卒業研究報告書

題 目

# パックマンのモンスター追跡 AI

指 導 教 員 石 水　隆 講師

報 告 者

18–1–037–0062

高 木 亮 多

近畿大学理工学部情報学科

平成 5 年 2 月 2 日提出

# 概要

パックマンは，パックマンと呼ばれる自キャラを操作し迷路内をモンスターから逃げながら迷路に散らばるドットを全て食べ尽くすことを目指すゲームである．本研究ではパックマンを単純化させたゲームにおいて，機械学習を用いてパックマン (以下逃走者とする) ができるだけ長時間モンスター (以下追跡者とする) から逃げる逃走者 AI および追跡者が，逃走者をできるだけ早く捕まえる AI を開発する．本研究では Open AI Gym [6, 7] のインターフェイス形式に則って環境を作成してその環境を使い，機械学習をしていく．

環境は 5 つのマップからランダムに選ばれたマップに逃走者が一人，追跡者が二人がそれぞれ迷路内にランダムで配置されて，それぞれ 1 ターンに 1 マス動き，追跡者が逃走者を捕らえたり一定ターン終了するとそのエピソードが終了する．

まずランダムに行動する逃走者に対して用意した 5 つのマップで 500,000 エピソード学習した追跡者を用意し，その追跡者を利用して逃走者をまた 500,000 エピソード学習させていく，その逃走者とランダムに行動する逃走者，単純に距離を取ろうとする逃走者の三者を使い 500,000 エピソード学習させていく．

それを使い今度は学習用マップとは別のテスト用マップを 3 つ用意して先ほど述べた三者の逃走者に対してそれぞれ 1000 回づつテストする．

その結果テストマップではあまり良い結果は残せなかったが，学習マップではまずまずの結果が出ていた，その結果を基に過学習が起こっていると考えられたが，新たに 5 ターンに一度逃走者がストップするルールを追加した上で学習回数を減らしてテストしても結果の改善にはつながらなかった．

# 目次

# 1 序論

## 1.1 本研究の背景

パックマン [1] はナムコ (現バンダイナムコエンターテインメント) から発表されたゲームである．ゲームのプレイヤーはパックマンと呼ばれる自キャラを操作し青い壁で囲まれた迷路で敵であるモンスターから逃げながら迷路に散らばるドットを全て食べ尽くすことを目指す．

## 1.2 パックマンとは

パックマン [1] はナムコ (現バンダイナムコエンターテイメント) より 1980 年に発売されたアーケードゲームである．プレイヤーの操作キャラクターであるパックマンは迷路で敵である 4 体モンスターから逃げながら迷路に散らばるドットを全て食べ尽くすゲームである．迷路内に配置された全てのドットを食べ尽くすとその面はクリアとなり次の面に進む．パックマンがモンスターに触れられると，モンスターに食べられてしまいミスとなる．ドットのほかパワーエサも配置されておりそれを食べるとマップのモンスターたちが一定時間イジケ状態となり，その状態でパックマンがモンスターに触れると逆にパックマンがモンスターを食べることができる．パックマンがドットおよびパワーエサを食べると得点が入る．また，パワーエサを食べたときにモンスターを食べると大きな得点が得られる．他にもボーナス得点の入るフルーツも存在する．

## 1.3 パックマンに関する既知の結果

本節ではパックマンに関する既知の結果について述べる．[8] では Ms.Pac-Man モンテカルロ木探索を利用してパックマンの挟み打ち問題について解決を試みている．[9] では同じく逃走者についての研究がされている，この研究ではモンテカルロ木での探索でゴースト同士の協力制御について述べられている．[10] ではサポートベクトルマシンを利用した教師あり学習を使いスコアの向上を試みた．[11] ではニューラルネットワークを使い GameGAN を構築してゲームエンジン抜きでパックマンのゲームを再現している．

## 1.4 本研究の目的

本研究ではパックマンを単純化させたゲームにおいて，機械学習を用いてパックマン (以下逃走者とする) ができるだけ長時間モンスター (以下追跡者とする) から逃げる逃走者 AI および追跡者が，逃走者をできるだけ早く捕まえる AI を開発する．本研究で用いる単純化したパックマンは，ゲームのマップはマスで区切られており追跡者は二人，逃走者は一人でそれぞれ 1 ターンに 1 マス動くと仮定する．また，パックマンには一定時間逃走者が追跡者を逆に食べられるようになるパワーエサがあるが，本研究ではパワーエサは無く逃走者は逃げるのみとする．この条件でいくつかのマップを作り機械学習させる，そのあとの評価方法として別のマップでのパフォーマンスをテストする．本研究ではパックマンを単純化する理由は追跡者の単純なパフォーマンスを正確に検証したりさまざまなマップに適応させるためである．

