Dynamic programming

The most important method to find the optimal control law for a recursive state-observed system is dynamic programming. In this chapter, we'll look at how it works. In the first part, we look at finite horizons. That is, time is limited. In the second part, we examine what to do if time can run on indefinitely.

1 Dynamic programming on a finite horizon

1.1 The problem and the cost function

Let's consider a recursive state-observed stochastic system, described by

$$x(t+1) = f(t, x(t), u(t), v(t)). \tag{1.1}$$

We assume that the time horizon is finite. So, $T = \{0, \ldots, t_1\}$. To control this system, we use the input $u(t)$. This input is given by the control law g, according to

$$u^g(t) = g(t, x^g(t), x^g(t-1), \ldots, x^g(t_0)). \tag{1.2}$$

The resulting state of the system is denoted by $x^g(t)$.

The question arises: which input $u(t)$ should we select? Our goal is to control the system in such a way that a **cost function** $J(g)$ is minimized. An example of such a cost function is

$$J(g) = E_g \left[\sum_{s=0}^{t_1-1} b(s, x^g(s), u^g(s)) + b_1(x^g(t_1)) \right]. \tag{1.3}$$

Here, the E_g operator means the expectation, given that the control law g is used. Also, b_1 is the **terminal cost** and b is the **current cost**. We thus need to find the optimal control law g^*, such that $J^* = J(g^*) = \inf_{g \in G} J(g)$. (Note that G is the set of all possible control laws g.) To find g^*, a helpful function is the **conditional cost-to-go** $J(g,t)$ at time t. It is defined as

$$J(g,t) = E_g \left[\sum_{s=t}^{t_1-1} b(s, x^g(s), u^g(s)) + b_1(x^g(t_1)) | F_t^{x^g} \right]. \tag{1.4}$$

It is interesting to note that we have $J(g) = E_g[J(g, t_0)]$.

1.2 The dynamic programming procedure

To find g^*, we make use of a **value function** $V(t, x^g(t))$. This value function satisfies $V(t, x^g(t)) \leq J(g,t)$. In fact, if we use the optimal control law $g = g^*$, then we have an equality. In other words, we have

$$J^* = J(g^*) = E_{g^*}[J(g^*, t_0)] = E_{g^*}[V(t_0, x_0)]. \tag{1.5}$$

But how do we find V? The value function is generally derived by backward recursion. First, we define V at time $t = t_1$. So, $V(t_1, x) = b_1(x)$ for all $x \in X$. We then find V recursively using

$$V(t,x) = \inf_{u \in U} \left(b(t,x,u) + E[V(t+1, f(t,x,u,v(t))) | F^{x,u}] \right). \tag{1.6}$$

The above equation is known as the **dynamic programming equation**. Once we have the value function, we can find the optimal control law g^*. We simply select the u for which the above relation is minimized. This value of u is denoted by u^*. We thus have

$$g^*(t,x) = u^* = \arg \inf_{u \in U} \left(b(t,x,u) + E[V(t+1, f(t,x,u,v(t))) | F^{x,u}] \right). \tag{1.7}$$

It can now be shown that the value function must satisfy $V(t_1, x^{g^*}(t_1)) = b_1(x^{g^*}(t_1))$ and

$$V(t, x^{g^*}(t_1)) = b(t, x^{g^*}(t), g^*(t, x^{g^*}(t))) + E_{g^*}\left[V(t+1, f(t, x^{g^*}(t), g^*(t, x^{g^*}(t)), v(t)))|F_t^{x^{g^*}}\right]. \quad (1.8)$$

Note that in the above **dynamic programming (DP) procedure**, we don't directly try to find g^* from the set G. Instead, we simply find the optimal value of u^* from the set U at every time-step. This is generally much easier.

