基于深度强化学习的群体协同策略研究

摘要

本文探究了深度强化学习在多主体群体协同任务中的应用。本文首先从单个体任务出发，实现了诸如DQN、...、...等经典的深度强化学习算法，并将这些算法与传统的启发式算法进行比较，通过实验证明深度强化学习算法相较于传统算法的优越性。随后，本文将单个体任务实验环境拓展到多个体任务实验环境，并针对多主体群体协同任务对经典的强化学习算法进行相应的优化，探究在不同的群体协同任务环境中，深度强化学习算法与传统的基于规则的算法与启发式算法之间表现的差异，及不同的深度强化学习算法之间表现的优劣。实验证明，在多主体群体协同任务中，深度强化学习算法整体相较于传统的启发式算法拥有更好的表现，在相同时间内可以获得更高的回报值。因此，在多主体群体协同任务环境中，深度强化学习具有很好的适应性、鲁棒性与可扩展性，可以为多主体群体协同任务中的每个个体提供行为策略指导支持。

关键词：深度学习、强化学习，多主体群体协同

第一章 绪论

1.1 研究背景

深度学习（DL、Deep Learning）作为近年来机器学习研究领域的重点不断取得各种突破性进展，已被广泛应用于计算机视觉、人机交互、自然语言处理等各个领域。深度学习是一种结构化神经网络学习模型，尝试从原始数据中构建高级抽象概念，因此也被称为层次化学习或深度网络学习。深度学习是一种深层模型，比浅层模型（例如SVM）具有更好的表示能力。深度学习通过多层次网络结构和非线性激活函数对输入数据进行非线性变换，提取组合数据的低层、局部特征，形成对数据的抽象高层表示。因此，深度学习侧重于对数据的抽象与特征提取。

另一方面，强化学习（RL、Reinforcement Learning）作为及其学习领域中的另一个热点算法模型也越来越得到学术界以及工业界的关注。强化学习同样不断被人们应用到实际生活中的各个领域，例如游戏博弈以及机器人控制等领域。RL的基本思想是通过最大化个体（agent）从环境中获得的汇报值（reward），从而使个体学习到完成目标的最优策略。因此，强化学习更侧重于对解决问题所需策略的学习。

在深度学习领域在近年来取得突破性进展之前，强化学习不能获得大规模实际应用的原因在于强化学习很难面对和处理过大的状态和行为空间。而深度学习善于对数据进行抽象与特征提取的特点正好可以用来对强化学习所面对的复杂状态和行为空间进行建模，使得人们可以应用强化学习去解决更为复杂的问题。2013年，DeepMind提出了第一个强化学习和深度学习结合的模型，深度Q网络（DQN）[ref]。虽然DQN模型相对比较简单，只是面向有限的动作空间，但依然在Atari游戏上取得了很大的成功，超越了人类水平的成绩。之后，深度强化学习开始快速发展。一些基于DQN的改进包括双Q网络[ref]、优先级经验回放[ref]、决斗网络[ref]等深度强化学习模型开始陆续被提出。

深度强化学习的典型应用是用于指导单个体在特定任务环境中的策略决策。但是，在很多的复杂应用环境中，特定任务的完成牵扯到多个体的交互以及协同合作，多个体的彼此联系带来更大的行为空间以及更复杂的策略决策行为。举例来说，多机器人控制、多玩家游戏博弈（如Moba游戏）等皆涉及到多个体的群体协同策略决策。因此，强化学习算法在多个体协同环境中的可扩展性对于指导我们在更为复杂的群体协同环境中构建人工智能控制系统是至关重要的。

1.2 实验环境&工具简介

在1.1“研究背景”中已经大致介绍了本文工作的背景和研究方向。现在再来介绍一下本文实验代码的开发环境和相关工具，为后续工作的进行确立一个基础。

1.2.1 开发环境

系统环境

- CPU：2.8 GHz Intel Core i7
- 内存：16 GB 2133 MHz LPDDR3
- 操作系统：macOS High Sierra 10.13.3

开发环境

- python 3.6.5
- TensorFlow深度学习框架

1.2.2 强化学习实验环境Gym

本文应用强化学习和启发式算法所解决的具体任务全部使用OpenAI公司的Gym库编写。本节仅就其与本文工作相关的部分做一下简单介绍，主要包括以下几点

- Gym是什么?