### 1.5　本報告書の構成

以下に本報告書の構成を示す．2 章では学習の環境などについて述べ，3 章では学習の手法について述べ，4 章ではプログラムについて述べる．5 章では結果について述べる．6 章では結論や今後の課題について述べる．

## 2　学習環境

本研究で使う学習環境は OpenAI Gym[6, 7] のインターフェイス形式を利用して環境を作成する．本研究の単純化されたパックマンの基本的なルールは (1) 逃走者と追跡者はそれぞれ全て幅 1 マス分の通路で構成されているマップで追跡，逃走する．(2) 逃走者一人と追跡者二人は順番に 1 ターンに一度それぞれ 1 マス動ける．(3) 追跡者のターンで逃走者と追跡者が隣り合って入れば追跡者の勝利になる．

本研究ではこのルールを元に 4 つの学習環境を用意した．

### 2.1　メインの学習環境

メインの学習環境は 9 × 9 のサイズの後述するマップ 5 枚のうち 1 枚と (A) ランダムで行動を決定するする逃走者，(B) 単純に距離を取ろうとする逃走者，(C) 後述する環境で学習した逃走者の三者のうち一者をエピソードの開始時点でランダムで決定し，さらにそれぞれの駒の初期位置も同様にランダムに配置する．付録 B に本研究で用いた学習用マップおよび検証用マップを示す．1 が通路を，0 が壁を表している．

### 2.2　逃走者の学習環境およびその学習のための追跡者の学習環境

まず，強化学習により逃走者を効率良く捕らえることができる追跡者を作成する．本研究では，ランダムな動きをする逃走者 (A) に対して，メインの学習環境と同様のマップの条件で 500,000 エピソード学習した追跡者 (X) を用意した．逃走者 (C) はメインの学習環境と同様のマップの条件で追跡者 (X) に対して同様に 500,000 エピソード学習する．

### 2.3　追加の学習環境

後述の理由で追加で 5 ターンに一回逃走者が止まるルールを追加した環境も用意した．

## 3　学習手法

本研究では python の機械学習ライブラリである PFRL を利用し PPO[5] と呼ばれる手法で機械学習を進めていく，ニューラルネットワークの入力層は 9 × 9 の数値で中間層の数は 13 で隠れ層のニューロン数は 128，出力層は 0-15 の int 型にする．

## 4　学習プログラム

本章では，本研究で作成したプログラムについて述べる．付録 A に本研究で作成したプログラムのソースを示す．

## 4.1 main_exam

このクラスではニューラルネットワークの組み立てから学習までを一括で行う，またテスト結果をファイルに保存する．本研究では読み込む学習環境やテスト環境を変更しやすくなるように主要な変更点はプログラムの最初にできるだけ記述している．

## 4.2 ChaceEnv3

ChaceEnv3 はメインの学習環境を表すクラスである．また ChaceEnv3 は Open AI Gym[6, 7] のインターフェイス形式に則って作成している．

### 4.2.1 コンストラクタ

Open AI Gym のインターフェイス形式に則って行動空間，観測空間，報酬範囲を定義する．

### 4.2.2 setMap

マップをリストにセットする．

### 4.2.3 set_runner_type

(テスト用) 逃走者の種類を指定する．

### 4.2.4 reset

Open AI Gym のインターフェイス形式に則って環境をリセットする，ここではマップをランダムに変更したり逃走者をランダムに指定したり，各関数のリセットを行う．

