**中山大学计算机学院**

**人工智能**

**本科生实验报告**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | **23336179** | 姓名 | **马福泉** |

# 实验题目

# 实现DQN算法

**在CartPole-v0环境中实现DQN算法。最终算法性能的评判标准：以算法收敛的reward大小、收敛所需的样本数量给分。 reward越高（至少是180，最大是200）、收敛所需样本数量越少，分数越高。**

# Submission

**作业提交内容：需提交一个zip文件，包括代码以及实验报告PDF。实验报告除了需要写writing部分的内容，还需要给出reward曲线图以及算法细节。**

**相关代码下载地址： https://github.com/ZYC9894/2024AI/tree/main/Homework/Experiment**

**相关环境的说明文档：https://www.gymlibrary.dev/ 或 https://www.gymlibrary.dev/environments/classic\_control/cart\_pole/**

# Supplement

**我们给出DQN在cartpole环境的训练曲线图作为参考。**

# 实验内容

1. 算法原理
2. 主程序模块 (main.py)
3. 参数解析(parse函数):

使用argparse模块定义和解析命令行参数

包含训练参数(学习率、批次大小、缓冲区大小等)

包含网络参数(隐藏层大小、更新频率等)

包含探索参数(epsilon及其衰减)

1. 运行控制(run函数):

对每个随机种子运行独立实验

初始化环境和DQN代理

调用训练过程

收集并可视化结果

1. 结果可视化(visualize\_results函数):

计算奖励的标准差

生成两种可视化图表:

所有实验的奖励曲线和标准差曲线

子图布局的详细分析(各实验曲线+标准差)

1. DQN代理模块 (agent\_dqn.py)
2. Q网络(QNetwork类):

实现深度Q网络的结构

包含两个隐藏层和一个输出层，使用ReLU激活函数

1. 经验回放缓冲区(ReplayBuffer类):

使用deque实现固定大小的缓冲区

提供push、sample和clean方法

支持随机采样训练数据

1. DQN代理(AgentDQN类):

初始化Q网络和目标网络

配置优化器和超参数

* 动作选择(make\_action方法):

实现ε-greedy策略

* 训练过程(run方法):与环境交互收集经验

存储经验到回放缓冲区

定期更新网络参数

实现epsilon衰减。

* 网络更新(\_update\_network方法):

从缓冲区采样批次数据

计算当前Q值和目标Q值

使用MSE损失更新网络

实现目标网络定期更新

1. 关键代码展示（可选）
2. 使用argparse库来定义和解析运行深度Q网络（DQN）训练所需的各种配置参数。它设置了参数的名称、类型、默认值和帮助信息，使得用户可以通过命令行灵活地指定训练配置，如环境名称、网络结构、学习率、折扣因子等。解析完成后，将这些参数打包成一个命名空间对象并返回，以便在程序的其他部分使用。



