Adaptive Control and Reinforcement Learning (16-899C, S09) Lecture #3 (1/20/09)

Dynamic programming, MDP, generalization of LQR

Lecturer: Drew Bagnell *Scribe: Slava Arabagi*

1 Controls Problems with State

Finite time (horizon), deterministic, Markov decision process

1.1 Problem parameters

- (s,x) - state (continuous/discrete)
  ex. $x \in \{1, 2, \ldots, n\}$
- (a,u) - actions (continuous/discrete)
  ex. $u \in$ {steer left, steer right, forward ...}
- System dynamics, state transition equation
  $f(x_t, y_t) \rightarrow x_{t+1}$
- Reward/Cost function $C(x_t, u_t)$
- x_0 - initial start state

Generally want to minimize expression $\sum_{t=0}^{T} C(x_t, u_t)$

1.2 General Approaches

1. Enumerate all action sequences, pick the best
2. Optimize over a particular policy class
3. Search in graphs of actions (A*,D*), used commonly in motion planning

1.3 Overall Procedure

Some Definitions

- Policy $\pi(x)$ - function that maps states into controls action, for observable states, or maps past data into controls, for stochastic modeling. That is:
  $\pi : z_{1:t-1}, u_{1:t-1} \rightarrow u_t$ for the general case (stochastic/observable)
  $\pi : x_t \rightarrow u_t$ for completely observable case
- Value function $V(x, u)$ - measures the expected value of following a policy.

In general, the algorithms of optimizing a control policy propagate their updates in reverse temporal order through the value function, through a process known as a "backup step". Hence, the algorithms start their optimal policy search in the last time-step t=T-1:

- t=T-1 (last time step)

$$\pi^*_{T-1} = \underset{a}{argmin} \quad C(x_{T-1}, a)$$
$$V^*_{T-1} = C(x_{T-1}, \pi^*_{T-1})$$

  Note: the * denotes optimal policy/value fcn.

- t=T-2

$$\pi^*_{T-2} = \underset{a}{argmin}\big(C(x_{T-2}, a) + V^*_{T-1}(f(x_{T-2}, a))\big)$$

- t=T-3
  Repeat recursive steps to obtain π^* at each time step.

Note that the order of this algorithm is not exponential, because we initially specified a policy in the last time step t=T-1. Instead this algorithm is $O(T|s||a|)$ (product of times, # of states and actions).

1.4 Generalization to stochastic, finite horizon Markov decision processes (MDP)

In the stochastic generalization, the Markov property needs to be applied:
$P(x_{t+1}|x_t, a_t, x_{t-1}, a_{t-1}, ..., anythingelse) = P(x_{t+1}|x_t, a_t)$

Furthermore, we still have the components of the deterministic model state (x), actions (a), initial state (x_0) and reward/cost function (R). However, now the transition dynamics are described by:
$x_{t+1} = D(x_{t+1}|x_t, a_t)$
and the reward function R, called now the "expected cumulative payoff", takes the form:
$R_T = E\left[\sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau}\right]$

Similarly the expected cumulative payoff of a policy is defined:
$R^\pi_T(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau}|u_{t+\tau} = z_{1:t+\tau-1}, u_{1:t+\tau-1}\right]$

The optimal policy is defined:
$\pi^* = \underset{\pi}{argmax}[R^\pi_T(x_t)]$

General Procedure
Want to maximize the expected reward or minimize the expected cost:

$$E\left[\sum_{t=0}^{T-1} C(x_t, u_t)\right] = \sum_{t=0}^{T-1} E\left[C(x_t, u_t)\right] = \sum_{t=0}^{T-1} \sum p(x_t) C(x_t)$$

1.4.1 Policies and Value Functions

The value of T in the expressions for R_T^π determines the number of time steps a policy is optimized with regards to, effectively a look-ahead value. If T=1 the policy is a greedy one, optimizing only the next step. For $T > 1$, the algorithm has finite horizon specified by T. $T = \infty$ denotes the infinite horizon case.

- 1-step policies
  - 1-step optimal policy
    $\pi_1^*(x) = \underset{u}{argmax}[r(x,u)]$
  - 1-step optimal value function
    $V_1^*(x) = \gamma \quad \underset{u}{max}[r(x,u)]$
- 2-step policies
  - 2-step optimal policy
    $\pi_1^*(x) = \underset{u}{argmax}\left[r(x,u) + \int V_1(x')p(x'|u,x)dx'\right]$
  - 2-step optimal value function
    $V_1^*(x) = \gamma \quad \underset{u}{max}\left[r(x,u) + \int V_1(x')p(x'|u,x)dx'\right]$
- T-step policies
  - T-step optimal policy
    $\pi_T^*(x) = \underset{u}{argmax}\left[r(x,u) + \int V_{T-1}(x')p(x'|u,x)dx'\right]$
  - T-step value function
    $V_T^*(x) = \gamma \quad \underset{u}{max}\left[r(x,u) + \int V_{T-1}(x')p(x'|u,x)dx'\right]$

This algorithm has been used successfully in the stochastic algorithm of value iteration for motion planning, as portrayed in Figs. 1, 2.