### 4.2.5 step

Open AI Gym のインターフェイス形式に則って環境を進める，ここでは各駒の行動の処理を各メソッドに委譲する．

### 4.2.6 imputBoard

マップに各駒の移動を反映させるメソッド．

### 4.2.7 isCatch

追跡者が逃走者を捕らえたか判別するメソッド．

### 4.2.8 where

引数の駒がどこにあるのか検索するメソッド．

### 4.2.9 reward

報酬を計算するメソッド．

#### 4.2.10 run

逃走者の行動を逃走者の種類によって決定するメソッド．

#### 4.2.11 sinple_run

純粋に距離を取るアルゴリズムを実装するメソッド．

#### 4.2.12 serch

純粋に距離を取るアルゴリズムを実装する上で追跡者との距離を計算するメソッド．

#### 4.2.13 algorithm

機械学習したアルゴリズムで行動を決定するメソッド．

#### 4.2.14 render

現在の環境を表示するメソッド．

## 5 結果

本研究でのメインの学習環境で 500,000 エピソード試行し，学習した追跡者 (X) に対し，別のマップで追跡者 (X) が逃走者 (A)，(B)，(C) を追いかけ捕まえるまでの手数を求めるテストを 1000 エピソードテストした結果を表 1 に，また学習マップと同じマップでテストした結果を表 1 に示す．ただし，300 以上経過しても追跡者 (X) が各逃走者を捕まえられなかった場合は逃走成功とする．

表 1　メインのテストの結果

| 逃走者 | (A) | (B) | (C) |
|---|---|---|---|
| 平均手数 | 36.8 | 211.4 | 57.4 |
| 逃走成功数 | 7 | 626 | 48 |

表 1 より，追跡者 (X) は単純に追跡者から距離を取ろうとする逃走者 (B) を 6 割以上逃しており，追跡者 (X) の学習がうまくいっていないことが分かる．一方，表 2 より学習マップと同じマップでテストした場合，捕まえる速度が上がり，300 手逃げられることはなくなったので，過学習が起こったと考えられる．

表 2　学習環境での結果

| 逃走者 | (A) | (B) | (C) |
|---|---|---|---|
| 平均手数 | 14.7 | 39.7 | 19.9 |
| 逃走成功数 | 0 | 0 | 0 |

そこで 2.3 で述べたような逃走者が 5 ターンに 1 回移動を行わないというルールを加えた学習環境を使い学習させやすくした上で学習させた．この条件の下で，10,000 回，50,000 回，100,000 回学習した時の追跡者 (X) が逃走者 (A),(B),(C) を追いかけ捕まえるまでの手数を求めるテストを行った結果を表 3 に示す．この結果から学習回数を減らしても過学習が解消されないとわかる．

表 3　追加の学習環境での結果

| 学習回数 | (A) 平均手数 | (A) 逃走成功回数 | (B) 平均手数 | (B) 逃走成功回数 | (C) 平均手数 | (C) 逃走成功回数 |
|---|---|---|---|---|---|---|
| 100,000 | 53.2 | 31 | 223.1 | 637 | 102.5 | 225 |
| 50,000 | 50.3 | 25 | 218.8 | 625 | 105.0 | 218 |
| 10,000 | 49.4 | 23 | 220.6 | 626 | 114.8 | 240 |

## 6　結論・今後の課題

本研究では Open AI Gym を用いて，できるだけ短時間に追跡者が逃走者を捉える AI を作成した．本研究の結果では，学習するマップでの追跡 AI はかなりのパフォーマンスを発揮したが，学習マップ以外での追跡 AI のパフォーマンスは著しく低下した．また，学習環境を簡易化させ学習回数を調整したが思うような結果が得られなかった．今後の課題として，本研究で調整しきれなかったハイパーパラメーターやニューラルネットワークの調整があげられる．さらに学習を効率化することで学習回数を抑え，過学習が起こらないようにすることもあげられる．

## 7　謝辞

この項を借りてここまで指導してくださった先生方に感謝します.

## 参考文献


[1] PAC-MAN Official Website, バンダイナムコエンターテイメント, https://www.pacman.com/jp/

[2] 燧暁彦，三輪誠，鶴岡慶雅，近山隆 : TD(λ) 学習を用いた Ms. Pac-Man AI のモンテカルロ木探索の改善，情報処理学会研究報告，Vol.2013-GI-29, No.2, pp.1-8 (2013) http://id.nii.ac.jp/1001/00090380/

[3] 半田久志 : 強化学習の Ms.PacMan への適用，第 27 回ファジィシステムシンポジウム，WG1-4, pp.1322-1323 (2011) https://www.jstage.jst.go.jp/article/fss/27/0/27_0_307/_pdf/-char/ja

