Bootstrapping Bayesian Inverse Reinforcement Learning in Robotics through VR Demonstration

Reed Sogabe*1 Tomoaki Kimura*2 Masaru Sogabe*3 Dinesh Bahadur Malla*2,3 Katsuyoshi Sakamoto*2 Tomah Sogabe*2,3,4

*1 K.International School Tokyo
*2 Department of Engineering, The University of Electro-Communications
*3 Grid Inc., Tokyo, Japan
*4 i-PERC, The University of Electro-Communications

In traditional reinforcement learning, the exploration has always relied on the use of a single clearly identified reward. However, when this is applied to robotics or real-world tasks, it could prove to be a challenge as the exploration is done in environments where there is a sparse reward. The two main contributing factors behind this is the low probability of the agent encountering the reward when random exploration is done, and the complicated nature of the surroundings in real-world applications. With machine learning already imposing a strong presence in the field of engineering, we turn to robotics to explore how demonstrations could be used to tackle this exploration hurdle. Although the demonstration data, or “teacher data” is commonly placed together with its original data in the action state value function, this could prove to be a problem as it could result in overlearning or under-learning, where the demonstrator’s data plays a much larger role or smaller role than intended. Thus, in order to alleviate this issue, inverse reinforcement learning (IRL) is used to calculate the reward, a hidden feature, from the demonstrator’s behavior trajectory(s,a), which is recorded from a VR embedded Gazebo environment. Meanwhile, in order to further tackle the learning issue, a bootstrapping Bayesian inverse reinforcement learning scheme is proposed to obtain the distribution of reward, instead of the single reward at maximum likelihood.

1. Introduction

Traditional reinforcement learning is difficult to apply to robotics, due to its differences in the way exploration is done. Games typically feature an already specified reward function with simple game mechanics and dynamics (Such as a grid-world). This is not the case with robotics, as its reward is an action, a maneuver, such as the maneuver of an object from a location to another through the action of pushing, generating what is known to be a sparse reward. Hence, it is difficult for the random exploration to be conducted, as the agent would find it difficult to find the reward in a sparse reward environment. With machine learning already imposing a strong presence in the field of engineering, we turn to robotics to explore how demonstrations could be used to tackle this exploration hurdle. Although the demonstration data, or “teacher data” is commonly placed together with its original data in the action state value function, this could prove to be a problem as it could result in overlearning or under-learning, where the demonstrator’s data plays a much larger role or smaller role than intended. Thus, in order to alleviate this issue, inverse reinforcement learning (IRL) is used to calculate the reward, a hidden feature, from the demonstrator’s behavior trajectory(s,a), which is recorded from a VR embedded Gazebo environment. Meanwhile, in order to further tackle the learning issue, a bootstrapping Bayesian inverse reinforcement learning scheme is proposed to obtain the distribution of reward, instead of the single reward at maximum likelihood. A VR headset (such as the HTC Vive) and motion tracking sensors could be used to control the robot to generate demonstration.

2. Related Method

2.1 Reinforcement Learning + Imitation Learning [1]

The aim of Reinforcement learning is for the agent to learn the policy, in order to make decisions. Its primary use is in complex and unfamiliar environments, and through series of trial and error, the agent learns the policy in order to maximize its expected return. After the action is done and the state is given then the expected reward can be written in the form of the following formula:

$$Q^\pi(s_t,a_t) = E_{\pi, s_{t+1} \sim E}[R_t|s_t, a_t]$$

This formula (eq 1) can then be written in a recursive form, in order to find the expected return, or the “average over the future”.

$$Q^\pi(s_t,a_t) = E_{\pi, s_{t+1} \sim E}[r_t + \gamma V(s_{t+1})]$$

And the state value function of the next state is given as

![Fig.1 Conventional learning scheme for integrating the demonstration behavior into the forward reinforcement learning](image-url)

Contact: Tomah Sogabe, i-PERC, The University of Electro-communications, sogabe@uec.ac.jp
This is known as the Bellman equation and allows calculation of the estimated value of $Q^\pi(s_t, a_t)$, or the action-value function. Demonstrations data is traditionally added to the $Q^\pi(s_t, a_t)$ in the Q-table together with the agent data. The epsilon greedy is then used to determine the probability of the agent entering exploration, or exploration. For example, when epsilon greedy ($\varepsilon$-greedy) is set at a value of 0.1 ($\varepsilon = 0.1$) then the agent exploits the best variant 90% of the time and chooses a random variant 10% of the time. This process of integrating demonstration into the reinforcement learning algorithm can be graphically expressed in Fig.1.