1.3 Linear quadratic-Gaussian stochastic control

Let's consider a special case of the above problem. First of all, we examine a Gaussian stochastic control system

$$x(t+1) = A(t)x(t) + B(t)u(t) + M(t)v(t). \quad (1.9)$$

Also, we use the quadratic cost function

$$J(g) = E_g\left[\sum_{s=t}^{t_1-1}\begin{bmatrix}x(s)\\u(s)\end{bmatrix}^T L(s)\begin{bmatrix}x(s)\\u(s)\end{bmatrix} + x(t_1)^T Q_1 x(t_1)\right], \quad \text{with} \quad L(t) = \begin{bmatrix}Q(t) & S(t)\\S(t)^T & R(t)\end{bmatrix} \quad (1.10)$$

and with Q_1 being the terminal cost. Also, we have $L(t) = L(t)^T \geq 0$ and $R(t) > 0$. For this case, we can derive an analytic expression for $g^*(t)$. First we define $P(t+1) = Q_1$. Now we recursively define

$$\begin{aligned}
H_{11}(t) &= A(t)^T P(t+1)A(t) + Q(t), & (1.11)\\
H_{12}(t) &= A(t)^T P(t+1)B(t) + S(t), & (1.12)\\
H_{22}(t) &= B(t)^T P(t+1)B(t) + R(t), & (1.13)\\
F(t) &= -H_{22}^{-1}(t)H_{12}^T(t), & (1.14)\\
P(t) &= H_{11}(t) - H_{12}(t)H_{22}^{-1}(t)H_{12}^T(t). & (1.15)
\end{aligned}$$

In this way, the above matrices can be determined for every time t. The optimal control law $g^*(t, x)$ can now be found using

$$g^*(t, x) = F(t)x. \quad (1.16)$$

The corresponding value function and optimal cost function can be found using

$$\begin{aligned}
V(t, x) &= x^T P(t)x + r(t), & (1.17)\\
r(t) &= r(t+1) + \text{trace}(M(t)^T P(t+1)M(t)V(t), & (1.18)\\
J^* &= E_g\left[x_0^T P(t_0)x_0\right] + r(t_0). & (1.19)
\end{aligned}$$

2 Dynamic programming on an infinite horizon

2.1 Cost functions

This time, we consider the case where we have an **infinite horizon**. So, $T = \mathbb{N} = \{0, 1, \ldots\}$. For the infinite-horizon case, variations in the system dynamics are often negligible. So we assume that the system is time-invariant, implying that

$$x(t+1) = f(t, x(t), u(t), v(t)). \quad (2.1)$$

The problem with infinite-horizon problems is that the cost function $J(g)$ often becomes infinite. To solve this problem, we have to use a different cost function. An example of such a cost function is the

discounted cost function, being

$$J_d(g) = E_g\left[\sum_{s=0}^{\infty} r^s b(x^g(s), u^g(s))\right], \tag{2.2}$$

where r is the **discount rate**. With this cost function, costs that occur in the future are weighted less. Another option for a cost function is the **average cost function**. This is defined as

$$J_{av}(g) = \lim_{t\to\infty}\left(\frac{1}{t}E_g\left[\sum_{s=0}^{t-1} b(x^g(s), u^g(s))\right]\right). \tag{2.3}$$

Which cost function is the most suitable depends on the problem. In the rest of this chapter, we will mainly consider the discounted cost function.

2.2 The transition measure

For simplicity, we will consider the case where the input space U and the state space X are both finite. We also assume that we can only use Markov control laws $u^g(t) = g(t, x^g(t))$. In this case, we can write the system dynamics in a completely different form. We don't use the function f anymore. Instead, we denote the chance that $x(t+1) = i$, given that $x(t) = j$ and $u(t) = u$, by

$$P(i, j, u) = P(x(t+1) = i | x(t) = j, u(t) = u). \tag{2.4}$$

Here, P is the **transition measure**. We can also use this transition measure for conditional expectation. We then get

$$E_g\left[V(t+1, x^g(t+1))|F^{x^g(t)}\right] = \sum_{x_1\in X} V(t+1, x_1)P(x_1, x^g(t), u_g(t)). \tag{2.5}$$

2.3 The forwardly defined value function

For the problem we are considering, the backwardly defined cost-to-go function $W(t, x)$ is defined as $W(t_1, x) = 0$ and

$$W(t, x) = \min_{u\in U}\left(r^t b(x, u) + \sum_{x_1\in X} W(t+1, x_1)P(x_1, x, u)\right). \tag{2.6}$$