[4] 岩谷徹 : ゲーム AI の原点『パックマン』はいかにして生み出されたのか？，人工知能学会誌 Vol.34, No.1, pp.86-99 (2019) https://doi.org/10.11517/jjsai.34.1_86

[5] J.Schulman, F.Wolski, P.Dhariwal, A.Radford, O.Klimov : Proximal Policy Optimization Algorithms, Machine Learning, Computer Science (2017) https://arxiv.org/abs/1707.06347

[6] Open AI https://openai.com/

[7] Open AI Gym https://github.com/openai/gym

[8] 池畑望，伊藤毅志 : Ms. Pac-Man におけるモンテカルロ木探索，情報処理学会論文誌，Vol.52, No.12, pp.3817–2827, (2011), http://id.nii.ac.jp/1001/00079584/

[9] Nguyen Quang Kien, THAWONMAS Ruck : Ms. Pac-Man におけるモンテカルロ木探索を用いたゴーストチームの協力制御，平成 23 年度 情報処理学会関西支部 支部大会 講演論文集, C-11, (2011), http://id.nii.ac.jp/1001/00080488/

[10] 中村昌弘，白川哲夫，THAWONMAS Ruck : Ms. Pac-Man シミュレータにおけるリスク戦術の使用頻度のオンライン調整, 2013 年度 情報処理学会関西支部 支部大会 講演論文集, B-05, (2013), http://id.nii.ac.jp/1001/00096793/

[11] Isha Salian : 40 Years on, PAC-MAN Recreated with AI by NVIDIA Researchers, NVIDIA, (2020), https://blogs.nvidia.com/blog/2020/05/22/gamegan-research-pacman-anniversary/

# 付録 A　ソースプログラム

本研究で作成した主なプログラムのソースファイルを以下に示す.

