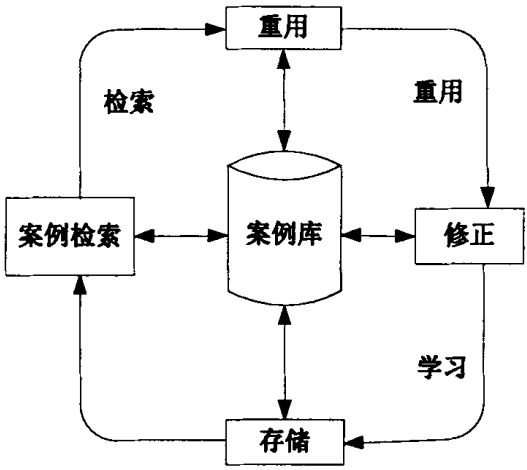
CBR+PGPE实验报告

**一、基于案例推理的方法简述**

基于案例推理(case-based reasoning)是一种基于现有知识的归纳从而产生新的知识的过程，其流程包括：生成案例库、选择案例、动作的适应、案例库的更新。案例的形式为：，其中P是指案例的条件，A是指结论。与规则的定义类似；选择案例是选择与目标任务最相近的案例，通过定义相似度函数的方式实现，相似度函数一般选择欧式距离；动作的适应是因为案例形成的动作往往不能直接应用于新的环境，需要对动作进行一定的改进，这是与强化学习结合的关键步骤；最后是案例库的更新，可以分为静态更新或者动态更新。基于案例推理的形式如图一所示。



图**1.** 基于案例推理的流程图

**二、场景和方法的选择**

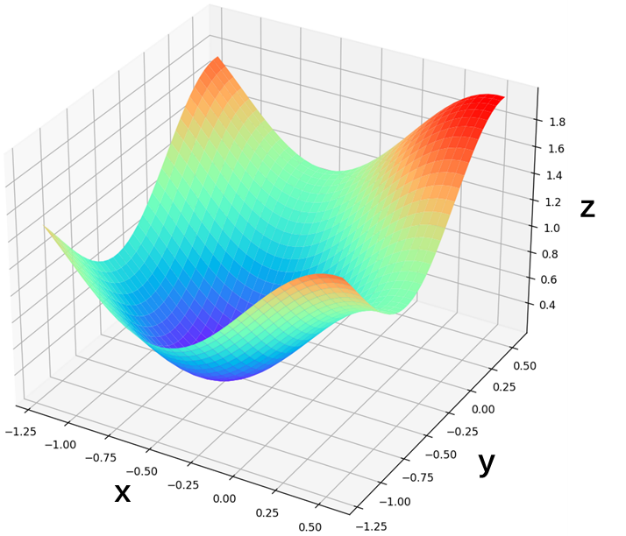
我选择了复现相关论文中mountain-car从二维到三维的场景作为验证算法的demo，如图2所示。在具体算法的选择上，由于很多文章已经研究过CBR+Qlearning了，为了进一步提高算法效率，我选择探究CBR+ rule-based PGPE的方法。

在三维mountain-car场景中，智能体初始处于的最低点，目标是通过自身的运动到达的区域中，智能体的动作选择是离散的，有：north/west/east/south/neutral五个动作选项，其中north/south是沿着y轴的正、负方向施加加速度；east/west是沿x轴的正、负方向施加加速度；neutral是不施加加速度。

与二维的mountain-car场景相比，三维的场景状态空间和动作空间的维数完全不同，探索的难度大大增加，使用其作为我们推理算法的验证场景是十分适合的。

PGPE算法的全称是基于参数探索的策略梯度算法，是策略梯度算法的一种改进，将对参数求取梯度更改为对参数的分布求取梯度，这样避免了当动作策略不可导时，无法使用策略梯度算法的问题。因此，有了这种算法，我们在设计动作策略时就可以使用更加直观的规则定义方式，这就是rule-based PGPE，在之前的demo中我们已经征明了这种算法的高效性，与Q-learning这种经典的算法相比，效率提高了50%以上，但是代价就是需要设计符合实际的动作策略规则。

