18 Reinforcement Learning - Deep Q-Learning

本节实验目标：在网格世界中实现和体会Deep Q-Learning算法。

报告说明（重要）：本部分“马尔可夫决策和强化学习”包含三周内容，本节课为第三周内容，请留意该文档目录中标有“报告应实现任务[分值]”的部分，请在实现后及时保存文件和运行结果截图，在这周的课程报告中一并提交。

Introduction

在该堂实验课，你将**实现一个Deep Q函数。**

实验代码包括一个autograder，以下命令对所有问题运行以测试你的实现：

```
python autograder.py
```

你也可以通过以下形式的命令做特定case的测试：

```
python autograder.py -t test_cases/q2/1-bridge-grid
```

实验代码

你应该要编辑的文件：	
valueIterationAgents.py	Q1所需：**用于解决已知 MDP 的价值迭代agent**
qlearningAgents.py	Q3-6所需：Q-learning agents for Gridworld, Crawler and Pacman
analysis.py	Q2所需
model.py	Q7所需：Deep Q Network，用于帮助 pacman 计算大型 MDP 中的 Q 值
你应该看但不要编辑的文件：	
mdp.py	Defines methods on general MDPs. 里面的methods可以调用。
learningAgents.py	Defines the base classes `ValueEstimationAgent` and `QLearningAgent`, which your agents will extend.
util.py	Utilities, including `util.Counter`, which is particularly useful for Q-learners.
gridworld.py	The Gridworld implementation.
featureExtractors.py	Classes for extracting features on (state, action) pairs. Used for the approximate Q-learning agent (in `qlearningAgents.py`).
deepQLearningAgents.py	Training loop for the Deep Q-learning agent.
你可以忽略的文件：	
environment.py	Abstract class for general reinforcement learning environments. Used by `gridworld.py`.
graphicsGridworldDisplay.py	Gridworld graphical display.
graphicsUtils.py	Graphics utilities.
textGridworldDisplay.py	Plug-in for the Gridworld text interface.
crawler.py	The crawler code and test harness. You will run this but not edit it.
graphicsCrawlerDisplay.py	GUI for the crawler robot.
autograder.py	Project autograder
testParser.py	Parses autograder test and solution files
testClasses.py	General autograding test classes
test_cases/	Directory containing the test cases for each question
reinforcementTestClasses.py	Specific autograding test classes

注意事项

- 请***不要***更改代码中提供的任何函数或类的名称，否则会对 autograder 造成严重破坏。

- **正确使用数据集**：你在此项目中的部分分数取决于你训练的模型在自动评分器的测试集上的表现。我们不提供任何 API 供你直接访问测试集。
- 如果你发现自己卡在某事上，请联系老师、助教或同学寻求帮助。我们希望这些项目是有益的和指导性的，而不是令人沮丧和沮丧的。

MDPs in Grid World

首先，试着用键盘上下左右箭头在手动控制模式下运行 Gridworld：

```
python gridworld.py -m
```

你将看到课堂上所讲Grid World的两个出口布局。 蓝点是agent。 请注意，**当你按下 *up* 时，agent实际上只有 80% 的时间向北移动。 这就是这个世界中agent的生活，总有不如意的时候！**

你可以控制模拟器的许多方面。 运行以下命令可获得完整的选项列表：

```
python gridworld.py -h
```

默认agent随机移动

```
python gridworld.py -g MazeGrid
```

你应该看到随机agent在网格上来回弹跳，直到它到达了exit的网格（右上角）时。对于 AI agent来说，这不是最好的情况。

- 注意：Gridworld MDP 是这样的，你首先必须进入预终止状态（**GUI 中显示的右上角双框），然后在一段经历（episode）实际结束之前采取特殊的"exit"操作（真正的终止状态称为 `TERMINAL_STATE` ，未在 GUI 中显示）**。如果你手动运行一次 episode，你的总回报可能会低于你的预期，这是由于**折扣因子（可通过 `-d` 更改；默认为 0.9）**的存在。
- 查看跟图形界面同时输出的**控制台(Terminal)输出，你可以看到agent经历的每次转换**（要关闭此功能，请在上述命令中使用 `-q` ）。
- 与 Pacman 中一样，agent 位置由 `(x,y)` 笛卡尔坐标表示，数组由 `[x][y]` 索引， `'north'` 动作增加 `y` ，等等。默认情况下，大多数动作将获得零回报。**你可以使用实时奖励选项 (`-r`) 更改回报值。**

