

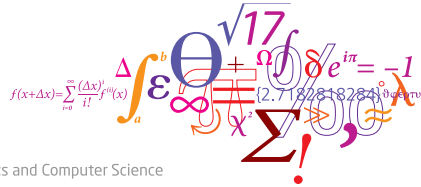
02465: Introduction to reinforcement learning and control

Dynamical Programming

Tue Herlau

DTU Compute, Technical University of Denmark (DTU)

DTU Compute
Department of Applied Mathematics and Computer Science



Lecture Schedule

Dynamical programming

- 1 The finite-horizon decision problem
7 February
- 2 **Dynamical Programming**
14 February
- 3 DP reformulations and introduction to Control
21 February
- 4 Discretization and PID control
28 February
- 5 Direct methods and control by optimization
7 March
- 6 Linear-quadratic problems in control
14 March
- 7 Linearization and iterative LQR
21 March

Reinforcement learning

- 8 Exploration and Bandits
28 March
- 9 Bellmans equations and exact planning
4 April
- 10 Monte-carlo methods and TD learning
11 April
- 11 Model-Free Control with tabular and linear methods
25 April
- 12 Eligibility traces
2 May
- 13 Deep-Q learning
9 May

Syllabus: <https://02465material.pages.compute.dtu.dk/02465public>
Help improve lecture by giving feedback on DTU learn

Reading material:

- [Her25, Chapter 5-6.2] Formalization of the decision problem and the DP algorithm

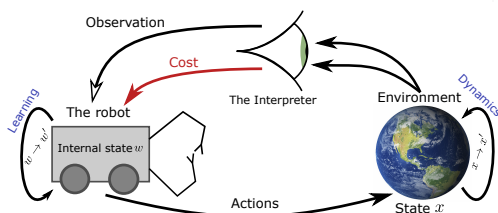
Learning Objectives

- Dynamical Programming
- Principle of optimality
- Optimal policy/value function using DP

Practicals

- Issue with recording 1
- Numpy-core issue fixed (old numpy; see guide)
- Fixed page numbering (thanks!)

The decision problem

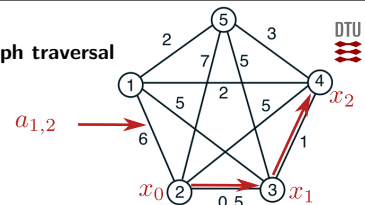


State The configuration of the environment x

Action What we do u

Cost/reward A number which depends on the state and action

Example: Shortest path graph traversal



Find shortest path from starting node $x_0 = 2$ to final node $t = 5$

State Current node $x_k = 4$

Actions next possible node: $u_k \in \{1, 2, \dots, 5\}$

Dynamics Deterministic, known

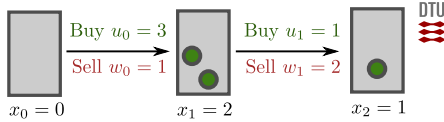
$$x_{k+1} = f(x_k = 4, u_k = 5) = 5$$

Cost Sum of edge weights

$$\sum_{k=0}^{N-1} a_{x_k, u_k} + \begin{cases} 0 & \text{if } x_N = t \\ \infty & \text{otherwise} \end{cases}$$

We want optimal path $\{2, 3, 4, 5\}$

Inventory control



- We order a quantity of an item at period $k = 0, \dots, N$ so as to meet a stochastic demand

x_k stock available at the beginning of the k th period,
 $u_k \geq 0$ stock ordered (and immediately delivered) at the beginning of the k th period.
 $w_k \geq 0$ Demand during the k 'th period

- Dynamics: $x_{k+1} = x_k + u_k - w_k$
- Cost function (in each step)

$$u_k + (x_k + u_k - w_k)^2$$

- Select actions u_0, \dots, u_{N-1} to minimize cost

We want proven optimal rule for ordering

The basic problem

Basic control setup: Environment dynamics

Finite time Problem starts at time 0 and terminates at *fixed* time N . Indexed as $k = 0, 1, \dots, N$.

