UROP Proposal and Exercises

Rohan Chitnis, `ronuchit@mit.edu`

This document describes possible projects and exercises for a UROP that is interested in getting involved in LIS (link) and being supervised by me. Our project would be advised by Leslie and Tomás, the lab's PIs. If you have other project ideas, I am very open to discussion!

1 Main idea: Learning for human-in-the-loop planning in robotics

1.1 General Problem Description

It is well-understood that in order for robots to function in the real world, they must be able to execute tasks over long time periods while in the presence of or under the supervision of a human. Even in a static environment, a robot should be able to plan out how it will act in the world, but also be willing to change its plans based on events such as intervention or information by the human. We would like to build a system that learns from data/experience how to plan efficiently and resourcefully, given that a human is in some way involved in the environment or task.

1.2 Concrete Suggestions

Here are some concrete examples of projects we could pursue. My current understanding is that these are all relatively under-explored research areas with lots of potential for a novel contribution. But there are many other ideas under this general umbrella that you may find more interesting!

- *Learning when to re-plan*: Suppose your household robot is executing its plan to clean your kitchen, and you go into the kitchen to pick up a glass of water. The robot should understand, based on your movements, that its plan for cleaning the kitchen does not need to change (except maybe that it should avoid running into you). However, suppose you accidentally drop the glass and it shatters. The robot should understand that now, it needs to add on "pick up the broken glass pieces" to its plan for keeping the kitchen clean. The research challenge here is to come up with a general formulation of the problem of when, and at what level of abstraction, to re-plan.

- *Learning to ground abstract goals based on the human's preferences*: Suppose a human asks their robot to "cook me something tasty." This is an under-specified task because it requires the robot to understand what the human considers to be tasty food. An interesting research question here is how to learn the human's preferences when given such abstract task specifications, while also making progress toward task completion. Further, can we devise a mechanism that reuses these learned preferences for similar tasks in the future?
- *Learning to incorporate unexpected human collaboration*: The field of human-robot collaboration has a rich body of literature, but there is not much work in building agents that must adapt their policies based on whether or not a human is currently collaborating. For example, suppose a robot is organizing a bookshelf, and a human comes along to help it for a few minutes, then leaves. During the time that the human is helping, the robot should recognize this and adapt its policy accordingly. The objective of this project would be to learn when and how to perform such adaptation.

2 Exercises

If you are interested in pursuing a project along these lines, please attempt some of these exercises. You're not expected to be able to do all of them, they're just here to give you a flavor of the skills you'll need to pick up. If you have any questions or would like to send your solutions for feedback, please feel free to email me at `ronuchit@mit.edu`. Note: we're not expecting perfect work here. We're just interested in how you think about these problems.

2.1 Planning

Directions: Choose either question 1 or question 2 to complete. Please also do your best at tackling question 3.

1. Markov Decision Processes (MDP) and Value Iteration
   - Implement a 2-dimensional grid-world MDP environment that has four actions: move up, move left, move right, move down.
   - Implement and run the value iteration algorithm for this environment.
   - Qualitatively check correctness of the learned value function and its corresponding policy, as you vary the transition and reward functions.
2. Graph Search
   - Create a data structure for a weighted graph.
   - Implement Dijkstra's search algorithm for finding shortest paths in the graph.

- Implement the A* search algorithm using a heuristic of your choice. What kinds of conditions do the heuristics need to meet in order for A* to return optimal solutions?

3. Understanding Research Papers: Task and Motion Planning

    - This exercise is intended get you familiar with reading and thinking about research papers. Here, we will focus on task and motion planning. TAMP is a framework for long-horizon planning that formulates planning as a search or optimization problem over both symbolic and continuous variables. There are many approaches to TAMP.
    - Read the following papers. You do not have to understand all the details, but try to understand the high-level ideas of each. If you're new to research, you can just focus on the abstract, introduction, and experiments sections of each paper.
        - Hierarchical Task and Motion Planning in the Now (link)
        - Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer (link)
        - Incremental Task and Motion Planning: A Constraint-Based Approach (link)
        - Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning (link)
    - Now answer the following questions.
        - Each paper presents a different approach to task and motion planning. What is the main idea of each approach? How do the approaches differ? How are they similar?
        - What are the biggest assumptions made by each approach?
        - What are the weaknesses of each approach? In what kinds of situations would they fail to perform well, or would the assumptions be difficult to satisfy?
        - What aspects of these systems could, if any, be learned by the robot from data or experience?

2.2 Learning

Directions: Choose one of questions 1, 2, or 3 to complete. Please also do your best at tackling question 4.

1. Deep Learning for Classification

    (a) Implement a convolutional neural network architecture that performs well at classifying digits on the MNIST dataset. You can use any popular deep learning library, such as PyTorch (recommended), Keras, TensorFlow, Theano, etc.

   (b) Why do convolutional neural networks perform well for this task? What types of robotic learning tasks would you expect them to perform well in?

2. Generative Models: VAE

   (a) Understand Variational Autoencoders (VAE) and what they are used for.

   (b) Implement a VAE and train it on the MNIST dataset. It should be able to generate its own images of digits.

3. Reinforcement Learning

   (a) Understand reinforcement learning and the Q-learning algorithm.

   (b) Implement Q-learning on your grid-world environment from a previous exercise. Remember that the agent should not have access to the transition or reward functions!

   (c) Verify that your implementation produces Q-values that converge to the optimal ones.

4. Understanding Research Papers: Machine Learning and TAMP

   - This exercise is intended get you familiar with reading and thinking about research papers in learning for task and motion planning.
   - Read the following papers. You do not have to understand all the details, but try to understand the high-level ideas of each. If you're new to research, you can just focus on the abstract, introduction, and experiments sections of each paper.
     - Learning to guide task and motion planning using score-space representation (link)
     - Active model learning and diverse action sampling for task and motion planning (link)
     - Guided Search for Task and Motion Plans Using Learned Heuristics (link)
   - Now answer the following questions.
     - Each paper presents a different approach to integrating machine learning into task and motion planning. What is the main idea of each approach? How do the approaches differ? How are they similar?
     - What are the biggest assumptions made by each approach?
     - What are the weaknesses of each approach? In what kinds of situations would they fail to perform well, or would the assumptions be difficult to satisfy?