2.2 The Neural Network Architecture

Neural network is widely used an approximation and inference model in machine learning, reinforcement learning. In this work all the modeling are based on the neural network. When integrating demonstrations into inverse reinforcement learning requires extensive use and understanding of the neural network architecture. The artificial neural network model was originally inspired by the biological neural networks that we humans, and other biological organisms hold. However, the model used here can be summarized in the following (Fig.2):

![Fig.2 The Neural Network Architecture](image)

The Neural network model has three primary layers, including the input layer, the hidden layer and the output layer. In the input layer, the action value and state value is fed in, as it is the layer that recives external data. The circular artificial neurons that make up the hidden layer ultimately result in an output, which is the output layer. In our case, this is the demonstrator’s reward ($R_d$).

3 Inverse reinforcement learning

The aim of the traditional reinforcement learning is to learn the best possible decision process, or policy in order to obtain the maximum possible, or best possible reward. In contrast, the aim of inverse reinforcement learning is to try to understand the behavior of an expert or demonstrator and use that to extract the reward function. The mathematical formula is similar to the normal forward reinforcement learning except the reward is given under different conditions:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, \pi} [ Q^\pi(s_{t+1}, a_{t+1}) ]$$

And its corresponding recursive form is:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim \pi, \tau^\pi} [ r_D + \gamma V(s_{t+1}) ]$$

From the formula above, it is easy to see that the in IRL, the reward is provided by modeling the demonstrator’s behavior ($s_t, a_t$). All the central task in IRL is how to find the reliable reward to used in the forward reinforcement learning to guide the learning agent. Due to the difficult nature of solving problems that are not clear, are arbitrary, and are not well defined, it is important for machines to understand “how” to learn. Fig.3 shows learning scheme proposed in this work. Starts off with the sole use of the demonstrator data, in the form of $(s,a)$. This is then fed into the neural network as input data, and results in the output of the $R_d$. This is then used to find the $(s,a)$ of the agent. The agent’s $(s,a)$ is similarly placed in the neural network to produce the output, $R_d$. By finding out the loss of the gradient between $R_d$ and $R_a$, and equating the difference to 0, it is possible to denote that the optimal policy matches the feature expectation of the demonstrator’s policy.

Meanwhile, in order to avoid overlearning the reward, a bootstrapped learning scheme is also provided[2].

4 Experimental Setup

Our method is evaluated through the following setup. We utilize Gazebo[3] simulation of the Sires-17 robot from RT-Cooperation, in order to maneuver the robotic arms to move an object, from a specific location to another. In order to create the demonstration
data, or Imitation data, a pair of Oculus Rift SDK II virtual reality goggles and its headset tracking device were used. The control of large movements, such as the movement of the upper part of the humanoid robot, was done through the use of a game console controller, such as that of the PS4. Through the use of ROS[4], and the CAD simulation, it is possible for us to obtain accurate data of the specific movement done by the robot. The need for the use of such VR headsets lie in the difference in the timeframe that actions of the robot and the demonstrator is done. In other words, in traditional systems, a delay was seen between when the action of the demonstrator was done, and when the action was executed by the robot (partly as a result of the differences in frequency, and fps). By using a pair of VR goggles it is possible to create imitation data in the same Frequency as that of the robot embedded in the Gazebo. The screenshot (Fig 4) captures the Gazebo simulation of the simulated robot used for the experiment. By using the built-in integration of the Oculus Rift VR Headsets, and ROS it is possible to interact with the simulation.

Another program, known as Rviz[5], is the 3D visualization tool for ROS also plays a crucial role in enabling us to see what the robot is looking at, thinking, and doing. The image in Fig. 5 and Fig. 6 showcase the capabilities of Rviz, and its ability to integrate the real-life surroundings and the environment through the use of a camera, with the control of the robot using ROS.

5 Conclusion
In this work we investigated how although Demonstration data is commonly integrated into Reinforcement learning, but due to its drawbacks including overlearning, under learning, amongst others, inverse reinforcement learning was introduced. This was done to significantly aid the learning process and success rate of the Series-17 robot in gazebo simulation moving a brick stationed on top of a table from one location to another using a sliding action. Demonstration data throughout the experiment was collected through the use of a VR headset, in rviz simulation environment. Detailed regarding the experimental procedure, parameter tuning as well as results analysis will given in a full depth at the conference.

References