Reinforcement Learning (Part 2/2)

Alexandre Bergel
DCC - University of Chile
https://bergel.eu
02-12-2025

Goal of today

Provide a *theoretical foundation of RL*

Relevant to *apply more complex techniques*, including Deep RL

Outline

1. Markov chains
2. Markov Decision Process

Outline

1. **Markov chains**
2. Markov Decision Process

Andrey Markov (1856–1922)

Andrey Markov was a Russian mathematician who studied *stochastic processes*

Stochastic refers to having a random probability distribution

Markov was particularly interested in systems that follow a *chain of linked events*

In 1906 Markov produced results about discrete processes that he called *chain*

Markov Chain

Has a set of states $S = \{s_0, s_1, \ldots, s_m\}$

A *Markov chain* is a process that *can move from one state to another*

Process: a series of actions or steps taken in order to achieve a particular end

Each move is a single step and is based on a *transition model* T

The transition model gives the probability to move from one state to another

Markov Chain Example: Weather model

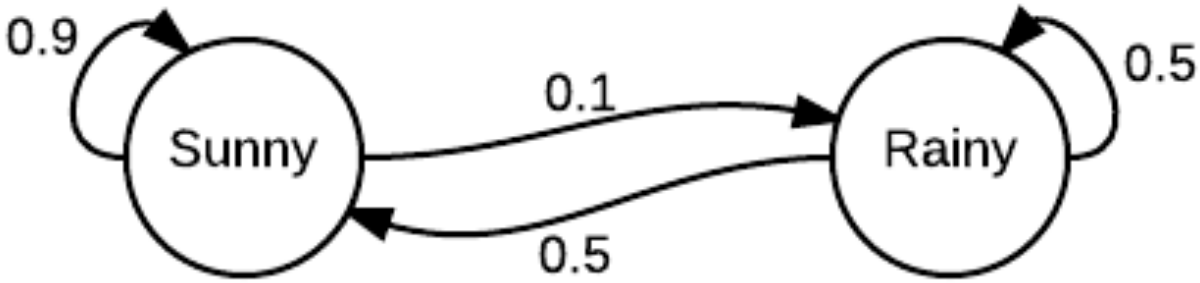
Consider the following simple weather model:

The weather is either *rainy* or *sunny*

A sunny day is 90% likely to be followed by another sunny day

A rain day is 50% likely to be followed by another rainy day

Markov Chain Example: Weather model



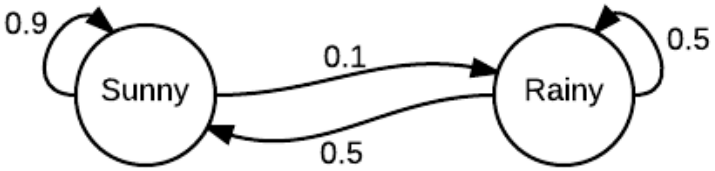
Consider the following simple weather model:

The weather is either *rainy* or *sunny*

A sunny day is 90% likely to be followed by another sunny day

A rain day is 50% likely to be followed by another rainy day

Markov Chain Example: Weather model



$$T = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

sunny rainy

sunny

rainy

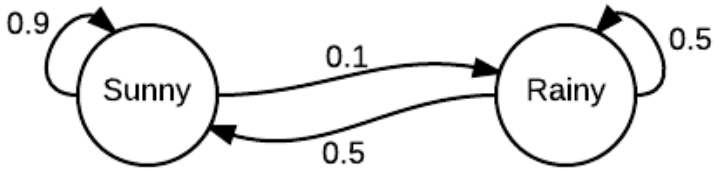
Consider the following simple weather model:

The weather is either *rainy* or *sunny*

A sunny day is 90% likely to be followed by another sunny day

A rain day is 50% likely to be followed by another rainy day

Markov Chain Example: Weather model



$$T = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

sunny rainy

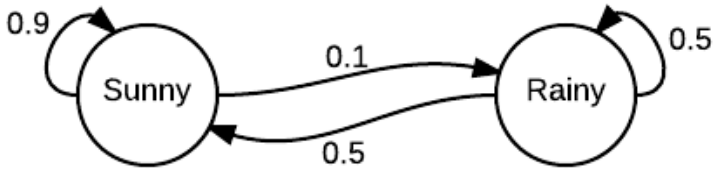
sunny

rainy

The rows of T sum to 1

We call T a *stochastic* matrix

Markov Chain Example: Predicting the weather



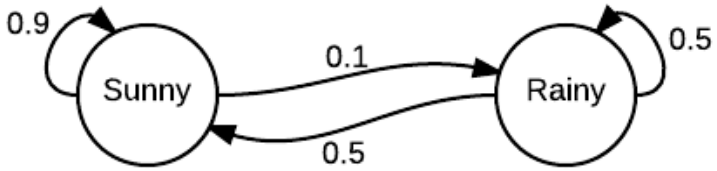
$$T = \begin{array}{c} \begin{array}{cc} \text{sunny} & \text{rainy} \end{array} \\ \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} \end{array} \begin{array}{l} \text{sunny} \\ \text{rainy} \end{array}$$

