Robot Learning 2023

02346908

Coursework 2

Description Of Your Implementation

High-level summary

The overall algorithm uses cross-entropy planning in order to calculate a sequence of actions aiming to reach the next milestone (explained later) on route to the goal state. The strategy for model predictive control is closed-loop planning, which replans a new sequence of actions every 2 steps. The algorithm uses a very short planning horizon of 2 steps which has the advantage of being computationally efficient, and leads to the robot making very direct and straight darts even at the initial training stage where the true underlying model dynamics are unknown. The algorithm uses a dense reward function, with slight modifications to the reward depending on the stage of training. In most cases the reward function is the distance from the robot state to the next milestone with a minimisation, however at some point in training when the goal is to refine the best path (explained later), the dense reward function also incorporates path uncertainty in order to encourage exploration to find a more optimal path. In order to avoid the robot getting stuck at a local optimum where the robot traverses through the obstacle (significantly slowing the robot down), the training and testing strategy incorporates 'milestones' which helps guide the robot outside of the obstacle path. The goal of training is thus to refine the milestones which are later used in the testing phase. Initially, the robot does a full outer circle (milestones are the corners) of the environment; this strategy incorporates a known property that the obstacle is always located centrally. At this stage, the planning algorithm uses the distance to the next milestone as the reward function, because at this stage we do not want to encourage path deviation yet. This initial loop is clock-wise, beginning at the start-state, looping around the environment and reaching the goal state, then looping back to the start state; yielding two semi-circular paths from the start-state to the goal-state (goal-to-start path is reversed in order to avoid resetting). Another reward function is used to score the better path; simply by their overall distancer. The key is to then select the better path, and recompute the milestones as every 10th state along the path from start to goal. The objective now is to refine this better path, which is done through rewarding high uncertainty in the dense reward-function, making it an 'intrinsic reward function' encouraging deviation along that general route. As such, in this stage of the training any time a new path from start to goal becomes the best path explored, the milestones are recomputed from the states of that path. At testing time, the robot exploits the best path explored, by planning towards each milestone then finally to the goal state, which results in always getting around the central obstacle.

Interesting findings and ideas

An interesting finding was that my initial hypothesis regarding using a long planning horizon being the best way to allow the robot to avoid getting stuck at a local optimum was completely incorrect. The only way I managed to avoid getting the robot stuck through the obstacle was by incorporating milestones, and moreover, using an extremely short planning horizon did not only cause the robot to move significantly faster due to computational efficiency, but also yielded much more direct and straight darts. Another interesting finding is that the relatively pessimistic approach of initially setting the milestones to the corners, leading to the initial start-to-goal path being quite semi-circular and not direct, was still considerably more reliable across the environments than simply planning from start-to-goal. Furthermore, in theory through refining the milestones through rewarding uncertainty, with a longer training time this approach should eventually shorten the loop and instead find a more direct path that bypasses the obstacle. Another interesting finding was that an intermediate approach was to keep exploring in a loop during the training stage (difference being not to refine the better semi-circle for the remaining training period), and exploit the reduced uncertainty at test time. This strategy actually yielded a path at testing time that was between the two semi-circular routes, which ended up being through the obstacle which was what the entire strategy was designed to avoid.