  OpenAI Gym是一款用于研发和比较强化学习算法的工具包，它支持训练智能体（agent）做任何事，从行走到玩Pong或围棋之类的游戏都在范围之中。Gym对于智能体的具体结构形式不做任何要求，并且Gym与其它数值计算库及深度学习框架兼容，如TensorFlow，Theano等。现在主要支持的是python语言。

  OpenAI Gym包含两部分：

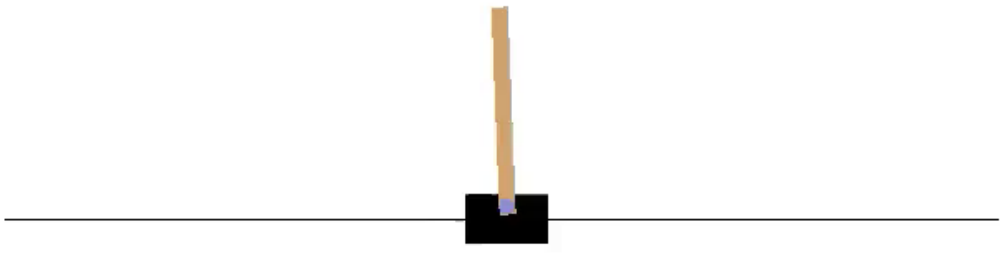
1. Gym开源：包含一个测试问题集，每个问题成为环境（environment），可以用于自己的强化学习算法开发。这些环境享有共享的接口，允许用户设计通用的算法，例如：Atari、CartPole等。
2. OpenAI Gym服务：提供一个站点和API，允许用户对自己训练的算法进行性能比较。

- Gym的核心类及方法
  1. Environment（环境）

     环境（Environment）是Gym的核心类，Environment定义并可视化一种具体的强化学习环境。个体（Agent）从环境中获取观察信息，采取策略并行动。环境接收个体的行为，向个体返回环境状态以及回报值等信息。

     以经典的cart-pole问题举例来说，执行以下代码可以创建一个cart-pole的强化学习环境：

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample()) # take a random
action
```



Gym库中的cart-pole实验环境

  2. Observation

     在Environment介绍部分的代码中，对于小车的控制是随机的。如果想要更好地在每一步控制环境中的个体，而不是单单采用简单的随机算法，那么我们就应该去了解环境中的个体的每一步行动对环境产生了什么样的影响。

     Environment类的step函数正是返回了我们所需要的环境信息。具体来说，step函数返回下面四个值：

- observation（object）：observation是一个由具体的实验环境所决定的object类型的变量，它代表着我们对于环境的观察信息。举例来说，observation可以是图片中的各个像素点信息，环境中机器人前进的速度和方向，以及棋类游戏的棋盘状态等。
- reward（float）：reward代表着个体在环境中已采取行为所获得的回报值。回报值在不同的环境中有着不同的取值方式和取值范围，但是我们的目标始终应当是最大化目标个体的回报值。
- done（boolean）：done代表着是否需要将当前的实验环境重置。大多数的任务被分割成许多预先定义好的迭代数目。如果done变量的返回值为true，则代表着当前的一轮迭代已经完成，需要将环境进行重置。

step函数实现了经典的“agent-environment循环”。在每一次迭代中，个体选择自己的行为（action），环境返回观察信息（observation）和回报值（reward）。

agent-environment循环

执行Environment类的reset()函数可以开始一轮新的迭代，reset函数返回一个初始的观察信息。所以当观察到step函数的done变量返回值为true时，应该执行reset函数来将环境重新初始化：

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {}
timesteps".format(t+1))
            break
```