However, in our case, there is no ending time t_1. So we would better consider the forwardly defined value function $V(t, x)$. It is defined as $V(0, x) = 0$ and

$$V(t, x) = \min_{u\in U}\left(b(x, u) + r\sum_{x_1\in X} V(t-1, x_1)P(x_1, x, u)\right). \tag{2.7}$$

It can be noted that we have $V(t, x) = r^{t-t_1}W(t_1 - t, x)$. As time t goes to infinity, it can be shown that $V(t, x)$ converges. In fact, if we write $V(\infty, x) = V(x)$, then we have

$$V(x) = DP(V)(x) = \min_{u\in U}\left(b(x, u) + r\sum_{x_1\in X} V(x_1)P(x_1, x, u)\right). \tag{2.8}$$

The above relation has a unique solution for the function $V(x)$. And it can also be shown that the resulting value function $V(x)$ minimizes the discounted cost function. So,

$$V(x_0) = \inf_{g\in G} E_g\left[\sum_{s=0}^{\infty} r^s b(x^g(s), u^g(s))|F^{x_0}\right]. \tag{2.9}$$

Once we have the value function $V(x)$, the optimal control law can be found using

$$g^*(x) = \arg\min_{u \in U} \left(b(x,u) + r \sum_{x_1 \in X} V(x_1) P(x_1, x, u) \right). \tag{2.10}$$

Note that this control law does not depend on time t. So, it is a stationary control law.

2.4 Algorithms for finding the value function

The question remains, how can we find $V(x)$? There are multiple methods. The first one we consider is **value iteration**. In this method, we make use of a function $h(m,x)$ which is similar to $V(t,x)$. We initialize $h(0,x) = 0$ for all x. We then keep on updating $h(t,x)$, according to

$$h(m+1,x) = \min_{u \in U} \left(b(x,u) + r \sum_{x_1 \in X} h(m, x_1) P(x_1, x, u) \right). \tag{2.11}$$

We know that, as $m \to \infty$, then $V(x) = h(m,x)$. The downside of this method is that we need infinitely many computations before convergence occurs.

The method of **policy improvement** solves the above problem. Before we discuss this algorithm, we make some definitions though. Let's suppose that the state space X has n elements $x_1, \dots x_n$. We define the vectors V and $b(g)$ as

$$V = \begin{bmatrix} V(x_1) \\ V(x_2) \\ \vdots \\ V(x_n) \end{bmatrix} \qquad \text{and} \qquad b(g) = \begin{bmatrix} b(x_1, g(x_1)) \\ b(x_2, g(x_2)) \\ \vdots \\ b(x_n, g(x_n)) \end{bmatrix}. \tag{2.12}$$

Also, let's define the matrix $P(g)$ as the matrix with elements $P_{ij}(g) = P(x_j, x_i, g(x_i))$. In other words, if we are in a state x_i and use an input $g(x_i)$, then the ith row gives us the chances that we reach a state x_j in the next time step, with j the column number. (Note that the sum of the elements in every row of $P(g)$ is 1.) We now need to find a control policy g such that

$$V(g) = DP(V(g)) = b(g) + rP(g)V(g). \tag{2.13}$$

To do this, we first take a random control law g_0 and solve the equation $V(g_0) = b(g_0) + rP(g_0)V(g_0)$ for $V(g_0)$. (Note that this simply is a linear equation.) We then derive a new control law g_m which satisfies

$$b(g_m) + rP(g_m)V(g_{m-1}) = \min_{g \in G} \left(b(g) + rP(g)V(g_{m-1}) \right). \tag{2.14}$$

Note that the set of possible control laws G has a finite size, because the state space X and the input space U are finite. So the above relation can be solved in finite time. Once we have the new control law g_m, we can also update our value function $V(g_m)$ by solving

$$V(g_m) = b(g_m) + rP(g_m)V(g_m). \tag{2.15}$$

This algorithm continues as long as improvements are made to the policy. That is, as long as

$$\min_{g \in G} \left(b(g) + rP(g)V(g_{m-1}) \right) < V(g_{m_1}). \tag{2.16}$$

If the above condition does not hold anymore, then the algorithm has converged to the optimal policy (or one of the optimal policies, in case there are multiple ones). This optimal policy is then simply given by $g^* = g_m$.