Listing 1　メインの学習環境

```
import pfrl
import torch
import torch.nn
import gym
import numpy as np
from torch import nn
import contextlib
from gym import spaces
import random
from RunEnv import ChaceEnv as c

maps=[]
turn=rew=runner_type=0
playMap=([1,1],[1,1])
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
A = torch.randn(3, 5)
A.to(device)



l=r=9
hidden=128
env = c()
env.setMap('nine_1.csv')
env.setMap('nine_2.csv')
env.setMap('nine_3.csv')
act = env.action_space.n


model = nn.Sequential(

        nn.Linear(l, hidden),
        nn.ReLU(),
```

```
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Flatten(),
        nn.Linear(hidden*r, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        pfrl.nn.Branched(
            nn.Sequential(
                nn.Linear(hidden, act),
                pfrl.policies.SoftmaxCategoricalHead(),
            ),
            nn.Linear(hidden, 1),
        ),
)
optimizer = torch.optim.Adam(model.parameters(), eps=1e-5)

phi = lambda x: x
agent=pfrl.agents.PPO(
```

```
    model,
    optimizer,
    gpu=0,
    phi=phi,
    update_interval=2048*32,
    minibatch_size=256*32
)

agent.load('best_run_agent')
class ChaceEnv(gym.Env):
    metadata = {'render.modes': ['console']}


    def __init__(self):
        super(ChaceEnv, self).__init__()
        self.action_space = spaces.Discrete(16)
        self.observation_space = spaces.Box(-1,5,(1,),dtype=np.float32)
        self.reward_range = (-5.0,1.0)

    @contextlib.contextmanager
    def setMap(self,fileName):
        m=np.loadtxt(fileName)
        maps.append(m)
        print('map ',fileName,'is roaded')
        return len(m),len(m[0])

    def set_runner_type(self,r_type):
        global runner_type
        runner_type=r_type

    def reset(self):
        global playMap,turn,rew,runner_type
        playMap=maps[random.randint(0,len(maps)-1)].copy()
        num=np.count_nonzero(playMap==1)
        rew=1.0
        turn=0
        runner_type=random.randint(0,2)
```

```
        for p in range(2,5):
            put=random.randint(0,num-1)
            num-=1
            for list in playMap:
                for i in range(0,len(list)):
                    if(list[i]==1):
                        if(put==0):
                            list[i]=p
                            put=-1
                            break
                        else:
                            put-=1
                            continue
        return torch.tensor(playMap.tolist(),dtype=torch.float32)

    def step(self, action):
        done=False;
        reward=self.reward();
        global turn,rew
        done=False;
        info=0;
        self.inputBoard(2,self.run())
        done=self.isCatch()
        self.inputBoard(3,int(action/4))
        self.inputBoard(4,action%4)
        turn+=1
        rew-=turn/50000
        if(done):
            reward=rew
        if(rew<0):
            done=True
        info={}
        return torch.tensor(playMap.tolist(),dtype=torch.float32), float(reward),
        done, info

    def inputBoard(self,playerType,mov):
        for list in playMap:
```

```
            movlist=[]

        line,row=self.where(playerType)
        if(mov==0):
            if(line!=0):
                if(playMap[line-1][row]==1):
                    playMap[line-1][row]=playerType
                    playMap[line][row]=1

        if(mov==1):
            if(row!=0):
                if(playMap[line][row-1]==1):
                    playMap[line][row-1]=playerType
                    playMap[line][row]=1
        if(mov==2):
            if(line!=len(playMap)-1):
                if(playMap[line+1][row]==1):
                    playMap[line+1][row]=playerType
                    playMap[line][row]=1
        if(mov==3):
            if(row!=playMap[0].size-1):
                if(playMap[line][row+1]==1):
                    playMap[line][row+1]=playerType
                    playMap[line][row]=1

    def isCatch(self):
        global playMap
        line=-1
        for list in playMap:
            movlist=[]

        line,row=self.where(2)
        if(line!=0):
            if(playMap[line-1][row]>1):
                return True
        if(row!=0):
            if(playMap[line][row-1]>1):
```

```
                return True
        if(line!=len(playMap)-1):
            if(playMap[line+1][row]>1):
                return True
        if(row!=playMap[0].size-1):
            if(playMap[line][row+1]>1):
                return True


        return False

    def where(self,playerType):
        line=row=-1
        for list in playMap:
            movlist=[]

            try:
                line+=1
                row=np.where(list==playerType)[0][0]
                movlist=list
                break
            except IndexError:
                pass
        return line, row

    def reward(self):
        rew=0
        line,row=self.where(2)
        l,r=self.where(3)
        rew=abs(line-l)+abs(row-r)
        l,r=self.where(4)
        if(rew>(line-l)+abs(row-r)):rew=abs(line-l)+abs(row-r)
        return 0.004-rew/250.0

    def run(self):
        global runner_type
        if(runner_type==0):
```

```
            return random.randint(0,4)
        if(runner_type==1):
            return self.sinple_run()
        if(runner_type==2):
        return self.algorithm(torch.tensor(playMap.tolist(),dtype=torch.float32),
        self.reward(), False, False)

    def sinple_run(self):
        line,row=self.where(2)
        if(line>0):
            u=self.search(line-1,row,3,3)
        else:
            u=0
        if(row>0):
            l=self.search(line,row-1,3,3)
        else:
            l=0
        try:
            d=self.search(line+1,row,3,3)
        except IndexError:
            d=0
        try:
            r=self.search(line,row+1,3,3)
        except IndexError:
            r=0
        if(u>d):
            if(l>r):
                return 1 if l>u else 0
            else:
                return 3 if r>u else 0
        else:
            if(l>r):
                return 1 if l>d else 2
            else:
                return 3 if r>d else 2

    def search(self,line,row,time,max):
```

```
global playMap
if(playMap[line][row]==0):
    if(max==time):
        return 0
    else:
        return time+1
if(playMap[line][row]>2):
    return 0
if(time==0):
    return 1
if(line>0):
    u=self.search(line-1,row,time-1,max)
else:
    u=time+1
if(row>0):
    l=self.search(line,row-1,time-1,max)
else:
    l=time+1
try:
    d=self.search(line+1,row,time-1,max)
except IndexError:
    d=time+1
try:
    r=self.search(line,row+1,time-1,max)
except IndexError:
    r=time+1
if(u<d):
    if(l<r):
        return l+1 if l<u else u+1
    else:
        return r+1 if r<u else u+1
else:
    if(l<r):
        return l+1 if l<d else d+1
    else:
        return r+1 if r<d else d+1
```

```
    def algorithm(self,obs, r, done, reset):
        with agent.eval_mode():
            agent.observe(obs, r, done, reset)
            return agent.act(obs)




    def render(self, mode='console'):
        if mode != 'console':
            raise NotImplementedError()
        print(playMap)

    def close(self):
        pass
```