输出节选：

```
[-0.061586   -0.75893141  0.05793238  1.15547541]
[-0.07676463 -0.95475889  0.08104189  1.46574644]
[-0.0958598  -1.15077434  0.11035682  1.78260485]
[-0.11887529 -0.95705275  0.14600892  1.5261692 ]
[-0.13801635 -0.7639636   0.1765323   1.28239155]
[-0.15329562 -0.57147373  0.20218013  1.04977545]
Episode finished after 14 timesteps
[-0.02786724  0.00361763 -0.03938967 -0.01611184]
[-0.02779488 -0.19091794 -0.03971191  0.26388759]
[-0.03161324  0.00474768 -0.03443415 -0.04105167]
```

3. Space

   每一种环境都对应着一个行为空间action_space和观察空间observation_space。这两个变量属于Space类，Space类描述了具体环境对应的合法的个体行为和观察信息。

```
import gym
env = gym.make('CartPole-v0')
print(env.action_space)
#> Discrete(2)
print(env.observation_space)
#> Box(4,)
```

   Discrete空间（离散空间）只允许有固定范围的非负整数，所以cart-pole环境中的个体的行为空间是离散空间，只能允许两种行动，用0和1代表。Box空间代表着一个n维框，在cart-pole环境中观察空间是一个包含四个数字的数组。Box空间存在着边界限制，在Box空间内的行动不能超出边界的限制：

```
print(env.observation_space.high)
#> array([ 2.4       ,         inf,  0.20943951,
inf])
print(env.observation_space.low)
#> array([-2.4       ,        -inf, -0.20943951,       -
inf])
```

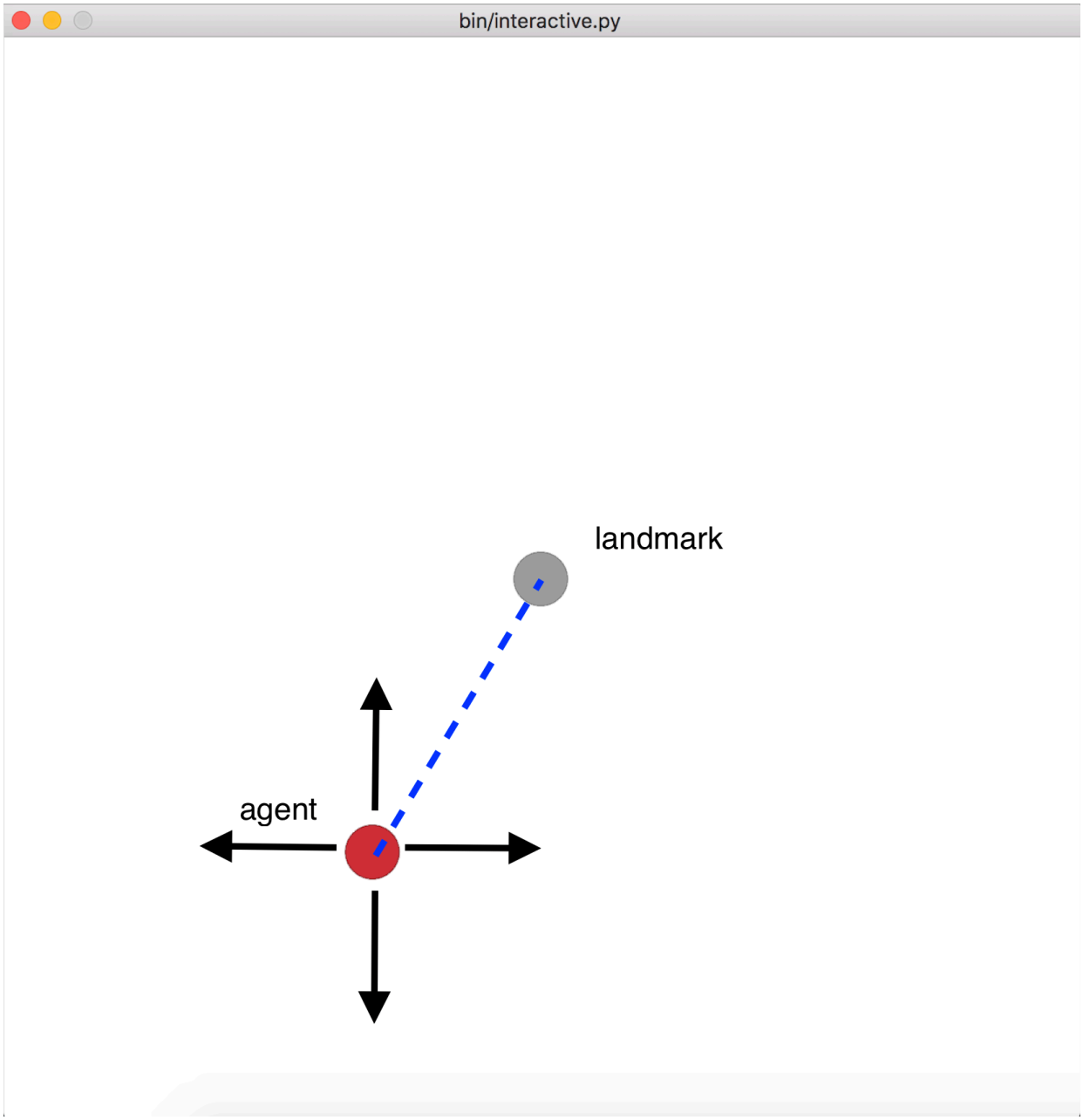
   可见cart-pole环境的观察空间的第一维取值范围是[-2.4, 2.4]，第三维是[-0.21, 0.21]，第二维和第四维不受限制。

