

Sequential Decision Making

Additional Readings: Bertsekas, Cha. 1

Notation

K	final stage
k	individual stages $\{1, \dots, K\}$
X	state space
x	individual states
x_k	state at stage $k \dots x_1 =$ initial state and $x_K =$ final state
x_G	goal state
F	stage after the final stage ($K + 1$)
I	first stage
U	action space for the decision maker
u	individual actions performed by the decision maker
u_k	action during stage k
$U(x_k)$	actions allowed for the decision maker from state x_k
$l(x, u)$	loss (cost) of being in state x and choosing action u
$l_F(x_F)$	cost of reaching the final state
$l_I(x_I)$	initial cost

0.1 Example

There should be link provided for the diagram since I'm unable to get it to work in here.

The loss function is:

$$L_{k,F}^*(x_k) = \min_{u_k \in U(x_k)} L_{k+1,F}^*(x_{k+1}) + l(x_k, u_k)$$

Initially, the values for all states except the goal start at infinity. The algorithm calculates the cost-to-go by starting at the goal and working backwards. In each iteration, the state with the lowest cost-to-go that has not been explored yet is explored to find all possible previous states. The cost-to-go for those states is then calculated and stored in them if it is lower than their current value. Once all states have been explored, the states are all labelled with their optimal cost-to-go.

To find the optimal cost-to-come, a similar algorithm that worked backwards could be used.

1 Sequential Games Against Nature

Additional Notation

$\Theta(x_k)$	actions nature can perform in state x_k
$\Theta(x_k, u_k)$	as above, but nature takes into account u_k
θ_k	actions allowed for nature from state x_k

State Transition Equation: $x_{k+1} = f(x_k, u_k, \theta_k)$ Cost Functional: $L = \sum_{i=1}^K l(x_i, u_i, \theta_i) + l_F(x_F) + l_I(x_1)$

Note: The θ in the cost functional does not need to be there for nature to cause trouble. Nature still helps to decide the next state and also causes trouble that way.

A termination action is used as before and the current state is always known.

1.1 Nondeterministic vs. Probabilistic

Assume current state is x_k and u_k is going to be applied.

Nondeterministic:

$$\theta_k \in \Theta(x_k, u_k)$$

$$F(x_k, u_k) \subseteq X$$

$$F(x_k, u_k) = x_{k+1} | \exists \theta_k \in \Theta(x_k, u_k) \text{ for which } x_{k+1} = f(x_k, u_k, \theta_k)$$

If k equals one, this is one-stage forward projection. Two-stage forward projection would be $F(F(x_1, u_1), u_2)$. Multi-stage forward projection can be done by following the pattern seen in the two-stage equation for F .

Probabilistic:

$$P(\theta_k | x_k, u_k)$$

$$P(x_{k+1} | x_k, u_k)$$

This is the one-stage case again. For a multi-stage forward projection, we would have:

$$P(\theta_k | x_1, x_2, \dots, x_K, u_1, u_2, \dots, u_{K-1}) = P(\theta_k | x_k)$$

Note: It is also possible to include u_{K-1} in the final term above.

The first P can be simplified to the second one by using a Markovian Assumption. This says that the last state contains all the information from the previous states and actions which is necessary to determine their affect on the next action by nature. If the last state does not reveal information about the previous states and actions, then they have no affect on nature's next action. This assumption is made to simplify the state space, but we must ensure that it is statistically valid.

$$P(A|C) = \sum_B P(A|B) * P(B|C)$$

Assume x_1 is known:

$$P(x_{k+2} | x_k, u_k, u_{k+1}) = \sum_{x_{k+1}} P(x_{k+2} | x_{k+1}, u_{k+1}) * P(x_{k+1} | x_k, u_k)$$

As the number of stages of forward projection increases, the amount of uncertainty about the result also increases.

2 Strategies

$$\gamma : X \rightarrow U$$

$$u_k = \gamma(x_k)$$

This is known as Feedback Strategy Control, Conditional Planning, or Reactive Planning. What happens when we fix γ ? Now, the decision maker's actions are determined, but not nature's.

Nondeterministic:

	1	2	3	n
1	0	1	0	1	.	.	.	1
2	1	1	0	1	.	.	.	1
3	1	0	0	0	.	.	.	0
4	0	1	0	0	.	.	.	0
.
.
.
n	0	0	1	1	.	.	.	0

The table above has one row for each possible current state. The columns represent the next state. The entries in each row indicate whether or not it is possible to go from the current state indicated by the row to the next state indicated by the column by using this strategy.