1. run函数：

初始化奖励记录列表：用于收集所有实验的奖励数据。

循环执行实验：对每个随机种子，执行以下步骤：

创建环境实例，用于与智能体交互。

设置当前实验的随机种子，以确保结果的可重复性。

初始化DQN智能体，并根据命令行参数配置其属性。

如果指定训练DQN，则调用智能体的train方法进行训练，并收集训练过程中的奖励记录。

收集实验结果：将每次实验的奖励记录添加到结果列表中。

结果可视化：调用visualize\_results函数，将收集到的实验结果进行可视化处理，并将图表保存到日志目录中。



1. QNetwork类

* 初始化（\_\_init\_\_ 方法）：

定义了三个全连接层（nn.Linear）：

layer1：输入层，将环境的观测状态（input\_size）映射到隐藏层（hidden\_size）。

layer2：第二个隐藏层，进一步处理来自第一个隐藏层的特征。

head：输出层，将第二个隐藏层的输出映射到动作空间的大小（output\_size），即每个可能动作的Q值。

这些层在初始化时被注册为模型的子模块，PyTorch会自动管理它们的参数。

* 前向传播（forward 方法）：

定义了数据通过网络的前向传播过程：

输入数据首先通过layer1，然后应用ReLU激活函数，增加网络的非线性表达能力。

接着，经过layer2和ReLU激活函数，进一步处理特征。

最后，通过head层输出每个动作的Q值。

这个方法在每次调用模型进行预测时被自动执行。



1. ReplayBuffer类

* 初始化：设置缓冲区的容量（buffer\_size），即最多可以存储的经验数量。

创建一个deque对象作为存储结构，它是一个双端队列，可以高效地从两端添加或删除元素。设置其最大长度为缓冲区容量，以确保缓冲区不会无限增长。

* 获取长度（\_\_len\_\_ 方法）：

返回缓冲区当前存储的经验数量。

* 添加经验（push 方法）：

将新的经验（如状态转移元组）添加到缓冲区中。如果缓冲区已满，旧的经验将被自动替换。

* 采样（sample 方法）：

从缓冲区中随机抽取指定数量（batch\_size）的经验样本。这些样本用于训练神经网络，以更新智能体的策略。



1. AgentDQN类  
   篇幅问题，展示伪代码

初始化智能体：

设置环境和参数

设置随机种子

选择设备（CPU或GPU）

初始化经验回放缓冲区

初始化Q网络和目标Q网络

