# Lecture Notes:
# CSCI 688 – Applications of Markov Chains

Daniela Hurtado-Lange
William & Mary

Spring 2023

# Contents

# Chapter 0

# Welcome and expectations of this course

Welcome to Applications of Markov Chains! I hope you learn what you are expecting this semester, and that we have fun together discussing about these beautiful stochastic models.

In this course, we aim to learn some of the most popular applications of Markov chains. This note is divided in three big chapters. We start reviewing the foundational probability knowledge we'll need this semester, along with the essential knowledge we need about Markov chains. Then, we will spend some time learning the foundations of Reinforcement Learning, and some of the key concepts in Queueing (or queuing) Theory.

This note is constructed based on the following three textbooks:


[1] Harchol-Balter, M. (2013). *Performance Modeling and Design of Computer Systems: Queueing Theory in Action.* Cambridge University Press.

[2] Ross, S.M. (2014). *Introduction to Probability Models.* Academic Press.

[3] Sutton, R.S. & Barto, A.G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

# Chapter 1

# Review of Probability and Discrete-Time Markov Chains

[This chapter is based on Ch 1-4 from Ross' book, and Ch 3, and 8 from Harchol-Balter's book]

In this chapter we review the foundations we need for the rest of the semester. We will start from basic probability, spend some time in conditional probability and then quickly review Markov chains. We start with a brief review of probability.

## 1 Basic concepts

When we think about probability or stochastic processes, we usually try to predict the outcome of some experiment. For example, rolling a die or flipping a coin. Even when we don't know the specific outcome we will get, we usually have an idea of what outcomes are possible, and a subset of outcomes that we are interested in.

**Definition 1.1.** *Given a probabilistic experiment,*

*(i) The sample space is the set of all possible outcomes. We usually denote the sample space by $\mathcal{S}$ or $\Omega$.*

*(ii) An event is a subset of the sample space, and we typically denote it with capital letters such as $E$, $F$, or $G$.*

*(iii) The probability of an event $E$ is denoted by $\mathbb{P}[E]$ and represents the proportion of times we observe the outcomes in $E$ if we repeat the experiment infinitely many times. We can compute $\mathbb{P}[E]$ as follows:*

$$\mathbb{P}[E] = \frac{|E|}{|\Omega|},$$

*that is, the number of outcomems in the event $E$ divided by the size of the state space $\Omega$.*

Let's see some examples.

**Example 1.1.** *In the following situations, determine the sample space of the experiment, the event described, and compute the probability of the event.*

*(a) Flip a coin once and observe if you get tails.*

*(b) Roll a die and observe if you obtain an odd number.*

*(c) Roll two dice and observe if the sum of the outcomes is 4.*

**Solution.**

(a) The state space is heads or tails. Mathematically, we may write

$$\Omega = \{H, T\}$$

We are interested in observing if we obtain tails, so the event is

$$E = \{\text{Tails}\} = \{T\}$$

which only containts one possible outcome. Then, the probability of $E$ is

$$\mathbb{P}[E] = \frac{1}{2}$$

(b) Now the possible outcomes are integer numbers between 1 and 6. Then, the state space is

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

We are interested in odd numbers. Then, the event is

$$E = \{1, 3, 5\}$$

which contains 3 possible outcomes. Then, the probability of $E$ is

$$\mathbb{P}[E] = \frac{3}{6} = \frac{1}{2}$$

(c) In this case we have 2 dice, and each of them will show an integer number between 1 and 6. Then, the set of possible outcomes is the set of **pairs** of integer numbers between 1 and 6. Specifically, we have

$$\Omega = \left\{ \begin{array}{cccccc} (1,1), & (1,2), & (1,3), & (1,4), & (1,5), & (1,6), \\ (2,1), & (2,2), & (2,3), & (2,4), & (2,5), & (2,6), \\ (3,1), & (3,2), & (3,3), & (3,4), & (3,5), & (3,6), \\ (4,1), & (4,2), & (4,3), & (4,4), & (4,5), & (4,6), \\ (5,1), & (5,2), & (5,3), & (5,4), & (5,5), & (5,6), \\ (6,1), & (6,2), & (6,3), & (6,4), & (6,5), & (6,6) \end{array} \right\}$$