1.3 论文结构

第二章 单一个体任务的深度强化学习探究

在进行多个体的群体协同策略研究之前，首先从更简单的情形，即单一个体的任务环境出发，实现后续多个体任务中将使用到的深度强化学习算法，在单一个体环境中检验实现算法的正确性及深度强化学习算法的有效性，为后续研究打好基础。

2.1 任务简介



首先用Gym实现一个最简单的单一个体的实验环境。该环境位于一个二维平面之中，平面包含了一个可以移动的agent，以及一个位置固定的landmark。agent的行动空间是离散空间。在任一时刻，agent可以有5种行动选择。即向上移动、向下移动、向左移动、向右移动和不采取行动。如果在某一时刻，agent选择向某一方向移动，则agent立

即结束此前的运动状态，并获得指定方向的一个初速度，不同时刻获得的初速度为常量。如果在某一时刻，agent选择不采取行动，若上一时刻agent处于静止，则在该时刻agent继续维持静止状态。若上一时刻agent正在向某一方向前进，则该时刻agent继续向该方向前进，但速度会衰减某一常数。

2.2 个体回报与任务目标

该单一个体实验的任务目标非常简单，即在尽可能短的时间内通过agent的上下左右移动到达landmark所在的位置。因此，agent所获得的回报值与当前agent和landmark的距离远近有关。agent与landmark相距越远，agent所获得的回报值越低；agent与landmark的距离越近，agent所获得的回报值越高。具体来说，agent所获得的汇报定义如下：

$$reward = -\left[(x_{agent} - x_{landmark})^2 + (y_{agent} - y_{landmark})^2\right]$$

其中x，y分别代表agent和landmark在二维平面内某一固定直角坐标系的横、纵坐标，即agent获得的回报值定义为当前状态下agent与landmark之间欧氏距离平方的负值。当回报值为0时，agent成功到达landmark所在位置。