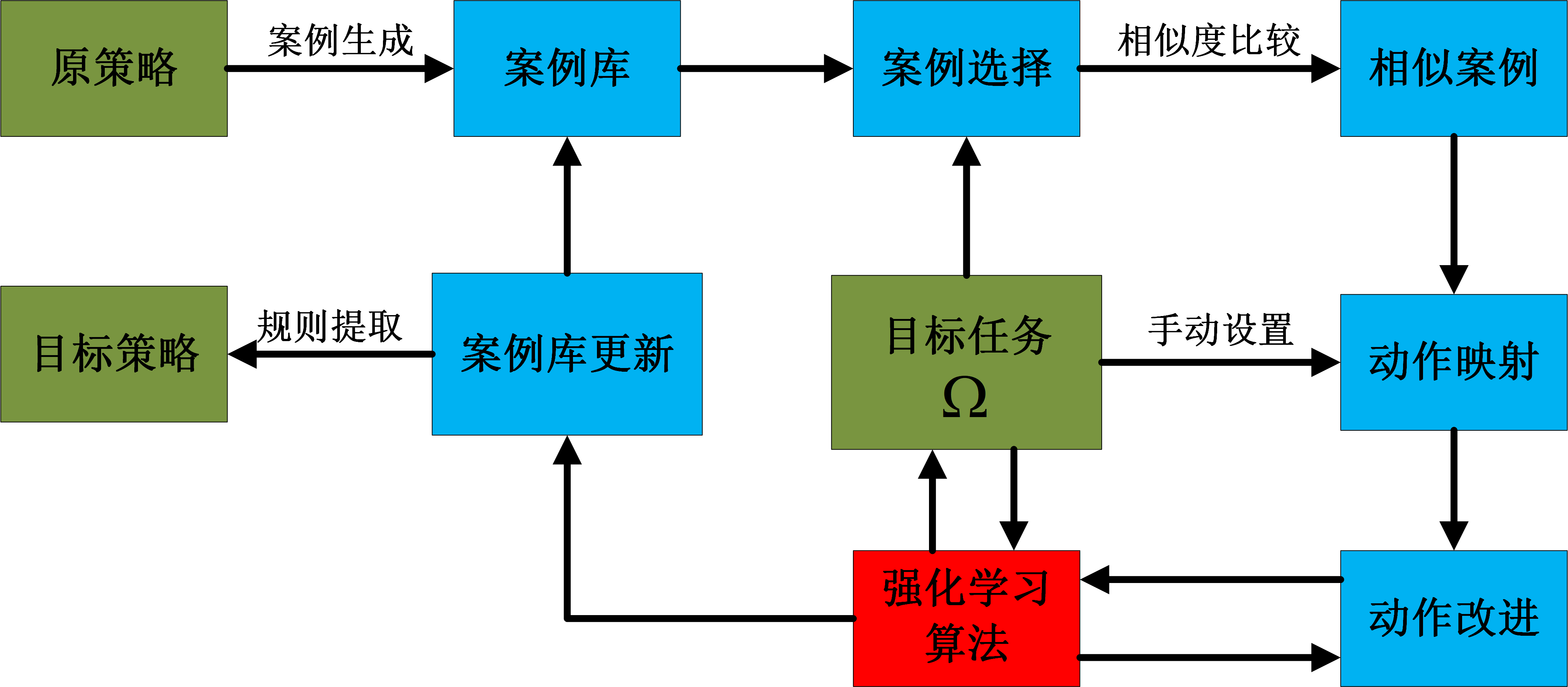
在本例中，我们首先使用rule-based PGPE算法训练mountaincar 2D场景中的智能体策略，然后使用基于案例推理的方法，将原策略扩展到mountaincar 3D场景中，并生成符合新环境的新动作策略。



图**2.** 三维mountain-car场景示意

**三、方法探析**

本例采取的方法流程图如下，下文会做具体说明。



1、案例库的生成：

在原mountain-car 2D环境中训练一个策略，原策略如下：

**if velocity >= 0: # velocity>=0 表示车辆正在向右移动**

**if (position > -0.5) and (abs(velocity) < abs(theta)): # 当到达右侧一定位置无法向右时, 则向左**

**action = 0**

**else:**

**action = 2**

**else: # velocity<0 表示车辆正在向左移动**

**if (position < beta) and (abs(velocity) < abs(theta)): # 当到达左侧一定位置无法向左时, 则向右**

**action = 2**

**else:**

**action = 0**

其中参数theta和beta为探索对象，经过1000回合探索、50次迭代，参数更新为：theta=0.018433,beta=-0.63454。

然后执行若干回合，每一时间步都取状态-动作对：作为一个案例。最后把状态过于接近的案例合并，大约保留600~700个案例构成案例库。

2、案例的选择：

在mountain-car 3D环境中，可取的状态参量为：，与原环境不同，因此需要构建新老环境之间的相似度函数：。我们选择相似度最高、即相似度函数最小的若干案例作为我们的参考案例。