初始化优化器

初始化训练参数

训练智能体：

对于每个帧：

重置环境状态

对于每个episode直到达到最大帧数：

如果未结束：

根据当前策略选择动作

执行动作并观察结果

将经验存储到缓冲区

如果缓冲区足够大：

从缓冲区中采样一批经验

更新Q网络

如果达到更新频率：

更新目标Q网络

更新探索率

如果结束：

记录奖励并重置环境状态

每10个episode打印一次进度和平均奖励

选择动作：

如果测试或探索率大于随机数：

选择具有最大Q值的动作

否则：

随机选择动作

更新Q网络：

从缓冲区中采样一批经验

计算当前Q值和目标Q值

计算损失并更新Q网络参数

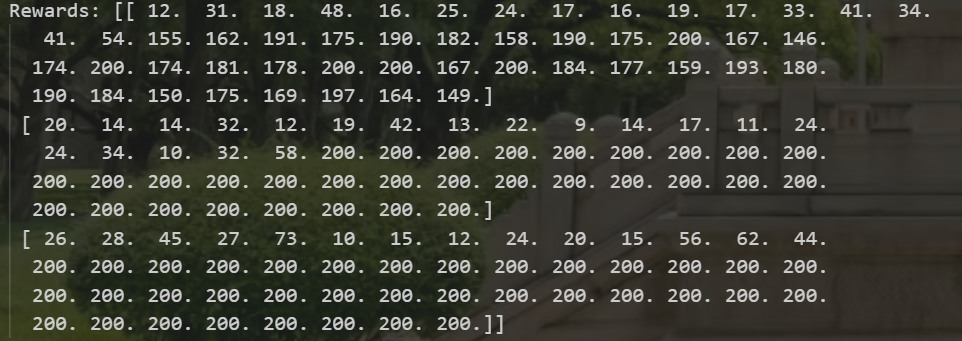
调整学习率：

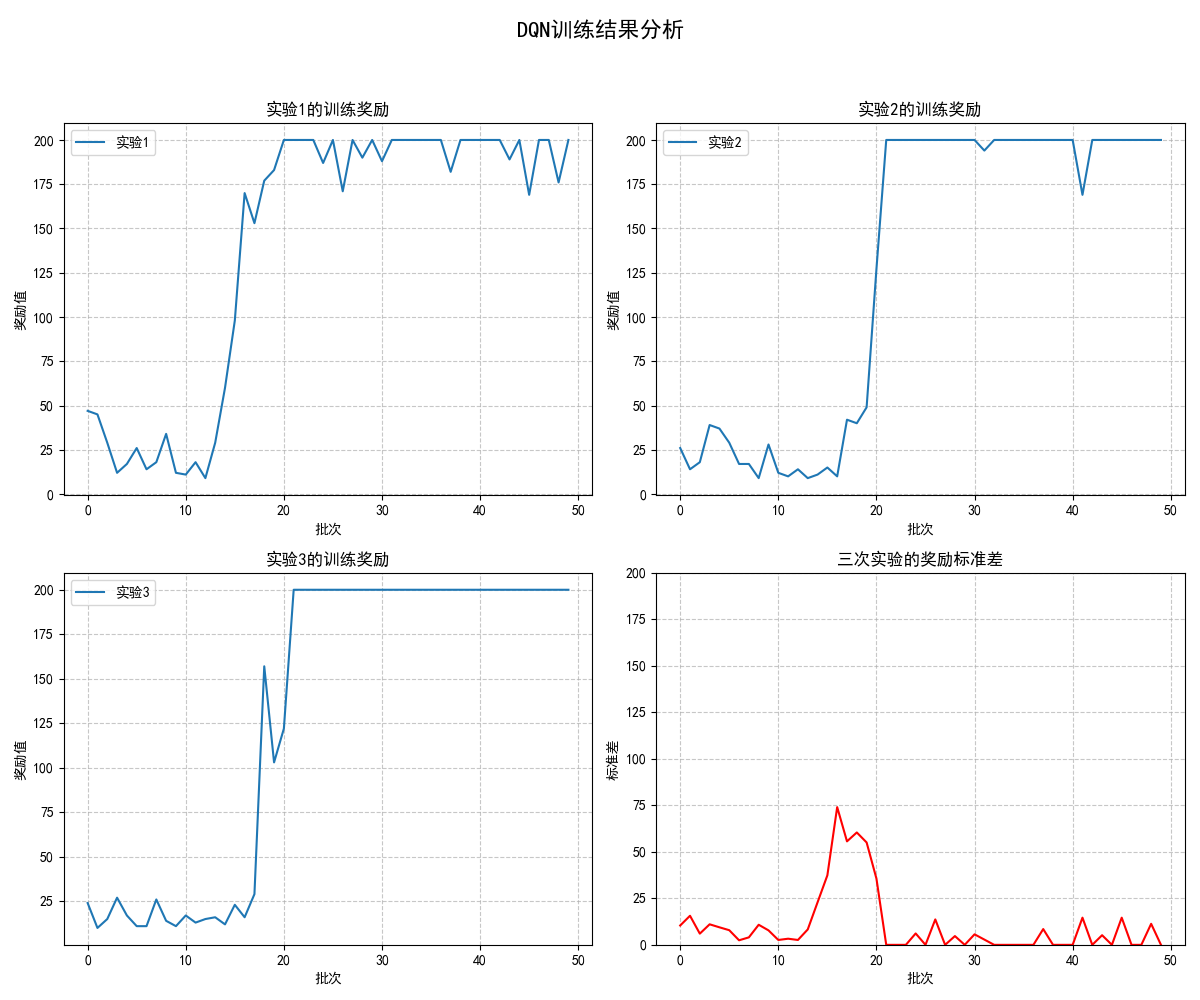
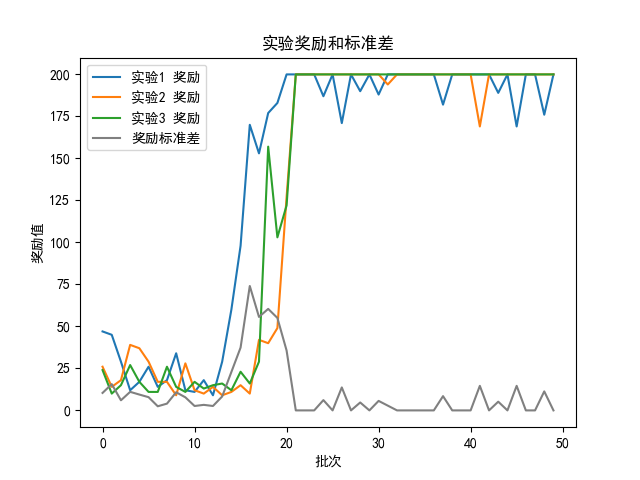
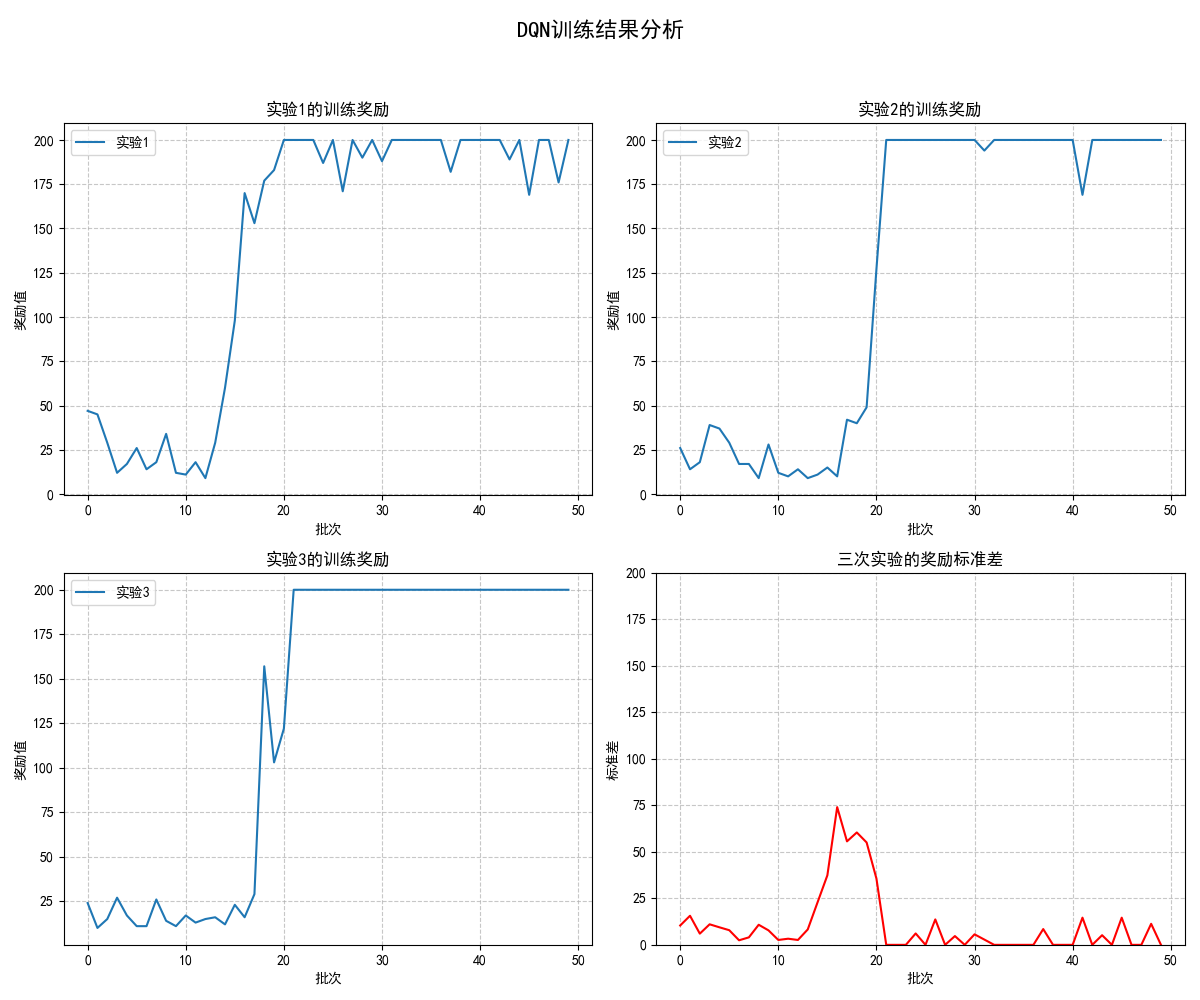
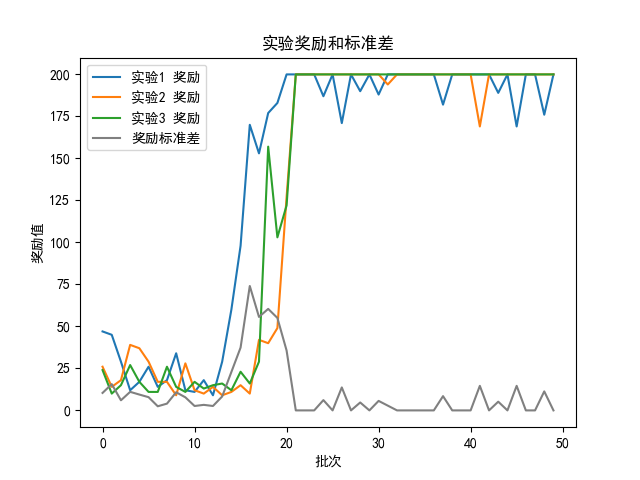
根据衰减因子更新优化器的学习率

