46765 – Machine learning for energy systems

Markov Decision Process for Energy Asset Scheduling and Control

Farzaneh Pourahmadi

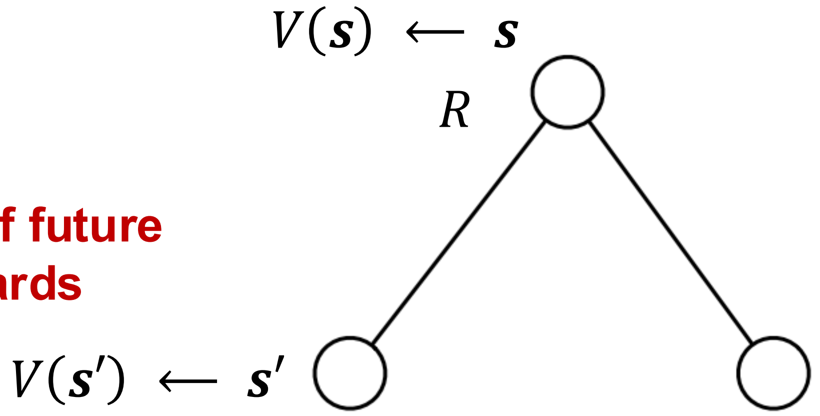
Recap

Bellman equation without actions

Let us consider a case where rewards and transition probabilities only depend on states:

$$V(\boldsymbol{s}) = E[R(\boldsymbol{s}_0) + \gamma R(\boldsymbol{s}_1) + \gamma^2 R(\boldsymbol{s}_2) + \cdots | \boldsymbol{s}_0 = \boldsymbol{s}]$$

$$V(\boldsymbol{s}) = \underbrace{\boxed{R(\boldsymbol{s})}}_{\text{Immediate reward}} + \underbrace{\boxed{\gamma \sum_{\boldsymbol{s}' \in S} P_s(\boldsymbol{s}') V(\boldsymbol{s}')}}_{\text{Expected sum of future discounted rewards}}$$


$V(\boldsymbol{s}) \leftarrow \boldsymbol{s}$
R
$V(\boldsymbol{s}') \leftarrow \boldsymbol{s}'$

Value function can be expressed in a vector format

$$V = R + \gamma P V$$

It can be calculated directly by

$$V = (I - \gamma P)^{-1} R$$

Introduction to Markov decision process

A Markov decision process is a tuple $(S, A, \{p_{\boldsymbol{sa}}\}, \gamma, R)$.

1. S is a set of states ($\boldsymbol{s} \in S$).
2. A is a set of actions ($\boldsymbol{a} \in A$).

$\xrightarrow{\quad}$ You have seen how to define states and actions!

3. $\{p_{\boldsymbol{sa}}\}$ are the state transition probabilities. For each state $\boldsymbol{s} \in S$ and action $\boldsymbol{a} \in A$, $p_{\boldsymbol{sa}}$ is a distribution over the state space. In other words, $p_{\boldsymbol{sa}}$ gives the distribution over what states we will transition to if we take action $\boldsymbol{a}$ in state $\boldsymbol{s}$. For problems that actions will not impact the uncertainty of next sate, we only have $\{p_{\boldsymbol{s}}\}$.
4. $R(\boldsymbol{s})$ or $R(\boldsymbol{s}, \boldsymbol{a}) \in \mathbb{R}$ is the reward function.
5. $\gamma \in [0,1]$ is called the discount factor.

Policies

- A policy is any function $\boldsymbol{\pi}: S \rightarrow A$ mapping from the states to the actions.
- We say that we are executing some policy $\boldsymbol{\pi}$ if, whenever we are in state s, we take action $\boldsymbol{a} = \boldsymbol{\pi}(\boldsymbol{s})$.
- A policy fully defines the behavior of an agent.
- MDP policies depend on the current state (not the history)
- In this course, we assume policies are stationary (time-independent).
- A policy $\boldsymbol{\pi}$ can be both deterministic or stochastic representing a distribution over actions given states. Here, we only consider deterministic policies.

