Dynamic Programming & Optimal Control

Lecture 3
The Dynamic Programming Algorithm (cont'd)

Fall 2023

Prof. Raffaello D'Andrea

ETH Zurich

Learning Objectives

Topic: The Dynamic Programming Algorithm (cont'd)

Objectives

- You are able to recognize *time lags*, *correlated disturbances* and *forecasts* in dynamic programming problems.
- You know how to convert these *non-standard problems* to *standard problems*.
- You know how to solve the *reformulated* problem with dynamic programming algorithms.

Outline

Example: Optimizing chess match strategy (revisited)

Consider a two game chess match with an opponent. Our objective is to come up with a strategy that maximizes the chance of winning the match.

Each game can have one of two outcomes:

1. Win/Lose: 1 point for the winner, 0 for the loser
2. Tie: 0.5 points for each player

If at the end of two games the score is equal, the players keep on playing new games until one wins, and thereby wins the match (sudden death).

There are two possible playing styles for our player: timid and bold.

- Timid: our player ties with probability p_d and loses with probability $(1 - p_d)$
- Bold: our player wins with probability p_w and loses with probability $(1 - p_w)$

Assume that $p_d > p_w$.

Solution: chess match strategy (1/8)

In order to apply the DPA, we must convert this problem to the standard problem formulation:

- A possible choice for the *state* x_k is the difference between our player's score and the opponent's score at the end of game k:

$$\begin{aligned} &x_0 = 0, \\ &x_1 \in \mathcal{S}_1 = \{-1, 0, 1\}, \\ &x_2 \in \mathcal{S}_2 = \{-2, -1, 0, 1, 2\}. \end{aligned}$$

- The *control inputs* u_k are the two playing styles:

$$u_k \in \mathcal{U} = \{\text{timid}, \text{bold}\}.$$

Solution: Chess match strategy (2/8)

- *Dynamics:* model as a finite state system using transition probabilities:

$$x_{k+1} = w_k, \quad k = 0, 1$$

where

$$\begin{aligned}
&\Pr\left(w_k = x_k | u_k = \text{timid}\right) = p_d \\
&\Pr\left(w_k = x_k - 1 | u_k = \text{timid}\right) = 1 - p_d \\
&\Pr\left(w_k = x_k + 1 | u_k = \text{bold}\right) = p_w \\
&\Pr\left(w_k = x_k - 1 | u_k = \text{bold}\right) = 1 - p_w
\end{aligned}$$

with $p_d > p_w$.

Solution: Chess match strategy (3/8)

- *Cost*: we want to maximize the probability of winning. This is equivalent to maximizing the expected value of this cost function

$$g_2(x_2) + \sum_{k=0}^{1} g_k(x_k, u_k, w_k)\,, \tag{1}$$

where

$$g_k(x_k, u_k, w_k) = 0, \quad \forall k \in \{0, 1\}$$

$$g_2(x_2) = \begin{cases} 1 & \text{if } x_2 > 0 \\ p_w & \text{if } x_2 = 0 \\ 0 & \text{if } x_2 < 0 \end{cases}.$$

Solution: Chess match strategy (4/8)

$$g_k(x_k, u_k, w_k) = 0, \quad \forall k \in \{0, 1\}$$

$$g_2(x_2) = \begin{cases} 1 & \text{if } x_2 > 0 \\ p_w & \text{if } x_2 = 0 \\ 0 & \text{if } x_2 < 0 \end{cases}.$$

To see that the expected value of (1) is equal to the probability of winning P_{win}, let

$$q_+ := \Pr(x_2 > 0), \quad q_0 := \Pr(x_2 = 0), \quad q_- := \Pr(x_2 < 0).$$

The probability of winning is

$$P_{\text{win}} = q_+ + q_0 p_w,$$

and the expected value of the cost is

$$\mathrm{E}[g_2(x_2)] = q_+ \cdot 1 + q_0 \cdot p_w + q_- \cdot 0 = q_+ + q_0 p_w.$$