Probabilistic:

	1	2	3	n
1	0.1	0	0.7	0	.	.	.	0
2	0	0.03	0	0.12	.	.	.	0.01
3	0.97	0	0.01	0	.	.	.	0
4	0.13	0	0.46	0	.	.	.	0
.
.
.
n	0.1	0	0.33	0	.	.	.	0.66

Now, instead of indicating whether or not the next state can be attained from the current state, the entries indicate the probability of attaining the next state from the current state if we use this strategy. Here is an example of the formula used to calculate the entry of row 2 column 4:

$$P(x_{k+1} = x_4 | x_k = x_2, u_k = \gamma(x_2))$$

In this case, the outcome is a "parameterized" Markov Chain. The table can also be visualized as a state machine with the probabilities as the weights on the transition arcs. For forward projection, matrix multiplication can be used.

Given a cost functional, L , we want to find γ to minimize the worst case for nondeterministic models or the expected case for probabilistic ones.

3 Backwards Dynamic Programming

3.1 Nondeterministic Cost-to-go

$$L = l_I(x_1) + \sum_{i=1}^K l(x_i, u_i, \theta_i) + l_F(x_F)$$

$$L_{F,F}^*(x_F) = l_F(x_F)$$

$$L_{K,F}^*(x_K) = \min_{u_K} \max_{\theta_K} l(x_K, u_K, \theta_K) + l_F(x_F) \text{ where } x_F = f(x_K, u_K, \theta_K)$$

The equation for $L_{k,F}^*(x_k)$ tries to minimize the worst case. As discussed earlier this semester, other variations of the formula could be used if you are less pessimistic. Continuing with this formula though, we obtain the following equation for forward projection:

$$L_{1,F}^*(x_1) = \min_{u_1} \max_{\theta_1} \min_{u_2} \max_{\theta_2} \dots \min_{u_K} \max_{\theta_K} [l_I(x_1) + \sum_{i=1}^K l(x_i, u_i, \theta_i) + l_F(x_F)]$$

3.2 Probabilistic Cost-to-go

$L_{F,F}^*(x_F) = l_F(x_F)$ as above

$$L_{K,F}^*(x_K) = \min_{u_K} E[l(x_K, u_K, \theta_K) + l_F(x_F)]$$

This gives us the following equation for forward projection:

$$L_{1,F}^*(x_1) = \min_{(u_1, u_2, u_3, \dots, u_K)} E_{(\theta_1, \theta_2, \dots, \theta_K)} [l_I(x_1) + \sum_{i=1}^K l(x_i, u_i, \theta_i) + l_F(x_F)]$$

3.3 Dynamic Programming

The equations above can be transformed so that they can be calculated recursively. This results in the following equations:

Nondeterministic:

$$L_{k,F}^*(x_k) = \min_{u_k \in U(x_k)} \max_{\theta_k \in \Theta(x_k, u_k)} [L_{k+1,F}^*(x_{k+1}) + l(x_k, u_k, \theta_k)]$$

where $k_{k+1} = f(x_k, u_k, \theta_k)$

Probabilistic:

$$L_{k,F}^*(x_k) = \min_{u_k \in U(x_k)} E_{\theta_k} [L_{k+1,F}^*(x_{k+1}) + l(x_k, u_k, \theta_k)]$$

This can be further transformed into the following equation:

$$L_{k,F}^*(x_k) = \min_{u_k \in U(x_k)} \sum_{\theta_k} [L_{k+1,F}^*(x_{k+1}) + l(x_k, u_k, \theta_k)] * P(\theta_k | x_k, u_k)$$

If $l(x_k, u_k, \theta_k) = l(x_k, u_k)$, then we get:

$$L_{k,F}^*(x_k) = \min_{u_k \in U(x_k)} l(x_k, u_k) + \sum_{x_{k+1}} [L_{k+1,F}^*(x_{k+1}) * P(x_{k+1} | x_k, u_k)]$$

Notice that this equation lacks θ , but nature is still present because it helps determine the states.

3.4 Cycles

To prevent the algorithms above from going into infinite loops, cycles in the graphs must obey the following conditions:

Nondeterministic: In the nondeterministic case, there must be no negative cycles in the graph.

Probabilistic: In the probabilistic case, negative cycles can be present, but there must always be a chance that we will escape. If there is a chance to escape the negative cycle, then the probability of remaining in it will decrease over time. This cost will become a geometric series and converge to some finite number instead of increasing to infinity as it would for the nondeterministic case.