

# Project 9 实验与工程进展报告（零基础友好版）

写给：未来第一次接触这个仓库、也第一次接触“逆向强化学习”的同学。

目标：看完之后，你应该能用自己的话说出：

- 1) 这个项目想干什么；
- 2) 目前代码已经做到哪里了；
- 3) 这些图和数字大概说明了什么；
- 4) 接下来如果继续做，还要补哪些环节。

本报告是在之前的“工程框架版”进展报告基础上，用更易懂的语言重写而成。

整体风格参考了《Project9 执行计划\_零基础入门版》。

## 1. 这个项目到底在干什么？（一句话版）

我们在做一件事：

看“菜鸟 AI”是怎么学会玩游戏的，然后反推这个游戏背后的“得分规则”。

这件事在机器学习里叫做 **逆向强化学习 (IRL)**。不同的是，我们看的是**多人游戏**，而且这些“菜鸟”不是老式的“背表格”学习者，而是现代的**策略梯度 / PPO 学习者**。

为了解释清楚问题，我们在执行计划里用了两个比喻：

- **老方法 MA-LfL = “书呆子侦探”：**  
它以为所有学习者都在背一个“Q 表格”，适合很小的离散环境，一到真实的大型游戏就会崩掉。
- **新方法 I-LOLA = “懂水床物理的聪明侦探”：**  
它假定玩家用的是策略梯度，而且考虑到对手也在同时学习，用了一套更聪明的“水床思维”来推奖励。

我们的实验设计，就是想在三个难度递增的环境里，**系统比较老侦探和新侦探的表现**：

- **T1：小迷宫 GridWorld** —— 完全匹配老方法的理想世界；
- **T2：MPE simple\_spread** —— 稍大一点的合作任务，用 PPO 学习，开始“折腾”老方法；
- **T3：MultiWalker** —— 连续动作的物理环境，老方法直接“进不来考场”，只能看新方法的表现。

## 2. 现在仓库已经能做什么？（三关通关情况）

### 2.1 T1：小迷宫 —— “老侦探”和“新侦探”都考了高分

这里的环境是一个 (3\(\times\)3) 的小格子世界，只有两个智能体，动作离散，完全符合 MA-LfL 论文里的设定。

当前代码已经能完成这样一条闭环：

#### 1. 生成数据：菜鸟走迷宫的录像

- 用 `data_gen.ma_spi.gridworld_sampler` 生成多阶段的 MA-SPI 轨迹；
- 数据保存在 `outputs/data/t1/...pk1`。

## 2. 老侦探 MA-LfL 出场

- 用 `python -m runners.run_ma_lfl --config configs/t1_gridworld.yaml`；
- 程序会算出两个数字：`err_1a` 和 `err_1b`。
- 这两个值大约都是 ( $9 \times 10^{-6}$ )，也就是“差异只有百万分之几”。

可以把它粗略理解成：

我们拿“真奖励”让模型预测一次策略变化，得到一个误差 (1a)；

再拿“侦探推出来的奖励”预测一次，得到另一个误差 (1b)；

最后看 1b 是否只比 1a 稍微大一点点。

在 T1 里，1a 和 1b 几乎一样，说明老侦探在自己擅长的小世界里工作得很好。

## 3. 奖励热力图：看到“整形”的影子

运行 `scripts/run_phase3_plots_reports.py` 后，会自动画出 T1 的两张图：

- KL 柱状图：**两根柱子都非常矮，表示 1a 和 1b 的 KL 都很小；
- 奖励热力图：**
  - 上面一条是“真奖励”：只有目标格子是 1，其它都是 0；
  - 下面一条是“恢复奖励”：呈现出一种“从某一侧逐渐升高”的形状。

这说明了执行计划里说的那个现象：**奖励整形**。

虽然恢复出来的奖励在数值上和真奖励差很远，但它诱导的策略几乎是一样的，所以 1a/1b 的 KL 仍然很小。

## 4. I-LOGEL / I-LOLA 在 T1 上的 sanity check

仓库里还有一些脚本会在 T1 上测试 I-LOGEL 和 I-LOLA（例如 `scripts/run_t1_ilogel_sanity.py`），主要是用来确认：

- 在**合成数据**上，恢复的奖励方向和真值的方向余弦相似度接近 1；
- 在**真实 MA-SPI** 数据上也能跑出合理的损失曲线和权重范数。

这相当于确认：“新侦探”在简单场景下不会出大问题。

### 小结 (T1) :

- 在完全匹配的世界里，老侦探 MA-LfL 的指标 1a/1b 都很小，说明它在“小学考场”表现优秀；
- 新侦探在这里也能跑通，但 T1 更像是一个“对照组”：用来证明老方法在“理想假设”下是OK的，而不是一无是处。