2 Continuous MDP's, Linear Quadratic Regulator (LQR)

Suppose, we require to design an optimal controller for a fully deterministic system whose state transition function is:
$x_{t+1} = Ax_t + Bu_t$
$x \in \Re^n, \quad u \in \Re^m$
Define the cost function to optimize with respect to:
$C(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$
Where the matrix Q is positive semi-definite, symmetric and R is positive definite, symmetric.
$Q = Q^T, \quad Q \geq 0, \quad R > 0.$
These constraints ensure that C is convex and $C > 0$.

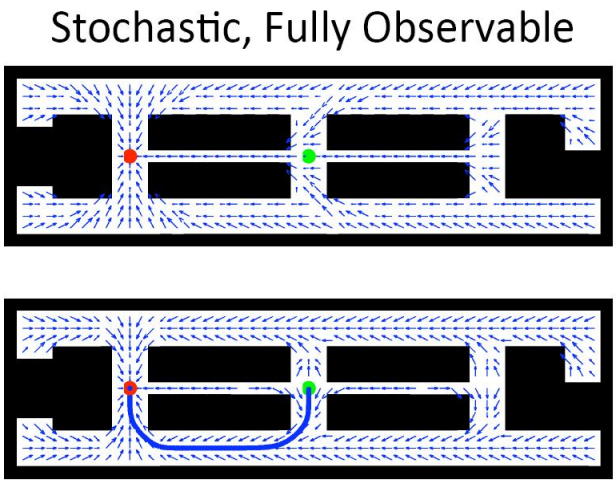

Figure 1: Motion plan map (S.Thrun, *Probabilistic Robotics*, MIT Press, 2005, pp. 491).

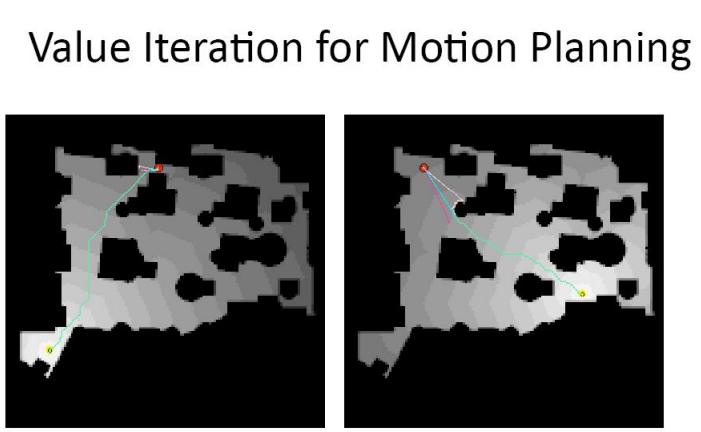

Figure 2: Value iteration in action without stochasticity (S.Thrun, *Probabilistic Robotics*, MIT Press, 2005, pp. 504).

2.1 Dynamic Programming

Apply dynamic programming to solve the problem. Define the value function $J(x_t, u_t) \equiv x_t^T V_t x_t$.

- Initial Step t=T-1

$$J_{T-1}(x_{T-1}, u_{T-1}) = x_{T-1}^T Q x_{T-1} + u_{T-1}^T R u_{T-1}$$

Because the iteration occurs at the last step of the algorithm, the function J_{T-1} is already optimal and hence no input u_{T-1} can improve it, hence we have $u_{T-1} = 0$, then:
$J_{T-1}^*(x_{T-1}) = x_{T-1}^T Q x_{T-1} = x_{T-1}^T V_{T-1} x_{T-1}$

- Recursive Step t=T-2

$$J_{T-2}(x_{T-2}, u_{T-2}) = x_{T-2}^T Q x_{T-2} + u_{T-2}^T R u_{T-2} + (Ax_{T-2} + Bu_{T-2})^T V_{T-1}(Ax_{T-2} + Bu_{T-2})$$

We want to obtain the minimum cost over the last 2 steps, and since we know that the cost function C, thus J_T must also be convex and have a global minimum. Hence, we set the derivative to 0 and solve for the input that achieves that.

$$\begin{aligned}
0 = \frac{\partial J}{\partial u} &= 2u_{T-2}^T R + 2(Ax_{T-2})^T V_{T-1} B + 2u_{T-2}^T B^T V_{T-1} B \\
u_{T-2}^T \left[R + B^T V_{T-1} B\right] &= -x_{T-2}^T A^T V_{T-1} B \\
u_{T-2}^T &= -x_{T-2}^T K_{T-2} \\
K_{T-2} &= A^T V_{T-1} B \left[R + B^T V_{T-1} B\right]^{-1}
\end{aligned}$$

The last line is the Kalman gain (named after its inventor). Note that the controller is linear with the state vector x, and is an optimal controller in a linear quadratic sense. The plant to be controlled in LQR has to have a linear form, however in case of nonlinearity the usual

procedure of linearization about the desired operating point can be utilized, although several iterations of gradient descent might be required to achieve the optimum controller, as opposed to a one shot mechanism presented above.