1. 创新点&优化（如果有）

* 学习率衰减：实现了学习率衰减机制，有助于在训练后期进行更精细的优化。
* 随机种子设置：设置了随机种子，确保了实验结果的可重复性。
* 奖励记录和可视化：记录了训练过程中的奖励，并实现了结果的可视化，便于分析和调试。
* 参数化配置：使用argparse库来解析命令行参数，使得实验配置灵活且易于调整。

# 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）  


2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

**可以看到实验1-3结果较为一致**

* **收敛的reward大小：实验的奖励值最终稳定在200。**
* **收敛所需的样本数量：实验在大约第20个批次后开始显著提高，并在第30个批次左右达到稳定状态。**
* **三次实验的奖励标准差：从标准差图表可以看出，实验2在中期有较大的波动，这可能表明算法在该阶段的稳定性稍差。实验1和实验3的标准差相对较小，表明算法在这些实验中的表现更为稳定。**
* **总体评估**

**算法性能：所有实验的算法性能都很好，奖励值都达到了180以上。且达到了最大奖励200。**

**收敛速度：所有实验都在大约30个批次后收敛，收敛速度较快。**

**稳定性：实验1和实验3的表现更为稳定，实验2在中期虽然有较大的波动，但是从三次实验的奖励标准差来看，实验较为稳定**

# 参考资料

**1.** **部分** **debug ，代码优化建议参考了ai 大模型的建议。**

2. **阅读参考了以下文章：**

# [深度强化学习——DQN算法原理-CSDN博客](https://blog.csdn.net/weixin_44732379/article/details/127821138?ops_request_misc=%7B%22request%5Fid%22%3A%224aac37ab223e0f3fb7336c1d508d3c7a%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=4aac37ab223e0f3fb7336c1d508d3c7a&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-127821138-null-null.142^v102^pc_search_result_base4&utm_term=DQN&spm=1018.2226.3001.4187) [强化学习4：DQN 算法-CSDN博客](https://blog.csdn.net/White_Ink_/article/details/139114381?ops_request_misc=&request_id=&biz_id=102&utm_term=DQN&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-139114381.142^v102^pc_search_result_base4&spm=1018.2226.3001.4187)[通俗讲解深度强化学习经典算法——DQN\_dqn算法-CSDN博客](https://blog.csdn.net/lingdulunkuo/article/details/141285239?ops_request_misc=%7B%22request%5Fid%22%3A%224aac37ab223e0f3fb7336c1d508d3c7a%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=4aac37ab223e0f3fb7336c1d508d3c7a&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-4-141285239-null-null.142^v102^pc_search_result_base4&utm_term=DQN&spm=1018.2226.3001.4187)