## 2.2 T2：MPE simple\_spread —— 新侦探在“真实世界”能工作

T2 是三个小球合作去覆盖三个地标的任务。我们在这里用的是 **PPO 学习者**，也就是执行计划里说的那种“爬山”风格学习者，而不是老方法假定的“背 Q 表”的 MA-SPI。

当前代码在 T2 上已经能做到：

### 1. 用 PPO 生成真实“菜鸟成长”轨迹

- 使用 `python -m runners.gen_data --config configs/t2_mpe_simple_spread.yaml --seed 0`；
- 自己实现的 PPO 会跑若干次更新，把每次更新前后的策略参数以及采样到的轨迹保存下来；

- 数据保存在 `outputs/data/t2/...pk1`。

## 2. I-LOLA 阶段 B (黑盒优化) 的主流程

运行 `python scripts/run_t2_ilogel_eval.py` 会做几件事：

- 用 CMA-ES 这种“聪明的猜测机”去搜索一组奖励权重 `w_hat`；
- 让 I-LOLA 模拟器用 `w_hat` 去“预测”策略如何更新；
- 计算 1a / 1b：
  - 1a 是“拿真奖励喂给模拟器”时的误差；
  - 1b 是“拿侦探找到的奖励喂给模拟器”时的误差；
- 目前 1a 和 1b 大约都在  $10^{-6}$  量级，差距不大。

直观理解就是：

**在 PPO 数据上，I-LOLA 这个“新侦探”可以把“未来策略”预测得相当准。**

## 3. T2 的 induced 训练 (指标 2 的最小版本)

- `evaluation.induced_train` 里已经有 T2 的最小实现版本；
- 运行 `python scripts/run_t2_induced.py` 会：
  - 估计随机策略在环境中的平均回报 `R_random`；
  - 用恢复的奖励 `w_hat` 训练一个新的 PPO 学习者，记录每几个训练阶段的平均回报，形成一条 `R_induced_curve`；
  - 结果输出到 `outputs/induced/t2_induced_seed0.json`，并画出回报曲线。

当前的曲线总体上略好于随机策略，有一定波动。

这说明：**用恢复的奖励重新训练出来的策略确实有一点用，但表现还不算特别亮眼**，后面还需要调特征、超参数，或者增加训练轮数。

## 4. 自动图表和报告

- `scripts/run_phase3_plots_reports.py` 会画出 T2 的 KL 柱状图；
- 并生成 `outputs/reports/report_t2_seed0.md`，自动写入关键数字和图片路径。

**小结 (T2) :**

- 在“真实世界”的 PPO 数据上，新侦探 I-LOLA 已经能跑通，而且指标 1a/1b 非常小；
- 我们已经有了一个最小版的“指标 2”（诱导训练）；
- 缺口在于：老侦探 MA-LfL 还没有在 T2 上跑出完整的结果，所以“老侦探 vs 新侦探”的正面 PK 还没有真正上演。

## 2.3 T3：MultiWalker —— 连续动作的博士级考场，新侦探可行

T3 是最难的考场：三个“机器人大长腿”一起抬着一个箱子往前走。动作是连续的，“加多少牛的力”这样的控制，老方法 MA-LfL 直接没有办法用 Q 表格来表示。

当前代码在 T3 上已经做到：

### 1. 用 PPO 训练 MultiWalker

- 通过 `python -m runners.gen_data --config configs/t3_multiwalker.yaml --seed 0` ;
- 得到多个更新阶段的策略参数和轨迹，存到 `outputs/data/t3/...pk1`。

## 2. I-LOLA 阶段 B + Monte Carlo KL

- 运行 `python scripts/run_t3_ilola_eval.py` 会：
  - 用 CMA-ES 在 T3 上搜索奖励权重 `w_hat`；
  - 用采样的方式近似计算 1b 对应的 KL（因为动作是连续的，只能用 Monte Carlo 估计）；
  - 输出一个 `t3_ilola_seed0.json`，里面记录了：
    - `omega_norm` (奖励权重范数)；
    - `err_1b` 大约在 (5\times 10^{-4}) 左右；
    - 还有一个调试信息，显示某个状态下样本 KL 在 0.003 左右。