State transition matrix with action

If the agent is in Markov state $\boldsymbol{s}$ and take action $\boldsymbol{a}$, the state transition probability to successor state $\boldsymbol{s}'$ is defined by

$$p_{\boldsymbol{s}\boldsymbol{a}}(\boldsymbol{s}') = \mathbb{P}[\boldsymbol{s}_{t+1} = \boldsymbol{s}' | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}]$$

State transition matrix P with considering policy π is defined by

$$P^{\pi} = \begin{bmatrix} p_{\boldsymbol{s}_1 \boldsymbol{\pi}(\boldsymbol{s}_1)}(\boldsymbol{s}_1) & \dots & p_{\boldsymbol{s}_1 \boldsymbol{\pi}(\boldsymbol{s}_1)}(\boldsymbol{s}_n) \\ \vdots & & \vdots \\ p_{\boldsymbol{s}_n \boldsymbol{\pi}(\boldsymbol{s}_n)}(\boldsymbol{s}_1) & \dots & p_{\boldsymbol{s}_n \boldsymbol{\pi}(\boldsymbol{s}_n)}(\boldsymbol{s}_n) \end{bmatrix}$$

Value function with policy π

We can also define the value function for a policy $\boldsymbol{\pi}$ according to

$$V^{\pi}(\boldsymbol{s}) = E[R(\boldsymbol{s}_0) + \gamma R(\boldsymbol{s}_1) + \gamma^2 R(\boldsymbol{s}_2) + \cdots | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{\pi}]$$

which is simply the expected sum of discounted rewards upon starting in state $\boldsymbol{s}$, and taking actions according to $\boldsymbol{\pi}$.

Given a fixed policy $\boldsymbol{\pi}$, its value function V^{π} satisfies the Bellman equations

$$V^{\pi}(\boldsymbol{s}) = \underbrace{\boxed{R(\boldsymbol{s})}}_{\text{Immediate reward}} + \underbrace{\boxed{\gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{s}\pi(\boldsymbol{s})}(\boldsymbol{s}') V^{\pi}(\boldsymbol{s}')}}_{\text{Expected sum of future discounted rewards}}$$

It can be calculated directly by

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} R^{\pi}$$

How can we find the optimal policy and its corresponding value? (What action to take given each state?)

Learning objectives

Through this lecture, it is aimed for you to be able to

- Model a problem as a Markov decision process
- Implement basic reinforcement learning algorithms, i.e. value iteration and policy iteration

Optimal value function and optimal policy

Our goal is to choose actions over time so as to maximize the expected value of the total payoff. We define the optimal value function according to:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{\pi}} V^{\boldsymbol{\pi}}(\boldsymbol{s})$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of the Bellman's equations for the optimal value function:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a} \in A}[R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{sa}}(\boldsymbol{s}')V^*(\boldsymbol{s}')]$$

How can we define optimal policy?

Optimal value function and optimal policy

Our goal is to choose actions over time so as to maximize the expected value of the total payoff. We define the optimal value function according to:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{\pi}} V^{\boldsymbol{\pi}}(\boldsymbol{s})$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of the Bellman's equation for the optimal value function:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}\in A}[R(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in S} P_{\boldsymbol{sa}}(\boldsymbol{s}')V^*(\boldsymbol{s}')]$$

We also define the optimal policy according to

$$\boldsymbol{\pi}^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}\in \boldsymbol{A}}[R(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in S} P_{\boldsymbol{sa}}(\boldsymbol{s}')V^*(\boldsymbol{s}')]$$

Optimal Policy and optimal value are coupled!

$$\begin{tikzcd} \pi^* \arrow[r, shift left] & V^* \arrow[l, shift left] \end{tikzcd}$$

$$V^*(\boldsymbol{s}) = V^{\boldsymbol{\pi}^*}(\boldsymbol{s}) \geq V^{\pi}(\boldsymbol{s})$$

The two can be computed interchangeably, as each provides enough information to derive the other. This is a fundamental property of MDPs and forms the basis of many reinforcement learning algorithms like Value Iteration and Policy Iteration.

Note that the optimal policy and its corresponding value is independent of initial state of MDP.

Let us develope two learning alorithms to find the optimal value and policy!

Let us develope two learning alorithms to find the optimal value and policy!