Solution: Chess match strategy (5/8)

Now apply DPA:

Initialization:

$$J_2(\mathrm{x}) = \begin{cases} 1 & \text{if } \mathrm{x} > 0 \\ p_w & \text{if } \mathrm{x} = 0 \\ 0 & \text{if } \mathrm{x} < 0 \end{cases}.$$

Recursion:

$$\begin{aligned} J_k(\mathrm{x}) &= \max_{\mathrm{u}\in\mathcal{U}} \mathop{\mathrm{E}}_{(w_k|x_k=\mathrm{x},u_k=\mathrm{u})} \left[g_k(x_k,u_k,w_k) + J_{k+1}(x_{k+1})\right],\ \forall \mathrm{x}\in\mathcal{S}_k,\ \forall k\in\{0,1\} \\ &= \max_{\mathrm{u}\in\mathcal{U}} \mathop{\mathrm{E}}_{(w_k|x_k=\mathrm{x},u_k=\mathrm{u})} \left[J_{k+1}(w_k)\right] \\ &= \max \Bigg\{ \underbrace{p_d J_{k+1}(\mathrm{x}) + (1-p_d) J_{k+1}(\mathrm{x}-1)}_{\text{timid}}, \\ &\qquad\qquad \underbrace{p_w J_{k+1}(\mathrm{x}+1) + (1-p_w) J_{k+1}(\mathrm{x}-1)}_{\text{bold}} \Bigg\}. \end{aligned}$$

Solution: Chess match strategy (6/8)

Henceforth, the first entry of the maximum will denote the cost associated with timid play and the second with bold play.

$\underline{k = 1}$:

$$J_1(\mathrm{x}) = \max\left\{p_d J_2(\mathrm{x}) + (1-p_d)J_2(\mathrm{x}-1)\,,\ p_w J_2(\mathrm{x}+1) + (1-p_w)J_2(\mathrm{x}-1)\right\}$$

- $x_1 = 1$:

$$J_1(1) = \max\left\{p_d + (1-p_d)p_w,\ p_w + (1-p_w)p_w\right\}$$

Comparing the two entries yields:

$$\begin{aligned}&(p_d + (1-p_d)p_w) - (p_w + (1-p_w)p_w)\\ &= (p_d - p_w)(1-p_w)\ > 0 \text{ (since } p_d > p_w).\end{aligned}$$

Therefore $\mu_1^*(1) =$ timid and $J_1(1) = p_d + (1-p_d)p_w$.

Solution: Chess match strategy (7/8)

- $x_1 = 0$:

  $$J_1(0) = \max\{p_d p_w + (1 - p_d) \cdot 0,\ p_w + (1 - p_w) \cdot 0\} = \max\{p_d p_w,\ p_w\}$$

  Therefore $\mu_1^*(0) =$ bold and $J_1(0) = p_w$.

- $x_1 = -1$:

  $$J_1(-1) = \max\{p_d \cdot 0 + (1 - p_d) \cdot 0,\ p_w p_w + (1 - p_w) \cdot 0\} = \max\{0,\ p_w^2\}$$

  Therefore $\mu_1^*(-1) =$ bold and $J_1(-1) = p_w^2$.

Solution: Chess match strategy (8/8)

$\underline{k = 0}$:

$$J_0(\mathrm{x}) = \max\left\{p_d J_1(\mathrm{x}) + (1-p_d)J_1(\mathrm{x}-1)\,,\ p_w J_1(\mathrm{x}+1) + (1-p_w)J_1(\mathrm{x}-1)\right\}$$

- $x_0 = 0$:

$$\begin{aligned} J_0(0) &= \max\left\{p_d J_1(0) + (1-p_d)J_1(-1)\,,\ p_w J_1(1) + (1-p_w)J_1(-1)\right\} \\ &= \max\left\{p_d p_w + (1-p_d)p_w^2,\ p_w(p_d + (1-p_d)p_w) + (1-p_w)p_w^2\right\} \\ &= \max\left\{p_d p_w + (1-p_d)p_w^2,\ p_d p_w + (1-p_d)p_w^2 + (1-p_w)p_w^2\right\} \end{aligned}$$

Therefore $\mu_0^*(0) =$ bold and $J_0(0) = p_d p_w + (1-p_d)p_w^2 + (1-p_w)p_w^2$.