State space The states x_k belong to the **state space** S_k

Control The available controls u_k belong to the **action space** $\mathcal{A}_k(x_k)$, which may depend on x_k

Dynamics

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \dots, N-1$$

Disturbance/noise A random quantity w_k with distribution

$$w_k \sim P_k(W_k | x_k, u_k)$$

The basic problem

Cost and control

Agent observe x_k , agent choose u_k , environment generates w_k

Cost At each stage k we obtain cost

$$g_k(x_k, u_k, w_k), \quad k = 0, \dots, N-1 \quad \text{and} \quad g_N(x_N) \text{ for } k = N.$$

Action choice Chosen as $u_k = \mu_k(x_k)$ using a function $\mu_k : S_k \rightarrow \mathcal{A}_k(x_k)$

$$\mu_k(x_k) = \{\text{Action to take in state } x_k \text{ in period } k\}$$

Policy The collection $\pi = \{\mu_0, \mu_1, \dots, \mu_{N-1}\}$

Rollout of policy Given x_0 , select $u_k = \mu_k(x_k)$ to obtain a **trajectory** $x_0, u_0, x_1, \dots, x_N$ and **accumulated cost**

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)$$

The basic problem

Expected cost/value function

Expected cost Given π , x_0 it is the average cost of all trajectories:

$$J_\pi(x_0) = \mathbb{E} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]$$

Optimal policy Given x_0 , an optimal policy π^* is one that minimizes the cost

$$\pi^*(x_0) = \arg \min_{\pi = \{\mu_0, \dots, \mu_{N-1}\}} J_\pi(x_0)$$

Optimal cost function The optimal cost, given x_0 , is denoted $J^*(x_0)$ and is defined as

$$J^*(x_0) = \min_{\pi = \{\mu_0, \dots, \mu_{N-1}\}} J_\pi(x_0)$$

J_π is the key quantity in control/reinforcement learning

The basic problem

Open versus closed loop

Our goal is to find the policy π which minimize:

$$J_\pi(x_0) = \mathbb{E} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]$$

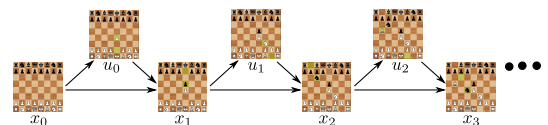
Closed-loop minimization Select u_k last-minute as $u_k = \mu_k(x_k)$ when information x_k is available

Open-loop minimization Select actions u_0, \dots, u_{N-1} at $k = 0$

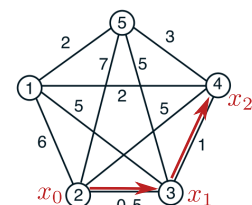
- Open-loop minimization is simpler

The basic problem

Open or closed loop



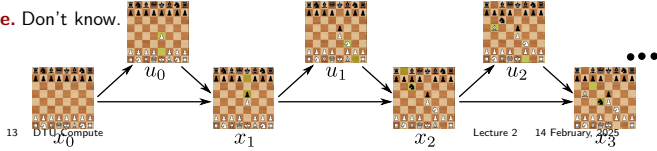
- If environment is stochastic, we need a closed-loop controller
- If environment is deterministic, we know the position x_k with certainty given u_0, \dots, u_{k-1} . Therefore, there is no advantage in delaying choice



Suppose the game of chess was formulated as dynamical programming (N , S_k , \mathcal{A}_k , etc.) with the intention of obtaining a good policy μ_k using dynamical programming.

This will lead to several practical problems, however, focusing just on the potential problems listed below, which one will be a main obstacle?

- The policy function μ_k will require too much memory to store
- Given a state x_k , it is not practical to define the action spaces $\mathcal{A}_k(x_k)$
- It will require too much space to store the state space S_2 .
- We cannot define a meaningful cost function g_k .
- Don't know.



