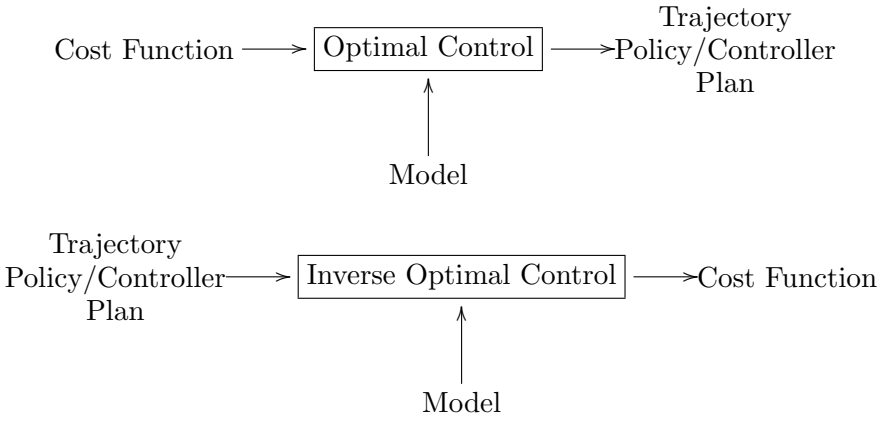
ACRL (16-899C, S09) Lecture #8 (February 10, 2009)

Inverse Optimal Control

Lecturer: Drew Bagnell *Scribe: Felix Duvallet*

1 (Inverse) Optimal Control



1.1 Motivations

Motivation for IOC:

- Find cost function of *real* trajectories.
- Cost function can be *very* hard to produce
  - Might know what you want but can't say it
- "Cost" functions and cost functions
  - More appropriate (simplified) cost functions
- Intent recognition
  - Predict a person's actions
- Scientific Interests
  - Find the cost function for some biological system
- Imitation learning
  - Command a system to replicate some demonstrated behavior
  - Conceptually, could do better than the teacher.

1.2 Complexity

Ordering in terms of generalizability:

$$\text{Trajectory} \longrightarrow \text{Policy} \longrightarrow \text{Value Function} \longrightarrow \text{Cost Function}$$

Least Generalizable *Most Generalizable*

Remarks: Cost functions generalize the best, but are more computationally expensive (need to do optimal control to generate a trajectory).

Computational complexity goes the other way ($\leftarrow$). Amount of Modeling required goes to the right.

2 Solving the General IOC problem

At a high level, this is what's happening:

1. Get a set of features
2. Drive robot (or show a path) to get a demonstrated path.
3. Generate a cost function that makes the demonstrated trajectory(ies) optimal.

The features can be anything salient in the available data. For our robot navigation example, some potential features are:

- Green-ness
- Density of LIDAR data
- ...

The simplest cost function on the features we can think of is a linear combination of the features:

$$c(w) = w^{\mathrm{T}} f \tag{1}$$

2.1 IOC as optimization

We can formulate the above problem as a big optimization problem:

$$\forall i \text{ cost of expert plan}_i \leq \text{ cost of arbitrary plan} \tag{2}$$

Since this is true for all plans, it must be true for any plan. Thus, it's true for the *best* plan. This reduces the size of the optimization problem.

So we can re-write more formally as:

$$min||w||^2 \tag{3}$$

subject to

$$\sum_{i \in \text{ demonstrated plan}} w^{\mathrm{T}} f(\text{state}_i) \leq \min_{\text{plan}} \sum_{i \in \text{ plan}} w^{\mathrm{T}} f(\text{state}_i) \tag{4}$$

This presents a problem that the zero-vector is "optimal" for w, so we will optimize with a margin.

2.2 Subgradient Optimization

With a loss term, we now have an unconstrained optimization problem:

$$c(w) = \sum_{i}^{N} \left(w^{\mathrm{T}} f_i(\xi_i) - \min_{\xi \in C_i} \left(w^{\mathrm{T}} f_i\left(\xi\right) - l_i\left(\xi\right) \right) \right) + \frac{\lambda}{2} ||w||^2 \tag{5}$$

where i are the examples, ξ are paths (ξ_i is a demonstrated path), and w is the current policy (weight vector). This tries to minimize the difference between human examples and the planned path using the current policy.

Since $c(w)$ is convex in w, we can use the gradient. However, it's ill-defined because of the min operator. Fortunately, the sub-gradient always does go towards the optimal:

$$\nabla c(w) = \sum_{i}^{N} f_i(\xi_i) - f_i(\xi^*) + \lambda w_i \tag{6}$$

where ξ^* is the "best" candidate path.

Basically, we are computing the over-counted and under-counted features and computing the difference between the demonstrated and the optimal path. Our update rule becomes:

$$w \leftarrow w + \alpha \nabla \tag{7}$$

2.3 Boosting Version

Here's a boosting version of the algorithm:

- +1 for places expert went to
- -1 for places current plan goes to

Classifier makes w best for this iteration.
Repeat until convergence, where $\text{cost\_function}(x) = \sum_i \text{classifier}_i(x)$.

2.4 Pathological cases and other failures

If a plan could never be generated by the Optimal Control algorithm, we will never find a weight vector that will generate that type of plan. Examples include a plan with a loop. This really only becomes a problem if the demonstrated path is very sub-optimal.

Another failure case is if there are "hidden" features in the plan (for example, an unknown objective), we will not be able to generate a weight vector.

In most cases, the failures of the IOC can be countered by adding the necessary features to the feature vector.

3 Relationship to LQR

$$\sum_{t=1}^{T} {x_t^d}^{\mathrm{T}} Q x_t^d + u_t^{\mathrm{T}} R u_t \leq \sum_{t=1}^{T} {x_t^*}^{\mathrm{T}} Q x_t^* + {u_t^*}^{\mathrm{T}} R u_t^* \tag{8}$$

where x_t^d is the demonstrated behavior, and x_t^* is the current best optimal control. $Q = Q^{\mathrm{T}} \succ Q$, $R = R^{\mathrm{T}} \succeq R$.

3.1 Gradient Descent

To do gradient descent, we define $\tilde{R} = R + I$.

$$\frac{\partial}{\partial Q} \left(\sum_{t=1}^{T} {x_t^d}^{\mathrm{T}} Q x_t^d + u_t^{\mathrm{T}} \tilde{R} u_t - \sum_{t=1}^{T} {x_t^*}^{\mathrm{T}} Q x_t^* + {u_t^*}^{\mathrm{T}} \tilde{R} u_t^* \right) \tag{9}$$

Differentiating element-by-element (using Matrix calculus) yields:

$$\frac{\partial}{\partial Q} = \sum_{t=1}^{T} x_t^d {x_t^d}^{\mathrm{T}} - \sum_{t=1}^{T} x_t^* {x_t^*}^{\mathrm{T}} \tag{10}$$