Listing 2　メインの学習プログラム

```
import pfrl
import torch
import torch.nn
import gym
import numpy
from ChaceEnv3 import ChaceEnv
from torch import nn



env = ChaceEnv()
#マップのサイズは同じものを使います
line=9
row=9

env.setMap('nine_1.csv')
env.setMap('nine_2.csv')
env.setMap('nine_3.csv')
env.setMap('nine_4.csv')
env.setMap('nine_5.csv')
```

```
hidden=128

print('observation space:', env.observation_space)
print('action space:', env.action_space)

obs = env.reset()
print('initial observation:', obs)

action = env.action_space.sample()
obs, r, done, info = env.step(action)
print('next observation:', obs)
print('reward:', r)
print('done:', done)
print('info:', info)


act = env.action_space.n

model = nn.Sequential(

        nn.Linear(line, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
```

```
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Flatten(),
        nn.Linear(hidden*row, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        nn.Linear(hidden, hidden),
        nn.ReLU(),
        pfrl.nn.Branched(
            nn.Sequential(
                nn.Linear(hidden, act),
                pfrl.policies.SoftmaxCategoricalHead(),
            ),
            nn.Linear(hidden, 1),
        ),
)

optimizer = torch.optim.Adam(model.parameters(), eps=1e-5)

phi = lambda x: x
agent = pfrl.agents.PPO(
    model,
    optimizer,
    gpu=0,
    phi=phi,
    update_interval=2048*64,
    minibatch_size=256*16
)



n_episodes = 500000
max_episode_len = 600
a=0
for i in range(1, n_episodes + 1):
```

```
    obs = env.reset()
    R = 0
    t = 0
    while True:
        action = agent.act(obs)
        obs, reward, done, _ = env.step(action)
        R += reward
        a+=reward
        t += 1
        reset = t == max_episode_len
        agent.observe(obs, reward, done, reset)

        if done or reset:
            break
    if i % 500 == 0:
        print('Last action:',action)
        print('reward avarage',a/500)
        a=0
        print('episode:', i, 'R:', R)
    if i % 5000 == 0:
        print('statistics:', agent.get_statistics())
print('Finished.')


with agent.eval_mode():
    for i in range(10):
        obs = env.reset()
        R = 0
        t = 0
        actionlist=[]
        while True:
            action = agent.act(obs)
            actionlist.append(action)
            obs, r, done, _ = env.step(action)
            R += r
            t += 1
            reset = t == 600
```

```
            agent.observe(obs, r, done, reset)
            if done or reset:
                break
        print('evaluation episode:', i, 'R:', R)
        print(actionlist)


# Save an agent to the 'agent' directory
agent.save('agent')
```

## 付録 B　マップ

本研究で用いた学習用マップおよび検証用マップを以下に示す.

Listing 3　学習マップ 1

```
1 1 1 1 1 1 1 1 1
1 0 0 1 0 1 0 0 1
1 0 0 1 0 1 0 0 1
1 1 1 1 1 1 1 1 1
1 0 0 1 0 1 0 0 1
1 1 1 1 1 1 1 1 1
1 0 0 1 0 1 0 0 1
1 0 0 1 0 1 0 0 1
1 1 1 1 1 1 1 1 1
```

Listing 4　学習マップ 2

```
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
```

Listing 5　学習マップ 3

```
1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 1
```

```
1 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1
```

Listing 6　学習マップ 4

```
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 0 0 1 0 1
1 1 1 0 0 0 1 1 1
1 0 1 0 0 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
```

Listing 7　学習マップ 5

```
0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 0
0 1 0 1 0 1 0 1 0
0 1 1 1 1 1 1 1 0
0 1 0 1 0 1 0 1 0
0 1 1 1 1 1 1 1 0
0 1 0 1 0 1 0 1 0
0 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0
```

Listing 8　テストマップ 1

```
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 0 0 1
1 0 1 0 1 1 1 1 1
1 0 1 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 0 0 1
1 0 1 0 1 1 1 1 1
```

```
1 0 1 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1
```

Listing 9　テストマップ 2

```
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
```

Listing 10　テストマップ 3

```
1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 0 1 0 1 1 1
1 0 0 1 1 1 0 0 1
1 1 1 1 0 1 1 1 1
1 0 0 1 1 1 0 0 1
1 1 1 0 1 0 1 1 1
1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
```