13 DTU Compute

Lecture 2 14 February, 2025

Summary: Discrete stochastic decision problem

- The states are x_0, \dots, x_N , and the controls are u_0, \dots, u_{N-1}
- $w_k \sim P_k(W_k = w_k | x_k, u_k)$, $k = 0, \dots, N-1$ are random disturbances
- The system evolves as

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \quad k = 0, \dots, N-1$$

- At time k , the possible states/actions are $x_k \in S_k$ and $u_k \in \mathcal{A}_k(x_k)$
- Policy is a sequence of functions $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, $\mu_k : S_k \mapsto \mathcal{A}_k(x_k)$
- The cost starting in x_0 is:

$$J_\pi(x_0) = \mathbb{E} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]$$

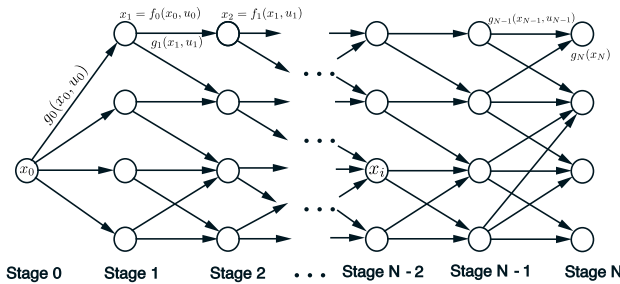
- The control problem:** Given x_0 , determine optimal policy by minimizing

$$\pi^*(x_0) = \arg \min_{\pi = \{\mu_0, \dots, \mu_{N-1}\}} J_\pi(x_0)$$

14 DTU Compute

Lecture 2 14 February, 2025

Starting in x_0 , decision problem can be seen as traversing a graph

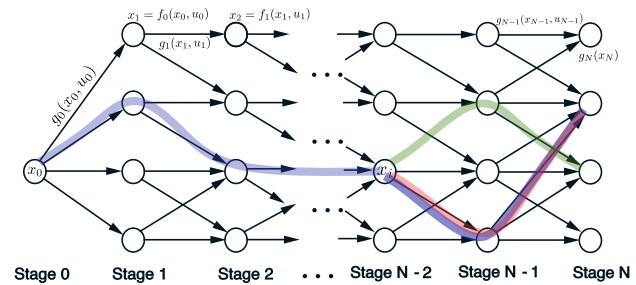


- Nodes are states, edges are possible transitions, cost is sum of edges
- In deterministic case, actions are edges and a policy is just a path

15 DTU Compute

Lecture 2 14 February, 2025

Principle of optimality (PO), deterministic case



The blue line is a path corresponding to an optimal policy

$$J^*(x_0) = J_{\pi^*}(x_0) = \min_{\pi} J_{\pi}(x_0)$$

Suppose at stage i optimal path $\pi^* = \{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$ pass through x_i

- PO:** The tail policy $\{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$ is optimal from x_i to x_N

16 DTU Compute

Lecture 2 14 February, 2025

For any policy $\pi = \{\mu_0, \mu_1, \dots, \mu_{N-1}\}$

- For any $k = 0, \dots, N-1$, $\pi^k = \{\mu_k, \mu_{k+1}, \dots, \mu_{N-1}\}$ is a **tail policy**
- For any x_k the **cost of the tail policy** is

$$J_{k,\pi}(x_k) = \mathbb{E} \left[g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right]$$

- And the **optimal cost of a tail policy** starting in x_k

$$J_k^*(x_k) = \min_{\pi^k} J_{k,\pi^k}(x_k)$$

- Note that $J_0^*(x_0) = J^*(x_0)$

17 DTU Compute

Lecture 2 14 February, 2025

Proof of PO in deterministic case

$$J_{\pi^*}(x_0) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k^*(x_k)) =$$

$$\begin{aligned} & \left(\sum_{k=0}^{i-1} g_k(x_k, \mu_k^*(x_k)) \right) + \left(g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k^*(x_k)) \right) \\ & \geq \left(\sum_{k=0}^{i-1} g_k(x_k, \mu_k^*(x_k)) \right) + g_N(x'_N) + \sum_{k=i}^{N-1} g_k(x'_k, \mu'_k(x'_k)) \\ & = J_{\pi'}(x_0) \end{aligned}$$

