Project：基于神经网络的MDP建模与多种动态规划算法比较实现

项目目标

实现一个用MLP表示离散MDP的框架，并比较不同动态规划算法在策略评估和策略优化中的性能差异。

实现要求

1. 神经网络表示MDP (MLP-MDP)

实现思路：

- 设计MLP网络结构：输入=状态s和动作a的one-hot编码，输出=所有s'的概率分布和reward
- 使用固定随机种子初始化权重保证可复现性
- 实现约束：转移概率矩阵需满足$\sum_{s'} P(s'|s,a) = 1$
- 示例：构建5状态3动作的MDP，可视化网络结构

测试分析：

- 检查转移概率的合法性
- 对比不同网络深度对MDP建模的影响

2. 同步策略评估 (Synchronous Policy Evaluation)

实现思路：

- 输入：随机策略$\pi(a|s)$
- 同步备份：$V_{k+1}(s) = \sum_a \pi(a|s)[R_s^a + \gamma \sum_{s'} P(s'|s,a)V_k(s')]$
- 使用L_2范数收敛判断(e.g., $\epsilon = 10^{-5}$)

测试实例：

- 通过测试实例比较不同γ的收敛速度
- 对比均匀随机策略与偏向策略的评估效率

3. 策略迭代 (Policy Iteration)

实现思路:

1. 策略评估阶段：同步备份直到收敛
2. 策略改进：greedy $\pi'(s) = \arg\max_a Q_\pi(s,a)$
3. 可视化策略变化过程

测试分析:

- 记录每次迭代的策略改进幅度
- 比较不同初始策略的收敛路径

4. 同步值迭代 (Synchronous Value Iteration)

实现思路:

- 直接优化值函数：$V_{k+1}(s) = \max_a[R_s^a + \gamma \sum_{s'} P(s'|s,a)V_k(s')]$
- 与策略迭代对比计算效率

测试实例:

- 通过测试实例分析其收敛过程

5. 异步就地值迭代 (In-Place Value Iteration)

实现思路:

- 状态更新顺序：随机遍历/按序遍历
- 立即覆盖旧值：$V(s) \leftarrow \max_a[R_s^a + \gamma \sum_{s'} P(s'|s,a)V(s')]$

测试分析:

- 记录单次扫描的状态值变化
- 比较不同遍历顺序的收敛性

6. 优先扫描值迭代 (Prioritized Sweeping)

实现思路:

- 维护优先队列：按Bellman error排序
- 只更新误差大于阈值的状态

- 动态调整阈值：e.g.，$\epsilon \leftarrow \epsilon/2$

测试实例：

- 通过实例观察和分析收敛速度
- 对比与前面两种value iteration的更新次数

7. 随机采样值迭代 (Sample-Based Value Iteration)

实现思路：

- 构建一个神经网络近似值函数$V_\theta(s)$
- 基于批量随机采样进行值迭代：$V_\theta(s) = \max_a[R_s^a + \gamma \sum_{s'} P(s'|s,a)V_\theta(s')]$
- 基于更新后的值函数对近似值函数神经网络进行参数更新

测试分析：

- 通过实例分析收敛性
- 比较分析随机采样与精确方法的差异

8. 对比分析前述3-7的迭代方法

1. 收敛速度对比：迭代次数-误差曲线图
2. 计算复杂度分析：单次迭代时间成本
3. 内存需求比较：各算法的存储开销
4. 策略质量评估：最终策略在测试环境的回报
5. 超参数敏感性：γ, ϵ 等参数的影响

交付内容

1. 完整Python实现代码（Jupyter Notebook格式）
2. 测试用的MDP实例
3. 实验报告（含实现思路、可视化图表和结果分析）