Statistical Techniques in Robotics (16-831, F08) Lecture #28 (Dec 02, 2008)

Imitation Learning

Lecturer: Drew Bagnell *Scribe: Arvind Rao*

1 Imitation Learning via Inverse Optimal control

what is it? : recall SVM for ranking footsteps, want the robot to recover the *high-level* behavior shown to the robot

Imitation Learning is different from supervised learning

- Decisions have consequences
- Actions and decisions are purposeful

Consider the ALVINN system: If you predict with all but ϵ mistakes the correct direction of car driving – how bad can ϵ be?. If the solutions you see are part of the training process, you'll be ok, but if you see something unexpected, then you might make some big mistakes. If the error is correlated with the direction that you are heading before, then the errors compound. Such situations are indicative of the problems that arise when the testing data has nothing to do with the training data.

A potential solution to the above problem is: reasoning over trajectories, i.e. instead of predicting single actions, predict trajectories of actions . Recall that in supervised learning we only take a decision one step into the future.

Question: How do we make a "sequence" of decisions?. This is a "planning problem" in robotics, i.e. think of a sequence of motions that takes you to a goal. There are two goals in imitation learning:

- Take advantage of the planning algorithm to take a sequence of actions,
- Learning tells us how to get the planning algorithm to do what we want.

Observe a sequence of actions to go from point A to point B and train robot to go on any arbitrary trajectory (as taken by the human).

Goal: come up with algorithm that sets costs so that the human trajectory is the best cost path.

Why is this *inverse* optimal control? In typical control problems, you find a planner/controller that minimizes some objective cost function over time. Whereas in "inverse" optimal control, you find what performance metric makes a particular plan (sequence of actions) optimal.

Formulation: Consider the set ξ of all possible paths between two points A and B. We need that, the cost of the example(expert) path i be less than the cost of any arbitrary path in the set ξ. Here, $i \in (1, 2, \ldots, N)$, N being the total number of data points over which data is available. Therefore, we have,

$\sum_{i=1}^{N}$ [cost of desired path for data point i - min cost over all paths for data point i]

if we instead consider a linear cost function of the form $w^T f_j$, as the cost of the j^{th} cell (recall the step planning setup from previous class). Let the optimal plan for data point point i be ξ_i, therefore, we now have,

$min_w[\sum_{j\in\xi_i} w^T f_j - min_\xi \sum_{j\in\xi} w^T f_j]$

adding a margin (recall the analogous SVM trick), and looking over all N data points, this becomes,

$min_w \sum_{i=1}^{N}[\sum_{j\in\xi_i} w^T f_j - min_\xi \sum_{j\in\xi}(w^T f_j - l_j)]$

Solution: Use gradient descent (taking gradient with respect to the weight vector w),

$w_{t+1} = w_t - \alpha[\sum_{j\in\xi} f_j - \sum_{j\in\xi^*} f_j]$ (where ξ^* is the argmin of $\sum_{j\in\xi}[w^T f_j - l_j]$ above)

A limitation of this approach is that it needs non-negative costs, but this is sidestepped if we use exponentiated gradient.

This is the linear version of max-margin planning approach , a non-linearized version can be constructed using kernelization.