If the optimal tail policy π'_i had a lower tail cost than the tail of optimal policy this means:

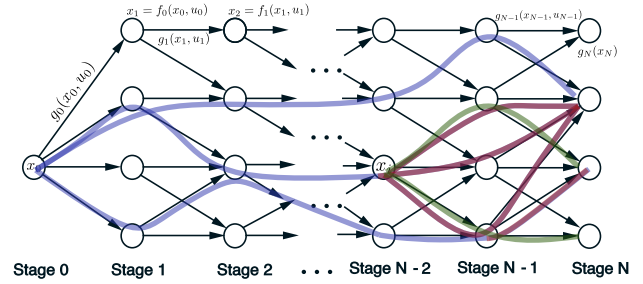
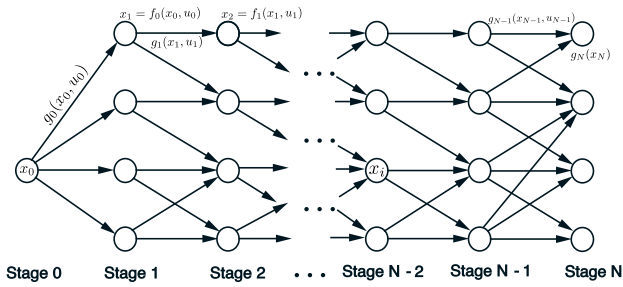
$$g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k^*(x_k)) > g_N(x'_N) + \sum_{k=i}^{N-1} g_k(x'_k, \mu'_k(x'_k))$$

and so the combined policy $(\mu_0, \dots, \mu_{i-1}, \pi'_i)$ would have lower cost than optimal policy π^*

18 DTU Compute

Lecture 2 14 February, 2025

Consider the stochastic case. Trajectories are now random



- Consider **tail policy** of π^* : $J_{i,\pi^*}(x_i)$
- Suppose **optimal tail policy** $J_i^*(x_i)$ is an improvement
- It seems true the combined policy is an improvement over π^* [Her25, appendix A]

Consider a general, stochastic/discrete finite-horizon decision problem

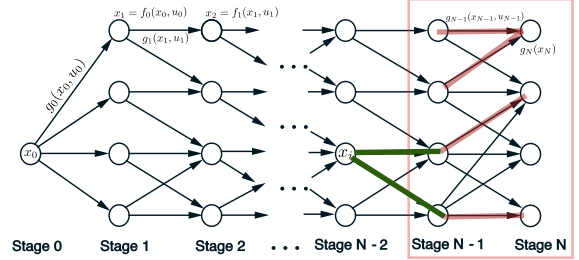
The principle of optimality

Let $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$ be an optimal policy for the problem, and assume that when using π^* , a given state x_i occurs at stage i with positive probability. Suppose $\tilde{\pi}_k^*$ is the optimal tail policy obtained by minimizing the tail cost starting from x_i

$$J_{k,\pi}(x_i) = \mathbb{E} \left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}.$$

Then the truncated policy $\{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$ of π^* is optimal for the tail problem

$$J_{k,\tilde{\pi}_k^*}(x_k) = J_{k,\pi^*}(x_k).$$



- Suppose we know the optimal tail policy π_{k+1}^* for all x_{k+1}
- Cost of optimal path π_k^* from k to N is the cost of optimal path $x_k \rightarrow x_{k+1}$ and then $x_{k+1} \rightarrow x_N$
- The later part is the same as $J_{k+1}^*(x_{k+1})$ by the **PO**
- We find optimal cost by minimizing

$$J_k^*(x_k) = \min_{u_k \in \mathcal{A}_k(x_k)} [g_k(x_k, u_k) + J_{k+1}^*(x_{k+1})], \quad \mu_k(x_k) = u_k^*$$