The weather on day 1 is known to be sunny

This is represented by a vector in which the “sunny” entry is 100% and the “rainy” entry is 0%:

$$x^{(0)} = (1 \quad 0)$$

Markov Chain Example: Predicting the weather

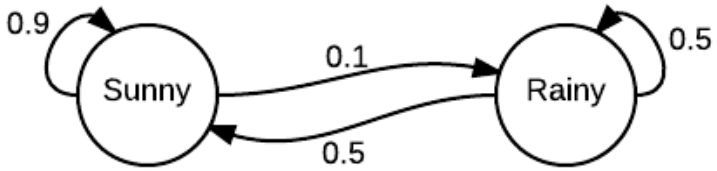


$$T = \begin{array}{c} \begin{array}{cc} \text{sunny} & \text{rainy} \end{array} \\ \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} \end{array} \begin{array}{l} \text{sunny} \\ \text{rainy} \end{array}$$

The weather on day 2 can be predicted by:

$$x^{(1)} = x^{(0)}T = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.9 & 0.1 \end{pmatrix}$$

Markov Chain Example: Predicting the weather

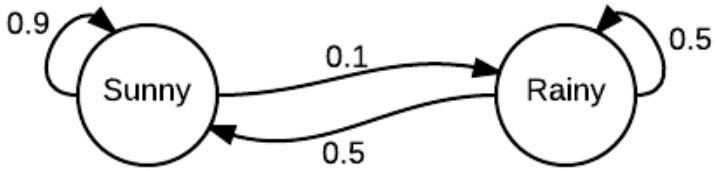


$$T = \begin{array}{c} \begin{array}{cc} \text{sunny} & \text{rainy} \end{array} \\ \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} \end{array} \begin{array}{l} \text{sunny} \\ \text{rainy} \end{array}$$

The weather on day 3 can be predicted by:

$$x^{(2)} = x^{(1)}T = \begin{pmatrix} 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.86 & 0.14 \end{pmatrix}$$

Markov Chain Example: Predicting the weather



$$T = \begin{array}{c} \begin{array}{cc} \text{sunny} & \text{rainy} \end{array} \\ \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} \end{array} \begin{array}{l} \text{sunny} \\ \text{rainy} \end{array}$$

The general rules for day n are:

$$x^{(n)} = x^{(n-1)}T$$

$$x^{(n)} = x^{(0)}T^n$$

Markov Chain Example: Snakes and Ladders

Start at position 1

Goal is to reach the exit (position 100)

Rolling a dice indicates how many spaces to go forward

Ladder: move up

Snake: move down

The next state of the board depends on the current state, and the next roll of the dice

Not a Markov Chain

Most of card games are *not directly* a Markov chain

What you play depends on what you have played

The cards represent a “memory” of the game

The probability for a certain event in the game depends on the history

Markov Chain

To summarize a Markov Chain is defined by

Set of possible states: $S = \{s_0, s_1, \ldots, s_m\}$

Initial state: s_0

Transition model: $T(s, s')$

Markov Property

Markov property refers to the *memoryless property* of a stochastic process

The state in which the process is now is *dependent only from the state* it was at $t - 1$

Selecting an event does not depend on the sequence of events that preceded it

Outline

1. Markov chains
2. **Markov Decision Process**

Markov Decision Process

Markov Decision Process (MDP) is a reinterpretation of Markov chains which *includes an agent* and a *decision making process*. An MDP is defined by:

Set of possible states: $S = \{s_0, s_1, \ldots, s_m\}$

Initial state: s_0

Set of possible actions: $A = \{a_0, a_1, \ldots, a_n\}$

Transition model: $T(s, a, s') = P_a(s, s') = Pr(s_{t+1} = s' \,|\, s_t = s, a_t = a)$ is the probability that action a in state s will lead to state s' at time t'

Reward function: $R_a(s, s')$ is the immediate reward received after moving from s to s' due to action a

Markov chain vs Markov Decision Process

We have introduced new elements with respect to Markov chains

In particular the *transition model depends* on the *current state*, the *next state*, and the *action of the agent*

The transition model returns the *probability* of reaching the *state* s' if the *action* a is done in *state* s

The reward function $R(s)$ returns a real value event time the agent moves from one state to another

Reward function

The reward function $R(s)$ means that *some states are more desirable* than others

The problem we face is: *the agent has to maximize the reward*

The solution is: *find a policy* $\pi(s)$ which returns the action with the highest reward

Policies

A policy is an *agent's strategy*

A policy defines the learning agent's way of behaving at a given time

A policy is written $\pi(s)$ which maps a state to an action: $\pi(s) = a$

Many policies exists, some of them are good, some of them are bad

A policy *evolves during the learning*

Two types of environments

An environment is called *deterministic* is where your *agent's actions uniquely determine the outcome*. For example in Chess, there is no randomness when you move a piece. 

An environment is called *stochastic* is where your *agent's actions do not uniquely determine the outcome*. For example in games with dice, you can determine your dice throwing action but not the outcome of the dice. 

