CS 知的システム演習

強化学習で迷路を解くプログラム

松吉 俊

強化学習版迷路を解くプログラム

- 雛形のQLearning.java を提供

1. QLearning.javaを完成させる
2. QLearningをもとにMazeModel.javaを修正する
   - 強化学習を組み込んだプログラムを作成

クラス QLearning

- コンストラクタ
- ε-Greedy法により選択した行動を返すメソッド
  - Q学習の最中に利用
- Greedy法により選択した行動を返すメソッド
  - Q学習完了後に利用
- Qテーブルを更新するメソッド

クラス QLearning の製作 (1)

- コンストラクタ

```
/**
 * Q学習を行うオブジェクトを生成する
 *@param states  状態数
 *@param actions 行動数
 *@param alpha   学習率(0.0～1.0)
 *@param gamma   割引率(0.0～1.0)
 */
QLearning(int states, int actions, double alpha, double gamma)
```

クラス QLearning の製作 (2)

- ε –Greedy 法により行動を選択する

```
/**
 * ε-Greedy 法により行動を選択する
 *@param state   現在の状態
 *@param epsilon ランダムに行動を選択する確率(0.0～1.0)
 *@return 選択された行動番号
 */
int selectAction(int state, double epsilon)
```

- Greedy 法により行動を選択する

```
/**
 * Greedy 法により行動を選択する
 *@param state   現在の状態
 *@return 選択された行動番号
 */
int selectAction(int state)
```

クラス QLearning の製作 (3)

- Q値を更新する

```
/**
 * Q値を更新する
 *@param before 状態
 *@param action 行動
 *@param after  遷移後の状態
 *@param reward 報酬
 */
void update(int before, int action, int after, double reward)
```

時間差分方程式の復習

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

- 学習率αと割引率γを与える (ここではα=1、γ=1)
- 状態S_1、行動a_1、遷移後の状態S_2、報酬100だった場合

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

$$Q(S_1,a_1) \leftarrow$$

$$Q(S_1,a_1) + (100 + \max_{a'} Q(S_2,a')) - Q(S_1,a_1)$$

$$= 10 + (100 + 78) - 10$$

$$= 178$$

時間差分方程式の復習

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

- 学習率 α と割引率 γ を与える (ここでは $\alpha = 1$、$\gamma = 1$)
- 状態S_2、行動a_2、遷移後の状態S_3、報酬 -20だった場合

	a_1	a_2	a_3	a_4
S_1	178	3	87	-5
S_2	32	5	2	78
S_3	67	13	23	9
S_4	0	-5	94	43
⋮				
S_n	17	42	8	32

$$Q(S_2,a_2) \leftarrow$$

$$Q(S_2,a_2) + (-20 + \max_{a'} Q(S_3,a')) - Q(S_2,a_2)$$

$$= 5 + (-20 + 67) - 5$$

$$= 47$$

時間差分方程式の復習

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

- 学習率 α と割引率 γ を与える (ここでは $\alpha = 1$、$\gamma = 1$)
- 状態S_2、行動a_2、遷移後の状態S_3、報酬 -20だった場合

	a_1	a_2	a_3	a_4
S_1	178	3	87	-5
S_2	32	47	2	78
S_3	67	13	23	9
S_4	0	-5	94	43
⋮				
S_n	17	42	8	32

$$Q(S_2,a_2) \leftarrow$$

$$Q(S_2,a_2) + (-20 + \max_{a'} Q(S_3,a')) - Q(S_2,a_2)$$

$$= 5 + (-20 + 67) - 5$$

$$= 47$$

MazeModelのrunメソッド

```
/**
 * 実行用関数
 */
public void run()
{
        try {
                // step 1: Q学習する
                // step 2: 学習したQテーブルの最適政策に基づいて
                //         スタート位置からゴール位置まで移動
        }
        catch (Exception e) {
                e.printStackTrace();
                System.exit(-1);
        }
}
```

MazeModelのrunメソッド: Q学習

```
/**
 * 実行用関数
 */
public void run()
{
        try {
          // step 1: Q学習する
          // QLearningのインスタンスを作る
          int states = 10; //状態数
          int actions = 10; //行動数
          double alpha = 0.5; //学習率
          double gamma = 0.5; //割引率
          QLearning ql = new QLearning(states, actions, alpha, gamma);

}
```

適切に値を決める

MazeModelのrunメソッド: Q学習

```
try{･･･
    QLearning ql = new QLearning(states, actions, alpha, gamma);

    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 */
       }
    }
}
```

MazeModelのrunメソッド: Q学習

```
try{･･･
    QLearning ql = new QLearning(states, actions, alpha, gamma);

    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

      /* ε-Greedy 法により行動

      /* 選択した行動を実行（ロ

      /* 新しい状態を観測＆報酬

      /* Q 値を更新 */

      /* もし時間差分誤差が十分小さくなれば終了 */
       }
    }
}
```

```
//ロボットを初期位置に戻す
robot.setX(mazeData.getSX());
robot.setY(mazeData.getSY());
```

MazeModelのrunメソッド: Q学習

```
try{･･･
   QLearning ql = new QLearning(states, actions, alpha, gamma);

   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

     /* 実

     /* Q

     /* 状
     }
   }
}
```

```
//現在の状態番号を取得する
int state ･･･ //頑張って作る
double epsilon = 0.5; //ランダムに行動を選択する確率

//qlインスタンスから呼び出す
int action = ql.selectAction(state, epsilon);
```

MazeModelのrunメソッド: Q学習

```
try{･･･
   QLearning ql = new QLearning(states, actions, alpha, gamma);

   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

      /* Q 値を更新 */

      /* もし時間差分誤差が十
       }
   }
}
```

```
//次の状態番号
int after //頑張って取得する
//状態afterにおける報酬
int  reward // 頑張って取得する
//qlインスタンスから呼び出す
update(state, action, after, reward)
```

第2回の出席確認

- 実装できたところまでで良いので、Q学習で迷路を解くプログラムをMoodle上で提出する
  - tar.gz形式のファイルを提出する
  - ここに提出されたものは、最終評定には直接関係しません

課題

- Q学習により迷路を解くプログラムを完成させ、関連ファイル一式をMoodle上で提出する
  - tar.gz形式のファイルを提出する
  - 必ずたくさんのコメントを書く
  - プログラムの動作だけでなく、コメントの量と書き方も評価します
- 上記に加え、以下を記述したレポートもMoodle上で提出する (PDF形式)
  - 「状態」と「報酬」の説明
  - 工夫した点と苦労した点
  - レポートの先頭に、学籍番号と名前を書く

- Special thanks:
  - 山本 泰生先生
  - 鍋島 英知先生
  - 岩沼 宏治先生