The Dynamical Programming algorithm

For every initial state x_0 , the optimal cost $J^*(x_0)$ is equal to $J_0(x_0)$, and optimal policy π^* is $\pi^* = \{\mu_0, \dots, \mu_{N-1}\}$, computed by the following algorithm, which proceeds backward in time from $k = N$ to $k = 0$ and for each $x_k \in S_k$ computes

$$J_N(x_N) = g_N(x_N) \quad (1)$$

$$J_k(x_k) = \min_{u_k \in \mathcal{A}_k(x_k)} \mathbb{E}_{w_k} \{g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\} \quad (2)$$

$$\mu_k(x_k) = u_k^* \quad (u_k^* \text{ is the } u_k \text{ which minimizes the above expression}). \quad (3)$$

- There are N μ 's and $N + 1$ J 's. This will also be the case in the code
- In the deterministic case:

$$J_k(x_k) = \min_{u_k \in \mathcal{A}_k(x_k)} \{g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k))\}$$

- Consider the inventory control problem where we plan over $N = 3$ stages
- Customers can buy $w_k = 0$ to $w_k = 2$ units and we can order $u_k = 0$ to $u_k = 2$ units
- We assume the stock can hold from 0 to 2 units (no excess stock; no backlog)
- The cost to buy an item is 1 plus quadratic penalty for excess stock and unmet demand:
- There is no terminal cost $g_N(x_N) = 0$
- The demand has distribution

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k \quad (\text{threshold s.t. } 0 \leq x_k \leq 2)$$

$$u_k + (x_k + u_k - w_k)^2$$

$$p(w_k = 0) = 0.1, \quad p(w_k = 1) = 0.7, \quad p(w_k = 2) = 0.2$$

Implementation

```

1 # inventory.py
2 class InventoryDPModel(DPModel):
3     def __init__(self, N=3):
4         super().__init__(N=N)
5
6     def A(self, x, k): # Action space A_k(x)
7         return {0, 1, 2}
8
9     def S(self, k): # State space S_k
10        return {0, 1, 2}
11
12    def g(self, x, u, w, k): # Cost function g_k(x,u,w)
13        return u + (x + u - w) ** 2
14
15    def f(self, x, u, w, k): # Dynamics f_k(x,u,w)
16        return max(0, min(2, x + u - w))
17
18    def Pw(self, x, u, k): # Distribution over random disturbances
19        return {0:.1, 1:.7, 2:0.2}
20
21    def gN(self, x):
22        return 0

```

Option 1: Pen-and-paper

First step: $J_3(x_3) = 0$ (for all x_3)

Step $k = 2$ For $x_2 = 0$

$$\begin{aligned}
 J_2(0) &= \min_{u_2=0,1,2} \mathbb{E} \left\{ u_2 + (u_2 - w_2)^2 \right\} \\
 &= \min_{u_2=0,1,2} \left[u_2 + 0.1(u_2)^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2 \right] \\
 &= \min_{u_2=0,1,2} \{ 0.7 \cdot 1 + 0.2 \cdot 4, 1 + 0.1 \cdot 1 + 0.2 \cdot 1, 2 + 0.1 \cdot 4 + 0.7 \cdot 1 \} \\
 &= \min_{u_2=0,1,2} \{ 1.5, 1.3, 3.1 \}
 \end{aligned}$$

Therefore $\mu_2^*(0) = 1$ and $J_2^*(0) = 1.3$

Until nails bleed Keep at it for $x_2 = 1, 2$ and then for $k = 1$ and finally $k = 0 \dots$

Quiz: Manual DP

Suppose that for a given k :