Question 1-6 请查看上两节课的实验文档并先完成

Question 7: Deep Q-Learning 报告应实现任务[2分]

- **说明：该小题截图只需截 python autograder.py -q q6 的运行结果，获得评分器给的3分中，这道题才算得它占的2分。**

注意：Question 7 还挺难的，以至于我们教学团队也没有一个很好的参考实现，如果你舍得放弃这两分实验课期末总评，可以在这次报告只提交Q1-6的代码和结果，把时间花在期末复习上。

对于本学期的最后一个上机练习问题，你将结合上一周的 Q-learning 和Project 3 机器学习。**在 `model.py` 中，你将实现 `DeepQNetwork`** ，这是一个神经网络，可以预测给定状态下所有可能动作的 Q 值。

你将实现以下功能：

- `__init__()` ：你将在这里初始化你的神经网络的所有参数。你还必须初始化以下变量：
  - `self.parameters` ：包含所有参数的列表，按前向传递的顺序排列。
  - `self.learning_rate` ：你将在 `gradient_update()` 中使用它。
  - `self.numTrainingGames` ：Pacman 将玩的游戏数量，以收集转移并学习其 Q 值；请注意，这应该大于 1000，因为大约前 1000 个游戏用于探索而不用于更新 Q 网络。
  - `self.batch_size` ：模型应该用于每次梯度更新的转移数。autograder将使用此变量；设置后你不需要访问此变量。
- `get_loss()` ：返回预测的 Q 值（由你的网络输出）和 Q_targets（你将其视为ground truth）之间的平方损失。

- `run()`：你将在其中返回通过 Q 网络的前向传递的结果。 输出应该是一个大小为 (batch_size, num_actions) 的向量，因为我们想要返回给定状态的所有可能动作的 Q 值。
- `gradient_update()`：遍历你的`self.parameters`并根据计算的梯度更新它们中的每一个。但是，你不会在此函数中迭代整个数据集，也不会重复更新参数直到收敛。这个函数应该只对每个参数执行一次梯度更新。autograder将重复调用此函数来更新你的网络。

为了方便理解：通过使用以下等式“引导(bootstrapping)”你的模型，为每个样本（s、a、r、s'）计算 $Q_{targets}$：

$Q_{target}(s,a) = r(s,a,s') + (1 - done)\gamma \max_{a'} \hat{Q}(s',a')$

其中变量 $done$ 表示一个episode是否已经结束（Pacman在从状态 s 采取行动 a 后赢或输），$\hat{Q}$ 是你的 Q 网络。请注意近似 Q 学习与深度 Q 学习中的 Q 目标公式之间的相似之处。

评分： 在你的agent训练完 `self.numTrainingGames` 游戏后，我们将运行你的 Deep Q learning Pacman agent 进行10场游戏。如果你的agent至少赢得了 6/10 的比赛，那么你将获得全部分数。如果你的agent赢得了至少 8/10 的比赛，那么你将获得 1 个额外的分数（5/4）。请注意，尽管我们在后端训练循环中实现了一些tricks，但深度 Q 学习并不以其稳定性著称。你的agent赢得的游戏数量可能因每次运行而异。要获得额外的分数，你的实现应始终超过 80% 的阈值。

```
python autograder.py -q q7
```

恭喜！ 你已经训练了一个深度 RL Pacman 并完成了我们这学期的上机练习！ 如果你认为这很酷，请尝试在更难的布局上训练你的模型：

```
python pacman.py -p PacmanDeepQAgent -x [numGames] -n [numGames + 10] -l testClassic
```

报告7 “马尔可夫决策+强化学习” 提交说明

- 按照第16周、17周和本周第18周的Question 1-7 的要求实现代码和获得运行结果截图
- **提交压缩包命名为”姓名_学号_报告序号.zip”** (如“彭振辉_2106666_报告7.zip”)
- 压缩包应**包含内容：**
  - **已实现的完整项目文件夹”Project_7_mdp_rl_Full“**
    - 其中应包含 Question 1-7 的要求实现代码
  - **一个doc或pdf说明文档，上面需要有：**
    - 开头一段说明 “**整体实现参考 + 2-3句简要体会（如教训、思路、拓展应用等）**”，如：-
      - “自行实现。挑战最大的是xxx内容，初始时报了什么错，通过什么方式解决，该部分的实现思路为xxx”
      - “xxx内容参考xxx同学/xxx网址。思考不出算法思路，探究后学习到了什么方法。”
    - **要求实现的7个任务的成功运行截图，说明截图对应任务。**