这些数字的含义是：

**在这么复杂的环境里，我们至少已经能用 I-LOLA 这台“飞行模拟器 + 猜翼型”的机器，跑出一个稳定的奖励，并给出一个合理的预测误差。**

## 3. T3 的 induced 训练

- `python scripts/run_t3_induced.py` 会：
  - 估计随机策略在 T3 环境里的平均回报 `R_random`；
  - 用恢复的奖励 `w_hat` 再训练一个新策略，记录 `R_induced_curve`；
  - 画出一条“训练步数 → 平均回报”的曲线，保存在 `outputs/plots/t3_induced_returns_seed0.png`。

当前的曲线大致在随机基线附近略有起伏，有时略好，有时接近。

这说明：**在 T3 上，I-LOLA 的诱导策略已经能跑起来，但要真正压过随机策略，还需要更多工程优化（特征、训练轮数、探索策略等）。**

## 4. 自动报告

- `run_phase3_plots_reports.py` 现在也会为 T3 生成 `report_t3_seed0.md` 和 KL 图像。

**小结 (T3) :**

- 在连续动作、多人协作的高级环境里，老侦探 MA-LfL 完全无法出场；
- 新侦探 I-LOLA 已经能完成“奖励恢复 + KL 评估 + 诱导训练”的完整闭环；
- 当前指标说明“可行，但还不够强”，后续可以在这里做更深入的调参和扩展实验。

## 3. 这一轮工程实现，整体意味着什么？

如果把执行计划看成一座大楼，现在的状态大致是：地基和主体结构已经完工，水电都接上了，灯也能亮，电梯能运行；真正还没做的是“精装修”和“对外展示用的海报”。

用更学术一点的说法：

### 1. 工程框架已完成

- 三个环境 (T1/T2/T3) 的数据生成、奖励恢复 (I-LOGEL / I-LOLA) 、指标计算 (1a/1b) 、诱导训练、图表绘制、Markdown 报告，都已经有可运行的代码路径。

### 2. 核心创新 I-LOLA 已在 T2 / T3 上证明确实可以工作

- 在与 PPO 相关的环境上，我们可以用黑盒优化 + LOLA 模拟器恢复奖励，并得到小的 KL 误差；
- 诱导训练也能跑出合理的回报曲线。

### 3. 执行计划中的一些“关键对比实验”还没有完全做完

- T2 上 MA-LfL vs I-LOLA 的正面较量；
- Stage A (I-LOGEL) vs Stage B (I-LOLA) 的系统对比；
- 多个随机种子上的统计结果；
- 扩展版实验 (T2.5) 和更强的诱导策略表现。

换句话说：

“工具箱”已经搭好，工具能用；

剩下的主要是——设计更完整的“实验故事”，把老侦探、新侦探在各种考场上的表现讲清楚，并用多次重复实验把结论坐实。

## 4. 如果要继续做，这个项目还需要哪些工作？（面向大一学生的 checklist）

下面这份 checklist 是给“后续开发者”的路线图。你可以按顺序完成，也可以根据时间挑重点。

### 4.1 任务一：让老侦探 MA-LfL 也来 T2 参加比赛

**为什么要做？**

执行计划里最关键的故事之一是：

“当学习者不再是 MA-SPI，而是 PPO 时，老侦探会明显考砸，新侦探能顶住。”

要讲清楚这个故事，必须让 MA-LfL 在 T2 上有一套完整的结果，然后和 I-LOLA 放在一张图里比较。

**大致步骤：**

1. 在 `runners.run_ma_lfl` 里，新增对 `env=mpe_simple_spread` 的支持：
  - 把 T1 用到的逻辑搬过来；
  - 改成从 `outputs/data/t2/...pk1` 读 PPO 生成的数据；
  - 即使假设不完全正确也没有关系，反而正好体现“模型失配”。
2. 调用 `evaluation.metrics` 计算 T2 上 MA-LfL 的 1a / 1b，写入 `t2_ma_lfl_seed0.json`。
3. 在 `evaluation.plots` 写一个专门画 T2 对比图的函数：
  - 比如画四根柱子：`LFL-1a`, `LFL-1b`, `ILOLA-1a`, `ILOLA-1b`。
4. 在 `report_t2_seed0.md` 里加入一段文字解释：

