGoal()メソッドがtrueを返す時点で、Qテーブルを出力するとよい

出力例:

```
q[0] = 2;
q[1] = 1;
     :
q[7] = 0;
```

簡略化したQテーブルの配列

状態の番号

行動の番号

MyRobotForNXT.java

```
public class MyRobotForNXT extends Robot {

  /** leJOS での起動用 main 関数 */
  static void main(String[] args)
  {
     try {
       // 時間計測
       Long time = System.currentTimeMillis();
       // ロボットオブジェクトを生成して実行
       new MyRobotForNXT().run();
       time = (System.currentTimeMillis() - time) / 1000;
       System.out.println("Time = "+time.intValue() + "sec");
       // 7秒待ってから停止
       Thread.sleep(7000);
      }catch (InterruptedException e) {
            ;
      }
  }
```

(次ページに続く)

MyRobotForNXT.java

```
(前ページからの続き)
  /** 実行用関数 */
  public void run() throws InterruptedException {
    /* 学習した最適政策を表す配列 */
    int[] q = new int[8];
    q[0] = 2;
    q[1] = 1;
    ...
    q[7] = 0;
    while (true) {
      /* 現在の状態を観測 */
      /* その状態における最適な行動を実行 */
      if (isOnGoal()) break;  // ゴールに到達すれば終了
    }
  }
}
```

各状態に対して最適な
行動番号を登録する
(学習後のQテーブルから)

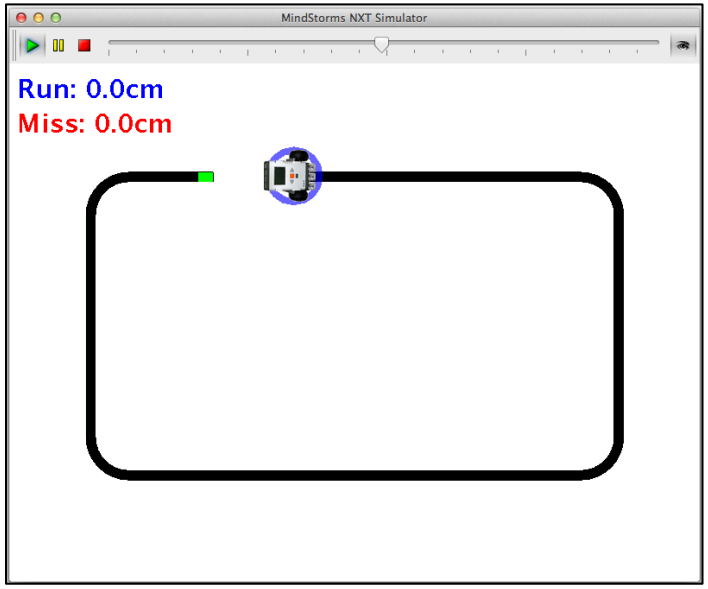
最適政策

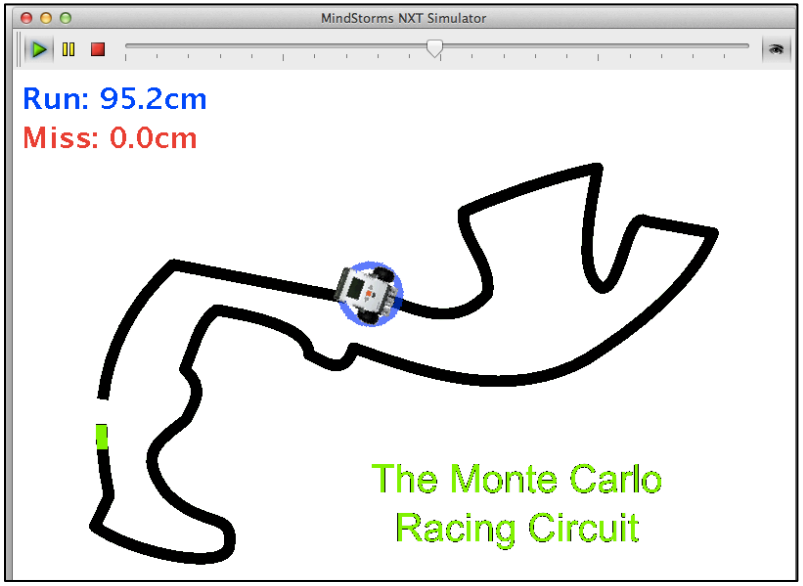
- 学習する度に変わる可能性がある
  - 何度も試行していると、
    さらに良いものが得られることがある

- 一般に、mapにより異なる
  - それゆえに、一般に、MyRobotForNXT.javaは、
    mapの数だけ、異なるものが出来る

演習6

- map1とmap6に対して、それぞれ、MyRobotForNXT.javaを作成せよ





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