- $\mathcal{A}_k(x_k) = \{0, 1\}$, $f_k(x_k, u_k, w_k) = x_k + u_k w_k$
- $g_k(x_k, u_k, w_k) = -x_k u_k$, $J_{k+1}(x_{k+1}) = x_{k+1}$
- $\mathbb{E}[w_k] = 1$

What is the value of $J_k(x_k = 1)$? **Tip:**

$$J_k(x_k) = \min_{u_k \in \mathcal{A}_k(x_k)} \mathbb{E}_{w_k} \{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \}$$

- $J_k(1) = -2$
- $J_k(1) = -1$
- $J_k(1) = 0$
- $J_k(1) = 1$
- $J_k(1) = 2$
- Don't know.

Option 2: Computer

```

1 # inventory.py
2 inv = InventoryDPModel()
3 J, pi = DP_stochastic(inv)
4 print(f"Inventory control optimal policy/value functions")
5 for k in range(inv.N):
6     print(f"J_{k}({x}_{k}={i}) = {J[k][i]:.2f}" for i in inv.S(k) ) )
7 for k in range(inv.N):
8     print(f"pi_{k}({x}_{k}={i}) = {pi[k][i]} for i in inv.S(k) ) )

```

```

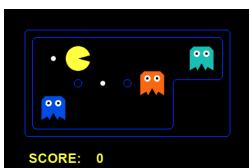
1 Inventory control optimal policy/value functions
2 J_0(x_0=0) = 3.70, J_0(x_0=1) = 2.70, J_0(x_0=2) = 2.82
3 J_1(x_1=0) = 2.50, J_1(x_1=1) = 1.50, J_1(x_1=2) = 1.68
4 J_2(x_2=0) = 1.30, J_2(x_2=1) = 0.30, J_2(x_2=2) = 1.10
5 pi_0(x_0=0) = 1, pi_0(x_0=1) = 0, pi_0(x_0=2) = 0
6 pi_1(x_1=0) = 1, pi_1(x_1=1) = 0, pi_1(x_1=2) = 0
7 pi_2(x_2=0) = 1, pi_2(x_2=1) = 0, pi_2(x_2=2) = 0

```

lecture_02_optimal_dp_g1.py

lecture_02_optimal_inventory.py

Project 1: Pacman



How?

- Define a DP model
- Apply the DP algorithm

```

1 # chapter3dp/pacman_dp_excerpt.py
2 from irlec.pacman.pacman_environment import PacmanEnv
3 from irlec.pacman.gamestate import GameState
4 from irlec.ex02.dp_model import DPModel
5 from irlec.ex02.dp import DP_stochastic
6
7 class MyPacmanDPModel(DPModel):
8     def __init__(self, env, N : int):
9         self.env = env
10        super().__init__(N)
11
12    def A(self, x : GameState, k):
13        # See online documentation!
14        return x.A()
15
16    # remember f, g, gN, S, ...
17
18 if __name__ == "__main__":
19     model = MyPacmanDPModel(PacmanEnvironment())
20     J, pi = DP_stochastic(model)

```

https://www2.compute.dtu.dk/courses/02465/models/models_dp.html

https://www2.compute.dtu.dk/courses/02465/models/week2_pacman_g1.html

Part 1 of the project

- you should be all set!



Information

Models and Environments

Exercises

Projects

Project 1: Dynamical Programming


Project 2: Control theory

Project 1: Dynamical Programming

Note		
When?	Thursday	Before 23:59
What?	To get started, download the project description here: 02465project1.pdf	
Where?	Under assignments on DTU Learn 02465	
What to hand in?	(see project description)	
	<ul style="list-style-type: none"> irlec/project1/Project1_handin_k_of_n.token irlec/project1/Latex/02465project1_handin.tex irlec/project1/Latex/02465project1_handin.pdf 	

Consult the project description (above) for details about the problems. To get the newest version of the course material, please see [Making sure your files are up to date](#).

Creating your hand-in

 Tue Herlau.
Sequential decision making.
(Freely available online), 2025.