3、新环境下动作的映射：

新环境中动作有5维，原环境中只有3维，因此案例的动作需要经过扩维才能应用于新环境。文献中使用的方法是神经网络，这里我直接使用了结论，动作对应关系如下：

|  |  |
| --- | --- |
| 三维 | 二维 |
| 无动作 | 无动作 |
| 向北 | 向右 |
| 向东 | 向右 |
| 向南 | 向左 |
| 向西 | 向左 |

4、基于随机探索方式进行规则的扩展（动作的适应）：

为了使策略在新环境中不断改进，需要设置动作规则与PGPE探索参数，目前这依赖于人手动设置。但是由于结合了案例库，可以降低设置新规则的难度。我设置的初始规则如下：

**当时，小车在west/east/neutral之间选择；否则小车在north/south/neutral之间选择。**

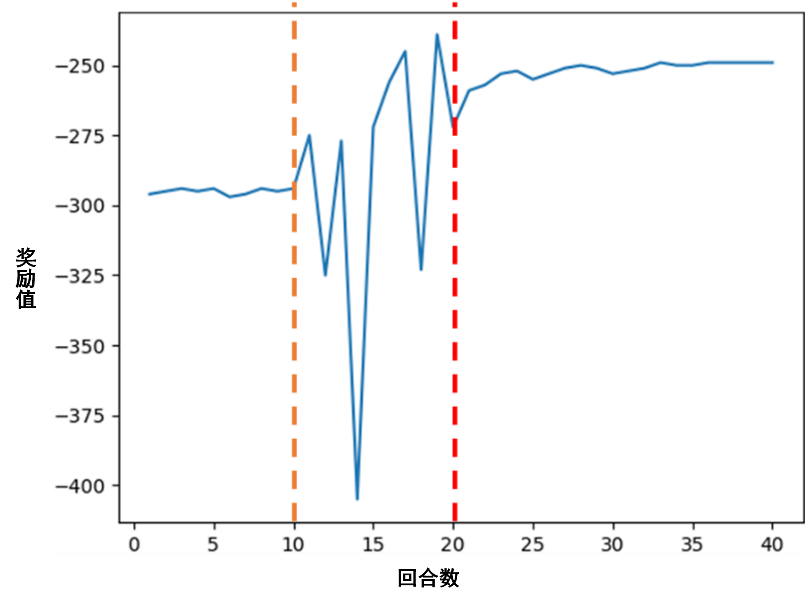
这条初始规则可以使得智能体完成任务并且可以使用PGPE强化学习算法进行优化，但是由于规则太少，虽然有案例库作为参考，但是很难达到理想的效果。由于PGPE算法完全依赖参数的探索，因此需要想办法生成新的参数，而环境的复杂性使人很难在短时间内得到新的规则。

对于RL算法来说，优化的核心就是探索，只有对环境了解地越充分，才越有可能找到更优解。单一规则限制了参数的探索范围，如果暂时想不到新规则，可以让智能体在一定范围内进行随机探索，这种探索本质上是改变物体的状态，达到一些原本达不到的状态。如果探索的结果优秀，就可以据此生成新的规则和选定新的参数：规则就是状态的变化趋势；参数就是状态的改变量。

在本例中，选择改变小车初始时的位置进行随机探索：选择随机生成初始时小车y方向的位置，其余沿用原策略并观察结果，然后选择奖励值最高的位置生成新的规则，新规则如下：

**当时，小车首先朝所在方向运动；当时，小车执行原规则。**

仿真结果如图3所示，前10回合是只有一条初始规则时探索的结果；中间10回合是随机改变小车初始时的位置探索的过程；最后20回合是选择奖励值最高的位置生成新的规则后探索的结果。可以看出，规则数量的增加可以明显改善奖励值的质量。