Assumption: We know the model of environment (transition probabilities and rewards)

Policy iteration algorithm

Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The policy iteration algorithm proceeds as follows:

1. Initialize π randomly.

Policy iteration algorithm

Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The policy iteration algorithm proceeds as follows:

1. Initialize π randomly.
2. **For** until convergence **do**
3. Let $V := V^{\pi}$

Policy iteration algorithm

Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The policy iteration algorithm proceeds as follows:

1. Initialize $\boldsymbol{\pi}$ randomly.
2. **For** until convergence **do**
3. $\boxed{\text{Let } V := V^{\boldsymbol{\pi}}} \longrightarrow$

For a specific policy $\boldsymbol{\pi}$, reward and transition probabilities only depend on states, therefor you can easily calculate V^{π} by

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} R^{\pi}$$

Policy iteration algorithm

Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The policy iteration algorithm proceeds as follows:

1. Initialize $\boldsymbol{\pi}$ randomly.
2. **For** until convergence **do**
3. Let $V := V^{\boldsymbol{\pi}}$
4. Update policy $\boldsymbol{\pi}$: For each state $\boldsymbol{s}$, let

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\max_{\boldsymbol{a} \in A_s} [R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{s}\boldsymbol{a}}(\boldsymbol{s}') V(\boldsymbol{s}')]$$

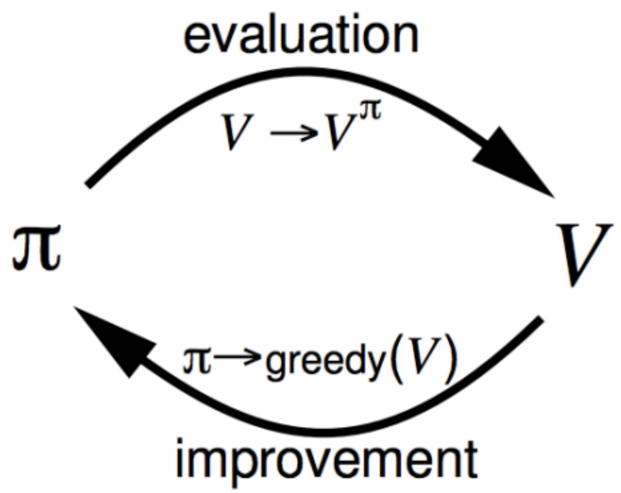
Policy iteration algorithm

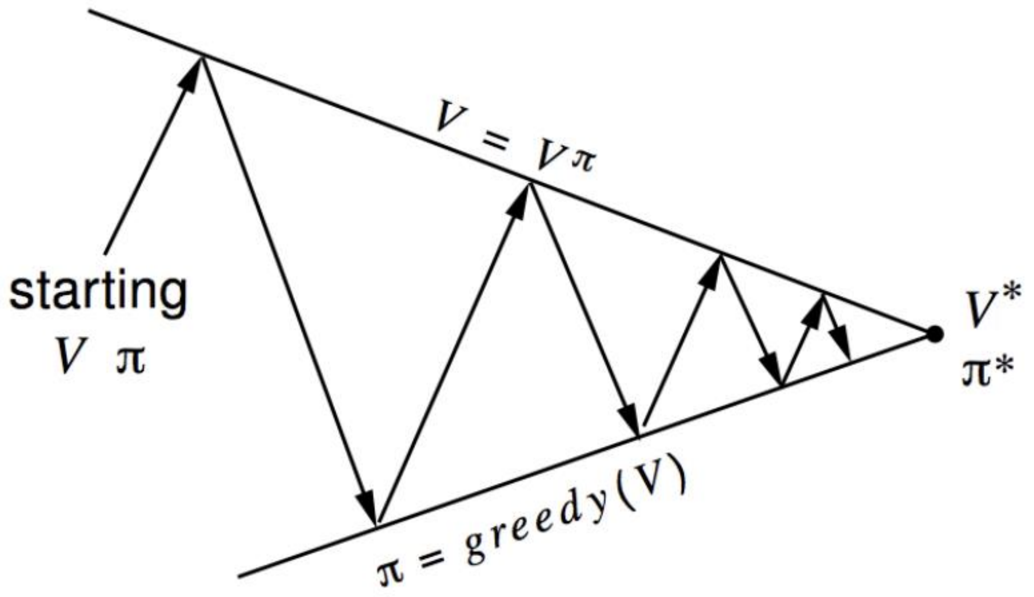
Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The policy iteration algorithm proceeds as follows:

1. Initialize $\boldsymbol{\pi}$ randomly.
2. **For** until convergence **do**
3. Let $V := V^{\boldsymbol{\pi}}$
4. Update policy $\boldsymbol{\pi}$: For each state $\boldsymbol{s}$, let

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\max_{\boldsymbol{a} \in A_s} [R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{sa}}(\boldsymbol{s}')V(\boldsymbol{s}')]$$

Policy iteration





Executing the policy iteration algorithm

	1	2
1		+1
2		-1

Problem description:

Grey cells are goal states.