2.3 观察信息

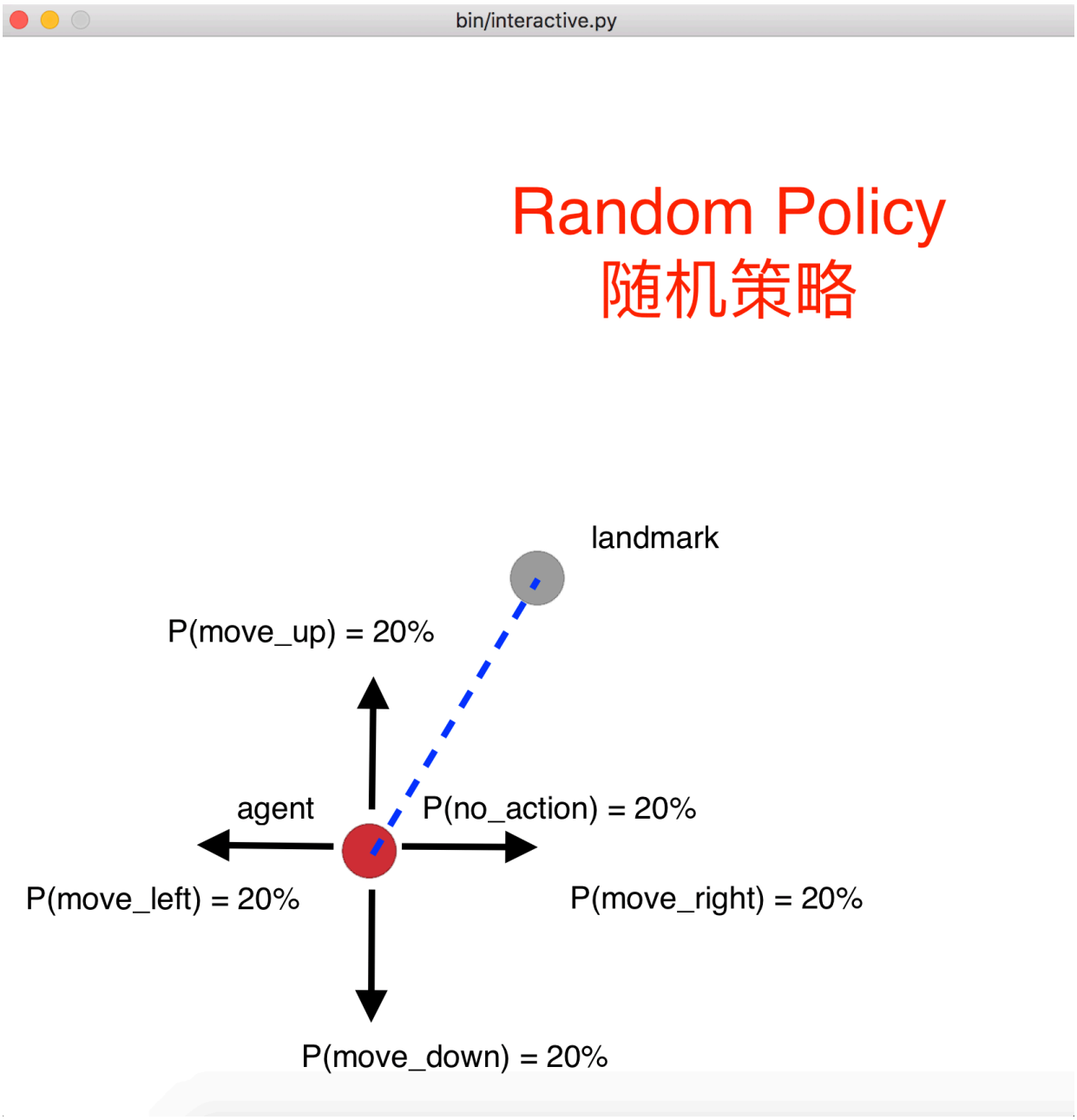
在任一时刻，agent可以从环境中获得的信息有agent自身的速度信息，即agent正在以多大的速度向哪一方向前进；agent与landmark之间的相对位置信息，即landmark处于agent的哪一方位，距离有多远；以及agent当前所能获得的回报值。

值得一提的是，agent尽管可以在每一时刻从环境中获得回报值，但agent并不了解回报值的具体定义形式，也不知道越靠近landmark回报值越高这一预先设定的规则。深度强化学习的目标就是让agent通过不断的尝试，观察回报值的变化规律，最终根据自身的观察信息作出正确的行为决策，学习到越靠近landmark回报值越高这一规律，从而通过最大化回报值来完成移动到达landmark位置的任务目标。