Optimal match strategy: Play timid if and only if ahead in score.

Outline

Time Lags

Assume the dynamics have the following form:

$$x_{k+1} = f_k\left(x_k, x_{k-1}, u_k, u_{k-1}, w_k\right).$$

- Let $y_k := x_{k-1}$, $s_k := u_{k-1}$, and the augmented state vector $\tilde{x}_k := (x_k, y_k, s_k)$.
- The dynamics of the augmented state then become

$$\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} f_k\left(x_k, y_k, u_k, s_k, w_k\right) \\ x_k \\ u_k \end{bmatrix} =: \tilde{f}_k\left(\tilde{x}_k, u_k, w_k\right),$$

  which now matches the standard form.

Note that this procedure works for an arbitrary number of time lags.

Correlated Disturbances

Disturbances w_k that are correlated across time (colored noise) can commonly be modeled as the output of a linear system driven by independent random variables:

$$w_k = C_k y_{k+1}$$
$$y_{k+1} = A_k y_k + \xi_k$$

where A_k, C_k are given and $\xi_k,\ k = 0, \ldots, N-1$, are independent random variables.

- Let the augmented state vector $\tilde{x}_k := (x_k, y_k)$. Note that now y_k must be observed at time k, which can be done using a state estimator.
- The dynamics of the augmented state then become

$$\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k\left(x_k, u_k, C_k(A_k y_k + \xi_k)\right), \\ A_k y_k + \xi_k, \end{bmatrix} =: \tilde{f}_k\left(\tilde{x}_k, u_k, \xi_k\right)$$

  which now matches the standard form.

Forecasts (1/4)

Consider a scenario in which at each time period we have access to a forecast that reveals the probability distribution of w_k and possibly of future disturbances.

For example, assume that w_k is independent of x_k and u_k.

At the beginning of each period k, we receive a prediction y_k (forecast) that w_k will attain a probability distribution out of a given finite collection of distributions $\left\{p_{w_k|y_k}(\cdot|1), p_{w_k|y_k}(\cdot|2), ..., p_{w_k|y_k}(\cdot|m)\right\}$.

In particular, we receive a forecast that $y_k = i$ and thus $p_{w_k|y_k}(\cdot|i)$ is used to generate w_k.

Forecasts (2/4)

The forecast itself has a given *a-priori* probability distribution:

$$y_{k+1} = \xi_k,$$

where the ξ_k are independent random variables taking value $i \in \{1, 2, ..., m\}$ with probability $p_{\xi_k}(i)$.

- Let the augmented state vector $\tilde{x}_k := (x_k, y_k)$. Since the forecast y_k is known at time k, we still have perfect state information.
- We define our new disturbance as $\tilde{w}_k := (w_k, \xi_k)$, with probability distribution

$$\begin{aligned} p(\tilde{w}_k|\tilde{x}_k, u_k) &= p(w_k, \xi_k|x_k, y_k, u_k) \\ &= p(w_k|x_k, y_k, u_k, \xi_k)\, p(\xi_k|x_k, y_k, u_k) \\ &= p(w_k|y_k)\, p(\xi_k)\,. \end{aligned}$$

  Note that w_k depends only on $\tilde{x}_k$ (in particular y_k), and ξ_k does not depend on anything.

Forecasts (3/4)

- The dynamics therefore become

$$\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k\left(x_k, u_k, w_k\right) \\ \xi_k \end{bmatrix} =: \tilde{f}_k\left(\tilde{x}_k, u_k, \tilde{w}_k\right),$$

  which now matches the standard form.