- T2 使用的是 PPO 数据；
  - MA-LfL 的理论假设与真实学习过程不符；
  - 因此我们预期它在 1b 上会比 I-LOLA 差很多。
- 

## 4.2 任务二：比较 Stage A (I-LOGEL) 和 Stage B (I-LOLA)

为什么要做？

在执行计划中，我们把 Stage A 和 Stage B 看作“基础版侦探”和“完全版侦探”。

理论上 Stage B 会更好，但需要用实验来支撑这一点。

大致步骤：

1. 写一个类似 `scripts/run_t2_stageA_eval.py` 的脚本：
    - 调用 `inverse.ilola.stage_a_independent` 在 T2 数据上跑一次；
    - 得到 `w_hat_stageA`；
    - 用和 Stage B 相同的 metrics 函数计算 1a / 1b；
    - 写入 `t2_stageA_seed0.json`。
  2. 在 `evaluation.plots` 画一个“Stage A vs Stage B”的简单对比图：
    - 至少比较 1b；
    - 如果同时有 MA-LfL，就画三组柱子：MA-LfL, Stage A, Stage B。
  3. 在 T2 报告里加一小节文字：
    - 解释 Stage A 假设对手不动，Stage B 会把对手的学习也考虑进去；
    - 用数字说明 Stage B 的误差确实更小。
- 

## 4.3 任务三：多随机种子，多次重复实验

为什么要做？

现在所有结果都是 seed 0，只能说明“这一次”长这样。

如果要写论文或者严肃报告，需要看**多次重复后的平均表现**。

大致步骤：

1. 写几个简单脚本，例如：
  - `scripts/run_all_t1_seeds.py`
  - `scripts/run_all_t2_seeds.py`
  - `scripts/run_all_t3_seeds.py`
- 这些脚本做的事就是：  
换不同的 `--seed`，循环调用已经有的命令（生成数据、跑 I-LOLA、跑 MA-LfL 等）。
2. 写一个 `scripts/aggregate_metrics.py`：

- 扫描 `outputs/metrics` 目录；
  - 把同一实验、不同 seed 的 `err_1a` / `err_1b` 聚合到一个表里；
  - 算出均值和标准差，保存成 `.csv`。
3. 在 `evaluation.plots` 里增加画“带误差条的柱状图”的小函数，展示：
- 比如：T2 上 MA-LfL, Stage A, Stage B 在多 seed 下的平均 1b 和标准差。
4. 在报告中新增一节“Across seeds”的小结，用自然语言解释这些误差条的含义。
- 

## 4.4 任务四：扩展实验 T2.5（可选加分项）

**这个部分不是必须，但如果有时间做，会让故事更完整。**

大致思路：在 MPE 的基础上稍微改造一下环境（例如多几个 landmark，或者改变观测方式），形成一个“比 T2 稍难”的新测试场 T2.5。

复用 T2 的代码路径，只是换一个 `config` 文件，然后看看：

- I-LOLA 在 T2.5 上的 1a / 1b 是不是仍然保持较小；
  - 诱导训练曲线是否比随机策略好更多一些。
- 

## 4.5 任务五：整理面向读者的最终文档

你现在看到的这份报告，主要是面向“项目内部”和“大一级别读者”的。

如果要面向助教或评审，还建议做三件事：

### 1. 再精简一份 README

- 用“从零跑到主要结果”的命令序列，把最小示例写清楚；
- 例如只保留：T1 + T2 的一条完整 pipeline。

### 2. 写一份“主报告”

- 在 `outputs/reports/` 下新建 `project9_main_report.md`；
- 把 T1/T2/T3 的主要数字和图表链接整合到一份文档里，形成一篇“论文风格”的稿子。

### 3. 简单的环境复现说明

- `check_env.py` 已经有了；
  - 可以在 README 里加一小节“环境搭建与一键脚本”。
- 

## 5. 总结：如果你是接手这个项目的人

如果你是一个刚接手这个仓库的大一学生，可以把当前状态理解成：

1. 代码骨架已经搭好，三大考场都能跑起来；
2. 新算法 I-LOLA 在 T2/T3 上已经有了“能工作”的证据；

3. 老算法 MA-LfL 的“失败故事”和 Stage A vs Stage B 的“对比故事”还需要你来补完；
  4. 剩下的工作更多是“补实验、画图、写故事”，而不是“从头写一堆新代码”。
- 如果你只想做一个“最小可运行示例”，可以按下面这样的顺序来熟悉项目：
1. 跑 T1: `gen_data` → `run_ma_lfl` → `run_phase3_plots_reports.py`，看看 KL 图和奖励热力图；
  2. 跑 T2: `gen_data` → `run_t2_ilogel_eval.py` → `run_t2_induced.py` → `run_phase3_plots_reports.py`；
  3. 跑 T3: `gen_data` → `run_t3_ilola_eval.py` → `run_t3_induced.py`。

当你能解释清楚这些命令各自做了什么、输出的数字和图表分别是什么意思时，你就已经非常接近完全掌握 Project 9 了。