Reward of a non-goal state = -0.04

Value of state S12= +1

Value of state S22= -1

Transition probabilities: 0.8 in the intended direction, 0.1 to the left of the intended direction and 0.1 to the right of the intended direction (three possibilities).

Find the optimal policy for non-goal states (S11, and S21).

Assume the initial policy is: $\pi(S11)$ = right, $\pi(S21)$ = right

Value iteration algorithm

We now describe two efficient algorithms for solving finite-state MDPs. Let us consider only MDPs with **finite discrete state and action spaces**

The first algorithm, value iteration, is as follows:

1. For each state $\boldsymbol{s}$, initialize $V(\boldsymbol{s}) := 0$.

Value iteration algorithm

We now describe two efficient algorithms for solving finite-state MDPs. Let us consider only MDPs with **finite discrete state and action spaces**

The first algorithm, value iteration, is as follows:

1. For each state $\boldsymbol{s}$, initialize $V(\boldsymbol{s}) \coloneqq 0$.
2. **For** until convergence **do**
3. For every state, update

This is the value function obtained from previous iteration!

$$V(\boldsymbol{s}) = \max_{a \in A}[R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{s}\boldsymbol{a}}(\boldsymbol{s}') \boxed{V(\boldsymbol{s}')}]$$

Value iteration algorithm

We now describe two efficient algorithms for solving finite-state MDPs. Let us consider only MDPs with **finite discrete state and action spaces**

The first algorithm, value iteration, is as follows:

1. For each state s, initialize $V(\boldsymbol{s}) := 0$.
2. **For** until convergence **do**
3. For every state, update

$$V(\boldsymbol{s}) = \max_{\boxed{a \in A_s}} [R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in S} P_{\boldsymbol{sa}}(\boldsymbol{s}') V(\boldsymbol{s}')]$$

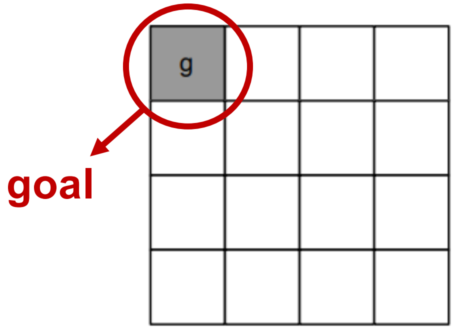
In case some actions are valid only for some specific states, you need to consider feasible actions for each states here.

Value iteration algorithm

Note that when you get the optimal value function, you can calculate the corresponding optimal policy!

$$\boldsymbol{\pi}^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}\in \boldsymbol{A}} [R(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in S} P_{\boldsymbol{s}\boldsymbol{a}}(\boldsymbol{s}')V^*(\boldsymbol{s}')]$$

Example: value iteration



Problem

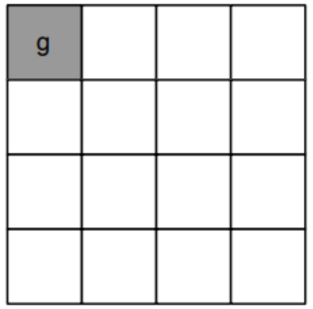
Problem description: This is the shortest path problem. We are trying to figure out what the optimal path is to reach this goal. In other words, how many steps does it take from any position on the grid to the grey state.

Reward for all states except state g is -1 and for state g is 0.

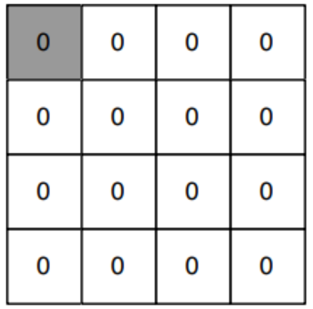
Actions: N,S,E,W, no move.

There is no uncertainty from environment.

