Homework 1:
MDP approximation to inverted pendulum

E. Todorov, AMATH/CSE 579

due April 30, 2015

Problem statement

In this homework we will solve a simple optimal control problem – balancing an inverted pendulum. This will be done by discretizing the problem as an MDP, which will then be solved with either value iteration or policy iteration. The continuous-time dynamical system is:

$$\begin{aligned} dx_1 &= x_2 dt \\ dx_2 &= (a \sin(x_1) - b x_2 + u)\, dt + \sigma d\omega \end{aligned}$$

x_1 is the angle of the pendulum, x_2 is the angular velocity, u is the control, ω is Brownian motion. The cost rate function is

$$\ell(\mathbf{x}, u) = 1 - \exp(k \cos(x_1) - k) + \frac{r}{2} u^2$$

The problem is defined in the *discounted cost* setting, with discount factor which is up to you. Here $a > 0$ models gravity, mass and length; $b > 0$ is damping; $\sigma > 0$ is the noise scaling; $k > 0$ determines the shape of the cost function (larger k makes the cost sharper); $r > 0$ is the weight of the control cost. The constants a, b, σ, k, r together with the discount factor can be adjusted to obtain different problem instances; you should experiment with them and try to understand how they affect the solution.

Part 1: Constructing the MDP

Construct the approximating MDP as follows. First decide on a time step h, maximum velocity $v_{\max}$ and control $u_{\max}$ to be represented in the MDP, and

numbers of grid points (n_1, n_2, n_u) along the state and control dimensions. Dimension (x_1) is circular and should be discretized with care, enforcing wrap-around. Dimension (x_2) and the control dimension (u) are discretized over the intervals $[-v_{\max}; v_{\max}]$ and $[-u_{\max}; u_{\max}]$ in a straightforward way. The MATLAB command "linspace" is useful here.

Now we have a grid of discrete states, each corresponding to some continuous state (x_1, x_2). For each discrete state and each discrete control, compute the continuous next state (y_1, y_2) under the deterministic part of the pendulum dynamics:

$$\begin{aligned} y_1 &= x_1 + hx_2 \\ y_2 &= x_2 + h\left(a \sin\left(x_1\right) - bx_2 + u\right) \end{aligned}$$

Now we have to create transition probabilities in the MDP such that the mean and covariance of the resulting mixture-of-delta-functions are close to the mean and covariance of the Gaussian density defined by the continuous dynamics. We discussed different methods for doing this in class. Use the easiest one: choose possible next states/grid points around the mean, evaluate the Gaussian at the chosen grid points, and normalize so that the outgoing transition probabilities sum up to 1 at each state. Note that (y_1, y_2) will sometimes fall outside the grid in the velocity dimension. In that case we cannot achieve the above requirement for matching the means and covariances. Instead we will use the nearest grid states and accept some approximation error (i.e. edge effect). The cost in the MDP is $h\ell$ where ℓ is evaluated at the discretized states and controls.

There are several unspecified constants: $h, v_{\max}, u_{\max}, n_1, n_2, n_u$ in the MDP construction. Similar to the above constants appearing in the problem formulation, you should define them as symbolic constants at the beginning of your code, experiment with them, and try to understand their effects.

Generally, denser grids produce more accurate solutions. It is also useful to think about "compatibility" of the grid in the following sense. Suppose the time step h is small and the discretization step in velocity is small, while the discretization step in position is large. Then it will be very difficult for the MDP to make any state transitions, i.e. the transition probabilities will be close to delta functions centered at the current state. This is to be avoided. Instead we want

$$h\frac{2v_{\max}}{n_2} \approx \frac{2\pi}{n_1}$$

Similar reasoning applies to the velocity and control discretization.

Part 2: Solving the MDP

Now that we have an MDP, we can use the policy iteration and value iteration algorithms discussed in class (feel free to use the MATLAB code from the lecture notes as a starter). For the purpose of exploring the effects of the different parameters you can use either algorithm. Once you settle on a set of parameters that you like, compare the convergence of policy and value iteration. This can be done by plotting a 2D image representing the value function and observing how it changes over iterations. You can also plot the amount of change in the value function between iterations, and observe how quick the convergence is for the two algorithms.

Part 3: Simulating the control law

Once the MDP is solved, you have a control law defined on the grid. Use interpolation to extend the control law to the continuous space, and simulate several trajectories for the continuous system, starting at different initial states. If your controller (and problem and discretization parameters) are sensible, you should see the pendulum going to the vertical state and balancing there. Note that the system is stochastic, so you should inject the necessary amount of noise at each simulation step. The simulation is done in continuous space, but discrete time.

What to submit

- MATLAB code.
- Figures showing the optimal value function and the optimal control law for different parameter settings you found interesting.
- Figures comparing the convergence of policy and value iteration.
- Figures showing simulated trajectories under the optimal control law for several initial states, with noise added to the simulation.
- Text summarizing your observations.