Adaptive Control and Reinforcement Learning (16-899C, S09) Lecture #5 (Date)

Planning using Dynamic Optimization

Lecturer: Chris Atkeson *Scribe: Elliot Cuzzillo*

1 Problem

We consider the problem of finding a plan that globally minimizes some cost function C(execution), over a discretized, deterministic state space and discretized timesteps.

2 Approaches

Primarily we consider solutions using dynamic programming. These solutions require that we represent all states explicitly, which means that they become unfeasible for higher-dimensional problems (the "curse of dimensionality"). We also consider some special-purpose solutions.

3 Trajectories

3.1 Theory

Recall from previous lectures that in the linear quadratic regulator problem. There, we linearize the dynamics of the system around a point. The dynamics are then given by

$$x_{t+1} = Ax_t + Bu_t$$

where A and B are matrices, x_t is the state at time t, and u_t is the controls applied at time t. The cost is given by

$$\text{cost}(x_t, u_t) = x_t Q_t x_t^T + u_t R_t u_t^T$$

Now we consider a more general dynamic system:

$$x_{t+1} = f(x_t, u_t)$$

One possible approach is to discretize the state space and apply dynamic programming methods over all the states. This is considered later in the lecture.

Another approach is to begin with a trajectory, for instance from a human teacher, and linearize the dynamics about that trajectory. Then, we have a time sequence $x_0^h, u_0^h, \ldots$ (where the h stands for 'human teacher'). About the trajectory point at each timestep, we make a taylor expansion of the dynamics $f(x_t, u_t)$.

A minor caveat is that if we linearize the dynamics about this trajectory point, there's an offset (the zeroth-order term in the taylor series), and the LQR equations are derived to requlate to zero, so we have to rederive the equations to use an offset:

- First, augment the state vector with a trailing 1, as is done in graphics with homogenous matrices:

$$\hat{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}$$

- Also just like in graphics, we augment A as follows:

$$\hat{A} = \begin{pmatrix} A & x_t^d \\ 0 & 1 \end{pmatrix}$$

  where x_t^d is the desired state of the system according to the trajectory at time t.

- We augment B entirely with zeros.

$$\hat{B} = \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix}$$

- Finally, we take the second-order taylor expansion of the cost function around the trajectory point. We don't care about the constant term in this expansion, since only cost differences are relevant to the problem. We do care about the linear and quadratic terms; the cost equation changes to add a linear term $q_t^T x_t$, and we define

$$\hat{Q} = \begin{pmatrix} Q & q_t/2 \\ q_t^T/2 & \gamma \end{pmatrix}$$

  where γ is chosen large enough that the matrix is positive definite. (In effect, γ specifies how much cost comes from the trailing 1 in x_t.

3.2 Evaluation

Having explored trajectories, we now note that trajectories can be spectacularly problematic in particular instances, such as riding a bicycle. If you use a trajectory-based method on a bicycle, the trajectory is mostly irrespective of state, so, as Chris colorfully put it, YOU ARE GOING TO DIE. The bicycle is an unstable system, so inevitably, even with a perfect model, you get pushed a little bit off the trajectory, and then you die.

So, we have to look at another approach.

4 Policies

Instead of trajectories, where we have a desired position at each timestep $x_d(t)$, we now consider general policies $u(x)$, mapping states to actions.

An entertaining interlude on shaping: *Shaping* is a technique where you generate a family of easier problems to use as stepping stones for the larger problem. An example given in class: if you want to get a killer whale to jump out of its tank and kiss its trainer, you start by conveniently noting a naturally occurring behavior in killer whales: when there are a bunch of seals on a beach, the whale will peek out of the water periodically, and then suddenly jump onto the seals, eat them, and then come back into the water.

So you get the whale to do that, but pull on its tail to prevent it from eating the trainer.

Interlude over.

- One idea for making a policy is to have a bunch of trajectories, and then at each point in the state space, find the nearest trajectory, and follow it. Difficulties include finding a good distance metric to decide which trajectory to follow, and also the possibility that the current state is outside the stable envelope of the chosen trajectory.

  However, this method (termed by Chris 'WWJD') does do well with state error and model errors. It has much larger issues when the environment changes and all the trajectories no longer do anything useful (for instance, if the target state changes). It also fails if the trajectories are not globally optimal (for instance, if they cross).

- Another approach is to parameterize a policy by $u(x, \theta)$, where θ is a vector of parameters. Then, the policy problem is turned into a normal optimization problem. Then, we run into the issue that nobody knows how to design good cost functions.

5 Bellman equation

Now we introduce dynamic programming and the Bellman equation as a way to find a globally optimal policy. Here, the lecture became briefer, and the slides became a superset of what was said verbally, so it would be better to refer directly to those slides:

`http://www.cs.cmu.edu/~cga/acrl/lectures/acrl-dp-public-09.ppt`