Example: value iteration



Problem



V_1

Example: value iteration

g			

Problem

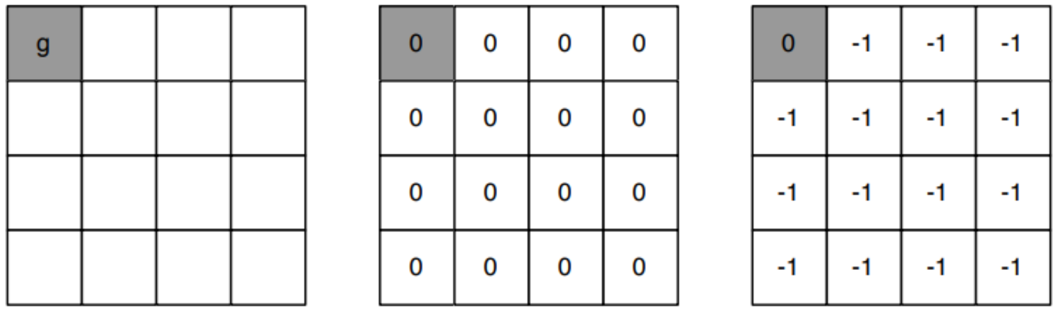
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

Example: value iteration



g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

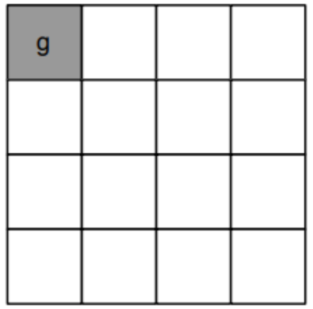
V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

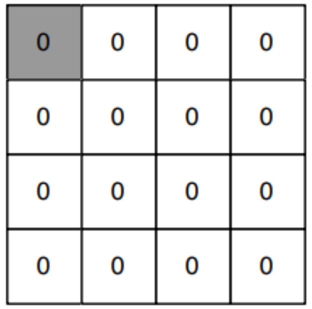
Continue the process until you get the optimal value of each state and the optimal policy!

Example: value iteration



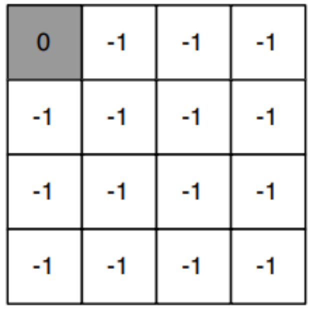
g			

Problem



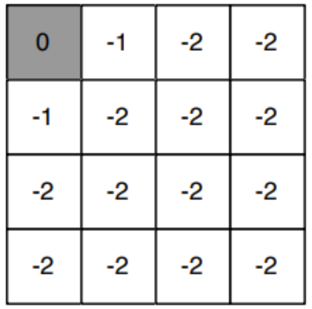
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1



0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2



0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

Example: value iteration

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

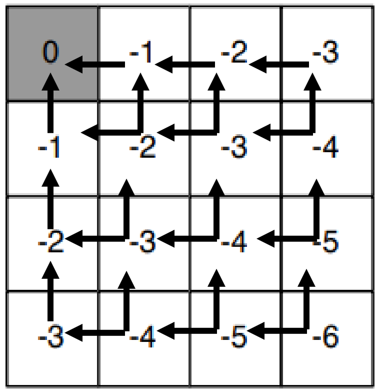
V_7

Example: value iteration

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

What is the optimal policy? What is the best action for each state?

Example: value iteration



What is the optimal policy? What is the best action for each state?