The associated DPA becomes:

Initialization:

$$J_N(\tilde{\mathrm{x}}) = J_N(\mathrm{x}, \mathrm{y}) = g_N(\mathrm{x}), \quad \mathrm{x} \in \mathcal{S}_N,\ \mathrm{y} \in \{1, \ldots, m\}.$$

Recursion:

$$\begin{aligned} J_k(\tilde{\mathrm{x}}) &= J_k(\mathrm{x}, \mathrm{y}) \\ &= \min_{\mathrm{u} \in \mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(\tilde{w}_k | \tilde{x}_k = \tilde{\mathrm{x}}, u_k = \mathrm{u})} \left[g_k(x_k, u_k, w_k) + J_{k+1}(f_k\left(x_k, u_k, w_k\right), \xi_k) \right]. \end{aligned}$$

Since $p(\tilde{w}_k | \tilde{x}_k, u_k) = p(w_k | y_k)\, p(\xi_k)$,

$$J_k(\tilde{\mathrm{x}}) = \min_{\mathrm{u} \in \mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(w_k | y_k = \mathrm{y})} \left[\mathop{\mathrm{E}}_{\xi_k} \left[g_k(\mathrm{x}, \mathrm{u}, w_k) + J_{k+1}(f_k\left(\mathrm{x}, \mathrm{u}, w_k\right), \xi_k) \right] \right].$$

Forecasts (4/4)

$$J_k(\tilde{\mathrm{x}}) = \min_{\mathrm{u}\in\mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(w_k|y_k=\mathrm{y})} \left[\mathop{\mathrm{E}}_{\xi_k} \left[g_k(\mathrm{x},\mathrm{u},w_k) + J_{k+1}(f_k(\mathrm{x},\mathrm{u},w_k),\xi_k) \right] \right].$$

Since $g_k(x_k,u_k,w_k)$ is not a function of the random variable ξ_k,

$$\begin{aligned} J_k(\tilde{\mathrm{x}}) &= \min_{\mathrm{u}\in\mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(w_k|y_k=\mathrm{y})} \left[g_k(\mathrm{x},\mathrm{u},w_k) + \mathop{\mathrm{E}}_{\xi_k} \left[J_{k+1}(f_k(\mathrm{x},\mathrm{u},w_k),\xi_k) \right] \right] \\ &= \min_{\mathrm{u}\in\mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(w_k|y_k=\mathrm{y})} \left[g_k(\mathrm{x},\mathrm{u},w_k) + \sum_{i=1}^{m} p_{\xi_k}(i)\, J_{k+1}(f_k(\mathrm{x},\mathrm{u},w_k),i) \right] \end{aligned}$$

$$\forall \mathrm{x}\in\mathcal{S}_k,\ \forall \mathrm{y}\in\{1,\ldots,m\},\ \forall k = N-1,\ldots,0.$$

Additional reading material

Sometimes, even if we can reformulate the problem in the standard form, it is computationally too demanding. However, the DPA suggests approximations that have proven very successful in recent years:

$$J_k(\mathrm{x}) := \min_{\mathrm{u}\in\mathcal{U}_k(\mathrm{x})} \mathop{\mathrm{E}}_{(w_k|x_k=\mathrm{x},u_k=\mathrm{u})} \Big[g_k(x_k,u_k,w_k) + J_{k+1}(f_k(x_k,u_k,w_k)) \Big],$$
$$\forall \mathrm{x} \in \mathcal{S}_k,\ k = N-1,\dots,0.$$

- Computing the expectation may be cumbersome. Ideas: Monte Carlo method, Certainty Equivalence.
- Minimizing may be cumbersome. Ideas: Gradient descent, Sampling.
- If we cannot proceed backwards (too many terminal states), we can sample forward: Monte Carlo Tree Search, Multi-step look-ahead, Cost-to-go approximation.

If you are curious about these methods, there is plenty of material available online: Try to search these keywords!