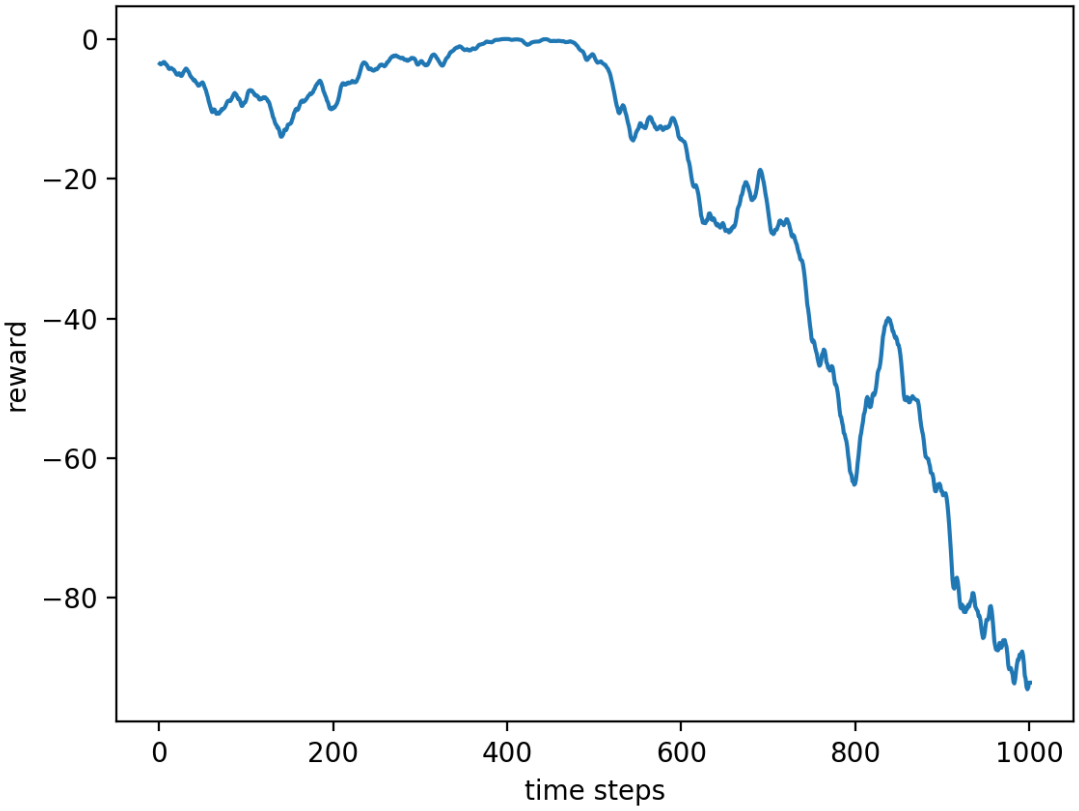
2.4 算法实现

2.4.1 随机策略

首先从最简单的策略入手，即随机策略。agent完全忽略从环境中获得的一切观察信息，包括自身速度、landmark相对位置以及汇报值，在每一次做出决策的时刻等概率地从向上移动、向下移动、向左移动、向右移动以及不采取行动中等概率的选择一种行为，用来指导agent的后续行动。



很显然，随机策略并不能完成既定的任务目标。agent只是随机地在二维空间中进行游走，而并没有任何主动去靠近landmark的行为尝试。下面的图片展示了经过1000轮循环，即agent采取1000次决策行为的过程中agent所获得的回报值变化情况：