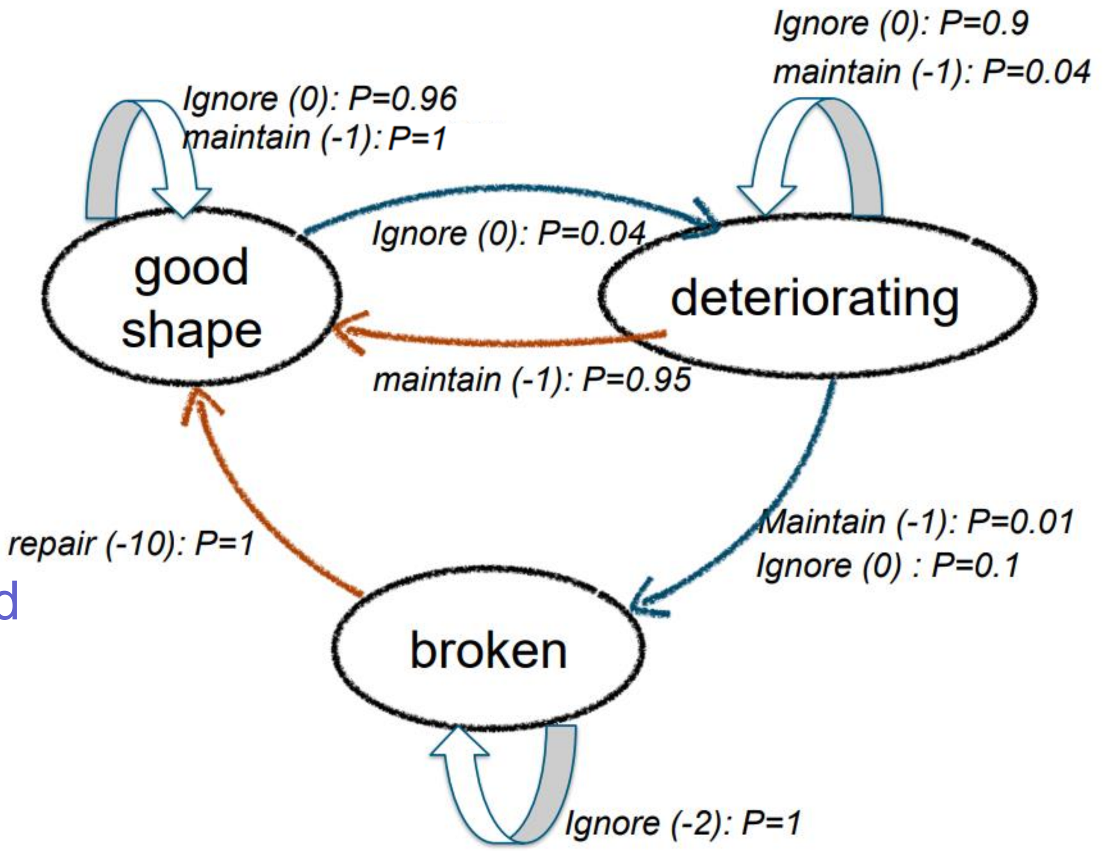
Exercise: Maintenace scheduling

Implement policy and value iterations!

Compare policy and value iterations in terms of computation time and number of iterations.

Increase and decrease the discount factor and explain how the optimal policy changes!

For discount factor 0.9, increase maintenance cost and see how the optimal policy changes.



What if we do not know transition probabilities?

Learning a model for an MDP

- So far, we have discussed MDPs and algorithms for MDPs assuming that the state transition probabilities and rewards are known.
- In many realistic problems, we are not given state transition probabilities and rewards explicitly, but must instead estimate them from data.

$$P_{sa}(s') = \frac{\#\text{times took we action } a \text{ in state } s \text{ and got to } s'}{\#\text{times we took action a in state } s}$$

Exercise: Learning transition probabilities

Suppose the regime sequence is: [L,L,M,M,H,H,M,L]

Compute the transition matrix.

Exercise: Learning Transition Probabilities

Let us revisit the 3-state maintenance problem:

States S = {G (good), D (deteriorating), B (broken)}

Actions A by state:

- G: Ignore, Maintain
- D: Ignore, Maintain
- B: Ignore, Repair

Assume we **do not** know . Instead, we observed N = 100 samples for each (state, action) from historic logs (counts next slide).

Exercise: Learning Transition Probabilities

Observed next-state counts (each row sums to 100):

State (s), Action (a)	to G	to D	to B
G, Ignore	96	4	0
G, Maintain	100	0	0
D, Ignore	0	90	10
D, Maintain	95	4	1
B, Ignore	0	0	100
B, Repair	100	0	0

Using the data above, estimate the transition probabilities $P_{sa}(s')$ for each state–action pair.

Learning objectives

Through this lecture, it is aimed for you to be able to

- Model a problem as a Markov decision process
- Implement basic reinforcement learning algorithms, i.e. value iteration and policy iteration

Thanks for your attention!

Email: farpour@dtu.dk