图**3.** CBR+rule-based PGPE算法在mountain-car 3D环境中得到的结果

5、案例库的更新与扩展：

前述提到更新的方式有静态更新或者动态更新。其中静态更新是指在新的策略训练完成后再生成一个案例库；动态更新是指在策略训练的过程中不断地将成熟的案例加入到库中。

本例中采用的是静态更新。这是因为新场景和原场景是两种完全不同的环境，状态、动作维数均不相同；如果新场景与原场景差别不大，可以考虑动态更新案例库。

6、新场景下动作策略的生成

使用更新后的案例库可以较好地适应新环境，但是由于存在案例库，需要较大的存储空间，查找搜索案例库的过程也会造成动作执行的延迟。更重要的是，我们使用推理的目的是生成能够适应新环境的动作策略，但是案例库距离成熟的动作策略还有一定的距离，对于我们所采用的rule-based PGPE算法来说，我们期望能够生成成熟的动作策略规则。因此，接下来要进行的工作就是把案例库抽象成规则式策略。

这个过程思路并不复杂，我们在动作的适应阶段已经定义了若干条规则，现在我们要做的是把原本由案例库生成的参考动作转变成规则定义的动作策略。不难发现，当原策略也是由规则定义时，这一步骤容易完成，只要把原策略作为子规则嵌套进新规则中即可；但是当原策略不是由规则定义时，就需要对原案例库进行解析，先生成一个原规则，再把它嵌套进新规则中。

具体到本例，由于原策略即为规则定义的动作策略，故新规则的定义如下：

**if self.flag == 1: # 生成的第二条规则**

**position\_x, velocity\_x, position\_y, velocity\_y = observation**

**if beta < -0.5:**

**if abs(position\_y) < abs(beta):**

**action = 3 + self.flag\_v # 向着目标点beta运动**

**else:**

**action = 2**

**if abs(position\_y) > abs(beta) and abs(velocity\_y) < 0.005:**

**self.flag = 0**

**else:**

**if abs(position\_y) > abs(beta):**

**action = 3 + self.flag\_v # 向着目标点beta运动**

**else:**

**action = 2**

**if abs(position\_y) < abs(beta) and abs(velocity\_y) < 0.005:**

**self.flag = 0**

**if self.flag == 0: #执行原策略**

**position\_x, velocity\_x, position\_y, velocity\_y = observation**

**if position\_x >= theta:**

**if velocity\_y >= 0: # velocity>=0 表示车辆正在向右移动**

**if (position\_y > -0.5) and (abs(velocity\_y) < 0.018433): # 当到达右侧一定位置无法向上时, 则向下**

**action = 3**

**else:**

**action = 4**

**elif velocity\_y < 0: # velocity<0 表示车辆正在向左移动**

**if (position\_y < -0.63454) and (abs(velocity\_y) < 0.018433): # 当到达左侧一定位置无法向下时, 则向上**

**action = 4**

**else:**

**action = 3**

**else:**

**if velocity\_x >= 0: # velocity>=0 表示车辆正在向右移动**

**if (position\_x > -0.5) and (abs(velocity\_x) < 0.018433): # 当到达右侧一定位置无法向右时, 则向左**

**action = 0**

**else:**

**action = 1**

**elif velocity\_x < 0: # velocity<0 表示车辆正在向左移动**

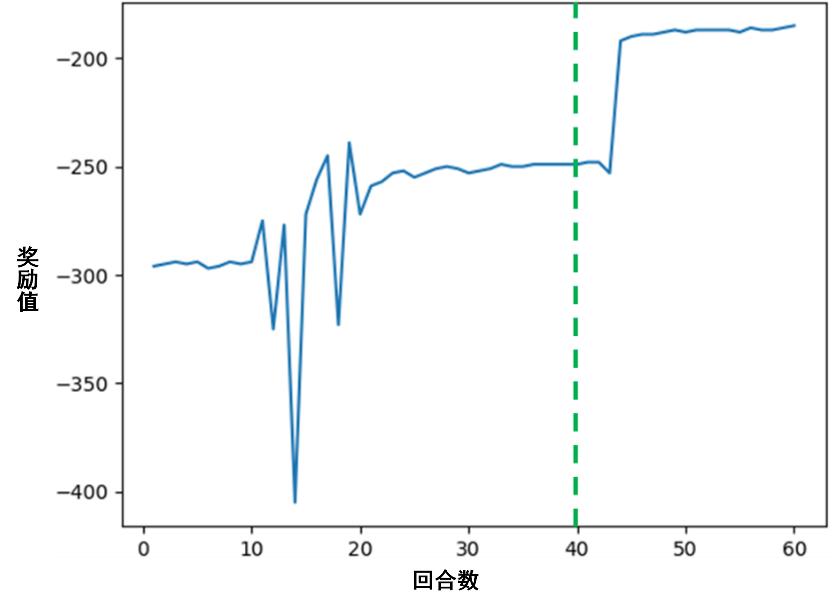
**if (position\_x < -0.63454) and (abs(velocity\_x) < 0.018433): # 当到达左侧一定位置无法向左时, 则向右**

**action = 1**

**else:**

**action = 0**

我们仍然保留了探索的参数，这是因为当不使用案例库后，可能会探索得到新的更优马尔科夫路径，仿真结果也验证了我们的观点。我们在原参数值的基础上继续迭代20次，结果如下图：

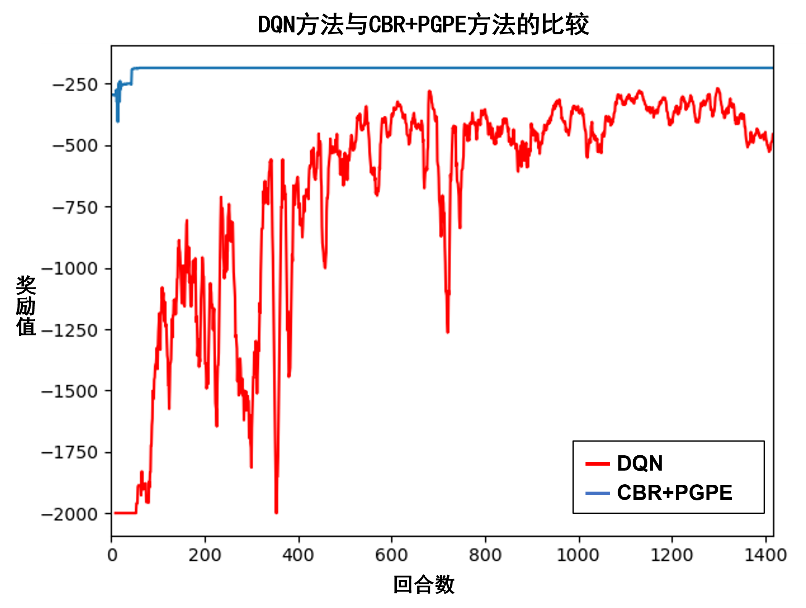


图**4.** 使用rule-based PGPE算法在mountain-car 3D环境中得到的结果

可以发现，在继续迭代3次左右，奖励值又有了较大幅度的提升，这表明规则式动作策略比查阅策略库的方法更加高效，也表明更加灵活的参数探索过程可以发现更优动作策略。

**四、结果的讨论**

我们选择使用Q-learning算法训练智能体，得到的结果与我们本例中采用的算法得到的结果进行对比，结果如图5所示，可以看出，Q-learning算法（红色曲线）在迭代1400回合之后仍然无法达到PGPE算法的最低效果，且奖励值与采用CBR+rule-based PGPE算法的结果（蓝色曲线）相比差距较大。这首先说明了传统的强化学习算法对环境的探索效率较低，需要花费更多的时间和算力与环境进行交互；同时，这个结果也说明基于规则的算法具有很强的鲁棒性，几乎不可能出现完不成任务的情况；最后，其奖励值的对比说明了采用推理可以大幅降低初始时对环境的探索代价。



图**5.** Q-learning与CBR+rule-based PGPE算法对比

对于rule-based PGPE算法来说，其应用的主要难点是如何设置合适的规则与探索参数，特别是当面临一个新的环境条件时，我们提出的CBR+rule-based PGPE方法能够简化规则生成的过程，虽然目前这一过程还较为依赖人的经验，但是已经在生成新的规则方面做出了探索。对于推理来说，达到了生成新知识的目的，并且我们提出的方法能够在推理完成后进一步提高其任务效率和质量，同时还能保持较少的探索次数，达到了技能转移的目的。