Hierarchical Feedback Planning

We formulate feedback planning on a hierarchical model using deductive reasoning driven by values.

Overview

We teach an industrial robot to learn from human demonstrations (such as how to fold clothes), using a spatial, temporal and causal and-or graph representation [ICRA 2016]. The demonstration is multi-modal, including both visual actions captured by a Kinect camera and natural language explanations through human speech [EMNLP 2016]. The learning process is interactive with a situated dialogue, so that the robot can ask questions to the human in order to improve the knowledge representation.

The robot abstracts the states of clothes into geometric descriptions and assigns a "value" to those abstract states. Then, when folding new clothes, the robot makes deductive plans to maximize the values, in contrast to the example-based inductive learning where the robot can only follow previous human examples.

Our formulations unify hierarchical planning with deductive value-based reasoning. We implement a policy using the Task-AOG model, and search for one that maximizes expected value. Our experiments aim to show compact knowledge representation compared to flat models, as well as deductive generalization of learned skills to novel cloth-folding scenarios.

Policy

A *policy* suggests an action given a state. Throughout the literature, there are 4 primary types of policies, each one subsuming the other:

(1) In a conformant policy, the robot infers optimal actions, but blindly performs them.

```
actions = conformant_policy(s0)
for a in actions:
    do(a)
```

(2) In a Markovian feedback policy, the robot's actions are a function of the previously observed state.

```
s = s0
policy = feedback_policy(s)
while not is_goal(s):
    a = policy(s)
    s = do(a)
```

(3) A hierarchical feedback policy includes memory as well as the previously observed state.

```
s, m = s0, m0
policy = feedback_policy(s, m)
while not is_goal(s):
    a, m = policy(s, m)
    s = do(a)
```

(4) A probabilistic program policy introduces random bits.

```
s, m, r = s0, m0, seed
policy = feedback_policy(s, m, flip(r))
while not is_goal(s):
    a, m = policy(s, m, flip(r))
    s = do(a)
```

Policy π	*Has Feedback*	*Is Hierarchical*	*Is Non-Deterministic*	*Is Decidable*	Implementation
(1) Conformant	✗	✗	✗	✓	Lookup table
(2) Markov	✓	✗	✗	✓	MDP
(3) Hierarchical	✓	✓	✗	✓	Task-AOG
(4a) Program	✓	✓	✗	✗	Turing Machine
(4b) Prob. Program	✓	✓	✓	✗	Prob. Program

In the most general sense, a policy is a function on the previous state S, memory M, and randomness R. It produces a tuple containing the action A and new memory M.

$$\pi : (S \times M \times R) \to (A \times M)$$

$$\pi : (x_0, m_0, r) \mapsto (a_1, m_1)$$

Planning is about finding the optimal policy π that maximizes the expected value of an action sequence.

$$\pi^* = \arg\max_{\pi} E[\ V(a_{[1:t]}) \mid \pi\ ] = \sum_{a_{[1:t]}} V(a_{[1:t]}) P(a_{[1:t]} | \pi)$$

Plan

Given a policy, the robot must find the optimal plan.

$$a^*_{[1:t]} = \arg\max_{a_{[1:t]}|\pi} V(a_{[1:t]})$$

The search for a value function is facilitated by the expansion below.

$$\frac{\partial V}{\partial a_{[1:t]}} = \underbrace{\frac{\partial V}{\partial x}}_{\text{“why”}} \underbrace{\frac{\partial x}{\partial u}}_{\text{“how”}} \underbrace{\frac{\partial u}{\partial a_{[1:t]}}}_{\text{“what”}}$$

To solve the first part $\frac{\partial V}{\partial x}$, we define a Lagrangian,

$$L(\dot{x}, x, t) = E_{kinetic} - E_{potential} = Cost(\dot{x}) + V(x)$$

minimizing the functional,

$$x^*(t) = \arg\min_{x(t)} \int_a^b L(\dot{x}, x, t)\ dt$$

to use the Euler-Lagrange equation.

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{x}}\Big) = \frac{\partial L}{\partial x}$$

$$\frac{\partial L}{\partial \dot{x}} = \frac{d}{d\dot{x}} Cost(\dot{x}) \qquad \frac{\partial L}{\partial x} = \frac{d}{dx} V(x)$$

$$\frac{d}{dt}\Big(\frac{\partial}{\partial \dot{x}} Cost(\dot{x})\Big) = \frac{d}{dx} V(x)$$

Where the cost of a fluent-change is defined as the expected cost of actions that can achieve it.

$$Cost(\dot{x}) \triangleq E_{u|\dot{x}}[Cost(u)] = \int_{\Omega_u} Cost(u)\ P(u|\dot{x})\ du$$

And in turn, the cost of an action is recursively defined as the expected cost of a fluent-change.

$$Cost(u) \triangleq E_{\dot{x}|u}[Cost(\dot{x})] = \int_{\Omega_{\dot{x}}} Cost(\dot{x})\ P(\dot{x}|u)\ d\dot{x}$$

We hypothesis the interlinked cost functions converge to a fixed point equilibrium, like PageRank.

Either way,

$$\frac{\partial V}{\partial x} = \frac{d}{dt}\Big(\frac{\partial}{\partial \dot{x}} \int_{\Omega} Cost(u)\ P(u|\dot{x})\ du\Big)$$