Own Project Notes

Fang, Thomas, Li Yang

August 27, 2016

1 INTRODUCTION

In our project we apply Reinforcement Learning to let Baxter's right hand to follow the marker which is moved around arbitrarily by a human. Of course this tasks could easily be done by primitive position control, but we thought learning is more fun. What we actually do is to evaluate several learners using this specific problem.

2 EXPERIMENT SET UP

The marker can move on the surface of a table, i.e. a convex 2D space parallel to the x-y-plane of the world frame. The arm can only move in the same x-y-space, but its z coordinate is 30cm higher than the maker (or the table). The learner should be rewarded if the arm is close to marker w.r.t. the x and y coordinates. We use the camera of the right hand to track the marker.

3 Q-LAMBDA

The problems seemed to be small. So, the first thought was to keep it simple and implement a discrete algorithm, we chose Q-Lambda. Just to spoiler the story: We actually never tested Q-Lambda on the robot; everything was software only. But we still spend a lot of time with bug fixing and parameter tuning.
At first we represented the state by the x and y coordinates of the marker and the arm, which is four dimensional. This performed very bad and professor Toussaint had the idea to take only the differences of the x and y coordinates as state information. With this, each dimension becomes twice as large but we only have two dimensions instead of four, which is much

better. The reason why this representation still suites our needs is that it actually does not matter what the exact positions are. For example, if the marker is left to the arm, the arm should move to the left.
After many hours of runtime improvement done, we agreed that it is still too slow. So, we tried something new.

4 Policy Search: Actor Critic

We implemented the basic single step actor critic algorithm. It approximates the policy function with linear regression and uses some information similar to the TD-error. The software implementation was easy and fast. The performance is very good (For concrete results, you can run the algorithm in simulation mode). Therefore we decided to test only this algorithm on the robot.
While trying to do so, we experienced some strange behavior of the maker tracker. The coordinates tracked were always wrong. Even worse, after the arm (camera) moved but the marker did not, it returns a different marker position than before. We made two mistakes: First, there are different marker sized, and you need to manually set the correct size. Second, there are actually two different 'base' coordinate frames and of course we took the wrong one.
After fixing this, we finally tested the actor critic algorithm on the robot, only to figure out that it takes too long to converge because robot movement is slow. Sadly, this the end of the story, we ran out of time.