Two types of environments

Deterministic environments are easier to solve because *the agent knows how to plan its actions* more accurately

Total reward

The *goal of the agent is to find the optimal policy*, which will maximize the accumulated rewards received from the environment

Assume we begins at s_0

At time 0: agent observes the environment state s_0 and picks an action a_0. After performing the action, the environment state becomes s_1 and the agent receives reward r_1

Total reward

At time 1: agent observes the environment state s_1 and picks an action a_1. After performing the action, the environment state becomes s_2 and the agent receives reward r_2

At time 2: agent observes the environment state s_2 and picks an action a_2. After performing the action, the environment state becomes s_3 and the agent receives reward r_3

Total reward

The accumulated reward is =

$$r_1 + r_2 + r_3 + r_4 + \ldots$$

It is common to use a discount factor to give *higher weight to near reward* than reward received further in the future

So, we have accumulated discounted reward is

$$= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots$$

Total reward

The general formula to define the accumulated discounted reward $= \sum_{i=1}^{T} \gamma^{i-1} r_i$

Where T is called the horizon, the episode length (number of steps per episode)

We have $0 \leq \gamma \leq 1$

Discounted reward

Why having a discount?

What do you prefer? 1000 USD right now, or 1000 USD in 10 years?

Imagine a robot looking for an exit

- Long path with a low reward
- Short path with a high reward

The γ discount value controls this kind of behavior

Optimal policy

The optimal policy is written $\pi^*(s)$

$\pi^*(s)$ gives the *highest accumulated reward* for a sequence of actions, starting in s

How to find $\pi^*(s)$?

Optimal policy

Remember that the reward at a given time depends on the policy $\pi(s) = a$. We therefore are looking the policy that maximizes the accumulated discounted reward

$$= \sum_{i=1}^{T} \gamma^{i-1} r_i$$

Optimal policy

Value iteration and *policy iteration* are two algorithms to find the optimal policy π^*

However, these algorithms *implies* that we know *the transition model* $T(s, a, s')$

The full MDP has to be known, which is not really practical in many situations (e.g., when offline planning is not an option)

Q-Learning

Q-Learning is an algorithm to find π^* *without knowing the transition model*

The agent discovers what is a good and bad action by trial and error

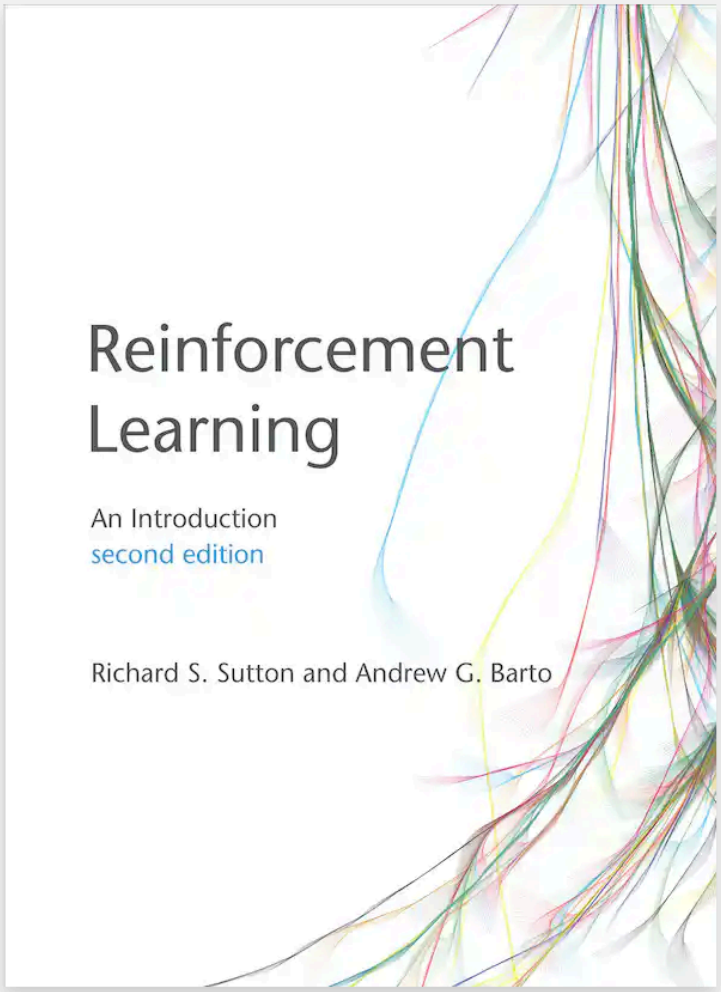
Concluding words

RL is associated to a Markov Decision Process

MDP requires states, actions, transitions, rewards

Q-Learning is an efficient algorithm to solve the MDP without knowing the transition and reward models

Different learning algorithms exist (e.g., Q-learning, SARSA, Monte-Carlo)



From: Adaptive Computation and Machine Learning series

Reinforcement Learning, Second Edition

An Introduction

By Richard S. Sutton and Andrew G. Barto

552 pp., 7 x 9 in, 64 color illus., 51 b&w illus.
Hardcover
ISBN: 9780262039246
Published: November 13, 2018

Penguin Random House | Amazon | Barnes and Noble | Indigo | Books a Million

This mathematical book is a reference in the field

dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE