Inverse Reward Design——hoho

论文试图解决什么问题

设计的Reward其实仅仅表现的是一种算法设计师基于他自身观测结果的期望，而并非是完全正确的定义，这个结果需要被放在环境的上下文里进行考虑。对于已经设计好的Reward，它们通常在训练环境中非常有效，但是放到新的测试环境中就不一定那么完美了。

论文中提到的解决方案之关键

本文探讨的Inverse Reward Design(以下简称IRD)，是一种类似IRL的方法，它的目标和IRL一致是为了求出最优的Reward。他们之间区别在于IRL是直接观察专家(或用户)的行为，而IRD是观测目前已经由专家设计好的Reward函数，并以此来进一步优化Reward函数本身。

算法过程：

大致分为两步：(1)先执行RL算法,(2)再执行IRL算法。

算法设计师会给出一系列在训练数据集上非常接近True Reward的Reward函数，虽然可能在测试集上不一定合理，但已经表现出了设计者希望机器人如何行动的倾向。所以在这里我们需要添加的其实是机器人面对场景的不确定性。

第一步的RL，是基于给出Reward去求解当前情况下的策略(Policy)，通过该策略来生成专家序列，这里的专家序列其实可以被认为是算法设计师对于场景的设想和偏好。在IRD中如果认为该设计师设计的越好，那么该策略生成的序列就应该越多。

第二步的IRL，在得到专家序列之后，还是需要通过IRL的算法去还原新的Reward，为了去应对环境的不确定性，这里使用了最大熵的IRL方法。最大熵的意义在于对于未知的情况，算法会以最大熵的思想去判别，而不是简单地把未知情况的Reward设置为0。