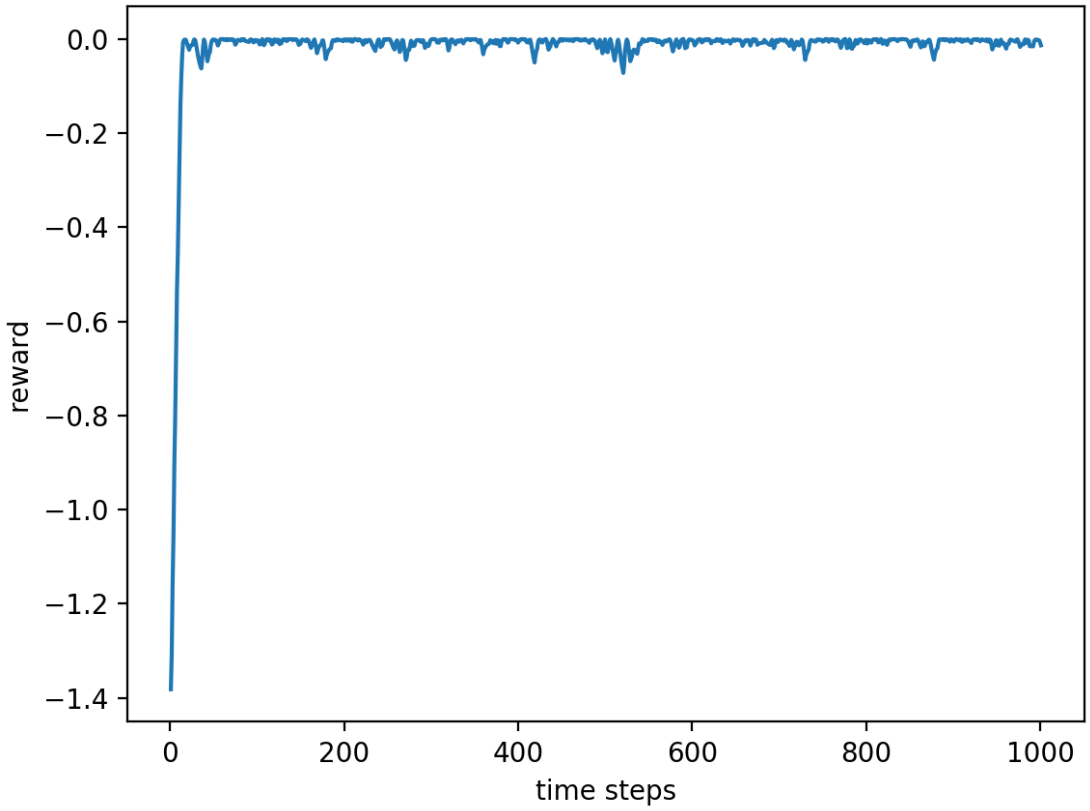


从图中可以看到，随机算法得到的实验结果是不收敛的。agent所获得的回报值越来越低，说明agent做无意义的随机游走，并最终趋向于远离landmark。随机策略完全不能完成该环境所定义的任务。

2.4.2 完全信息策略

在2.4.1中考虑了一种极端情况，即agent完全忽略环境中的一切信息，随机的采取行动。下面考虑另一种极端情况，即假设agent已经完全获得了环境中的一切信息，包括回报值的定义方式以及任务的最终目标。换句话说，用我们的人类知识去指导agent的行为，完成任务。对这一简单任务，一种简单直观的人类策略方式是首先观察landmark相对于agent的位置，由于任务的最终目标是使得agent运动到landmark所在的位置，那么agent在每一步的行为应当缩短其与landmark之间的距离，即agent在每一时刻始终向landmark的位置采取行动，并且在水平和垂直两个方向上优先选择相对距离较大的方向采取行动。如下图所示：

在当前状态下，agent选择向上移动，因为landmark此时位于agent的右上方，并且landmark在垂直方向上距离agent的相对距离要大于水平方向上的相对距离。agent优先在相对距离较大的方向上采取行动，因此agent向上移动。对于完全信息策略，同样地进行1000次循环，让agent执行1000次决策行为，观察agent所获得的回报值情况，获得的结果如下：



可以看到，不同于随机策略，完全信息策略在很短时间内得到收敛结果。agent所获得的回报值在短短几个time step内便趋近于0，即agent很快到达了landmark所在位置，并在landmark所在位置附近震荡。完全信息策略很好的完成了任务目标，当然这也是理所当然的。但完全信息策略的意义在于它为后续的深度强化学习算法制定了标准，如果后续的深度强化学习算法可以接近甚至超过完全信息策略的表现，那么深度强化学习算法就是有效的。