The event only includes the pairs that add up to 4. Then, we have

$$E = \{(1,3),\ (2,2),\ (3,1)\}$$

and the probability of the event $E$ is

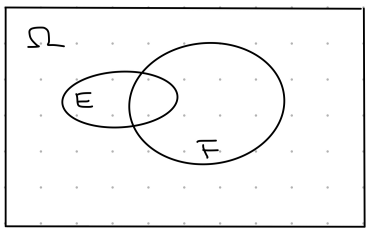
$$\mathbb{P}[E] = \frac{3}{36} = \frac{1}{12}$$

□

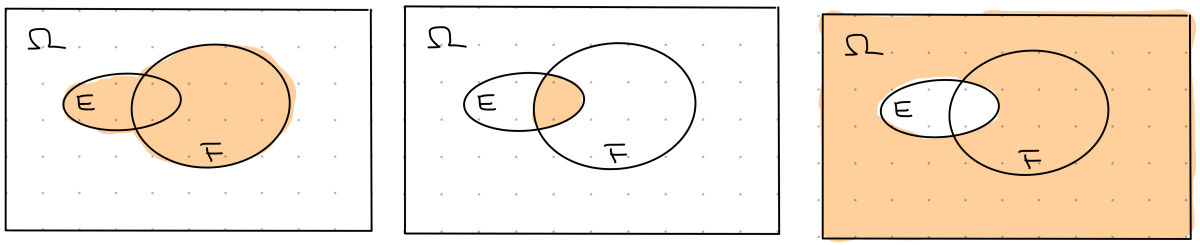
It is important to notice that the events are sets and, hence, we can take unions, intersections and complements. Recall:

- Union of events $E$ and $F$ includes all the elements in $E$ **or** $F$ (or both) and is denoted by $E \cup F$
- Intersection of events $E$ and $F$ includes the elements in **both** sets and is denoted by $E \cap F$
- Complement of the event $E$ is everything in the state space that is **not part of** $E$, and is denoted by $E^c$

**Example 1.2.** *For the events of the figure below, color the union and intersection of the events $E$ and $F$, and the complement of the event $E$.*



**Solution.** Based on the definitions above, we obtain the following representation of the union, intersection and complement.




(a) Union: $E \cup F$ (b) Intersection: $E \cap F$ (c) Complement: $E^c$

□

The union, intersection and complement of events are also events and, as such, we can compute their probability.

**Theorem 1.1.** *The probability of the union of two events $E$ and $F$ is*

$$\mathbb{P}[E \cup F] = \mathbb{P}[E] + \mathbb{P}[F] - \mathbb{P}[E \cap F]$$

Intuitively, we can think of probabilities as areas (or number of outcomes). On the left-hand side, we have the total area of the union of events, and on the right-hand side we show one way to compute it using the area of each event alone. Observe that if we only do $\mathbb{P}[E] + \mathbb{P}[F]$ we would be counting twice the area where the events $E$ and $F$ intersect. Then, we need to subtract $\mathbb{P}[E \cap F]$.

**Example 1.3.** *Suppose we roll two dice. What is the probability that the sum or the product of the outcomes equals 4 (or both)?*

**Solution.** We already know the state space from Example 1.1(c). There are two events to consider:

$$E = \{\text{Sum of the outcomes equals } 4\} = \{(1,3), (2,2), (3,1)\}$$
$$F = \{\text{Product of the outcomes equals } 4\} = \{(1,4), (2,2), (4,1)\}$$

Then, $E \cap F = \{(2,2)\}$.

The probability that the sum **or** multiplication of the outcomes is 4 can be computed as:

$$\begin{aligned}\mathbb{P}[E \cup F] &= \mathbb{P}[E] + \mathbb{P}[F] - \mathbb{P}[E \cap F] \\ &= \frac{3}{36} + \frac{3}{36} - \frac{1}{36} = \frac{5}{36}\end{aligned}$$

□

To compute the probability of the intersection using the probability of each event we need to introduce the concept of conditional probability.

**Definition 1.2.** *The conditional probability of event $E$ given event $F$ is written $\mathbb{P}[E|F]$ and is given by*

$$\mathbb{P}[E|F] = \frac{\mathbb{P}[E \cap F]}{\mathbb{P}[F]}$$

Intuitively, $\mathbb{P}[E|F]$ is the probability of observing an outcome of event $E$ if we already know that event $F$ happened. Then, if we go to our intuitive definition of probability, $\mathbb{P}[F]$ represent the state space where event $F$ happens, and $\mathbb{P}[E \cap F]$ represents the "successful" outcomes.

**Example 1.4.** *In our example of rolling 2 dice, compute the probability that the sum of the outcomes is 4 given that the product of the outcomes is 4.*

**Solution.** We use the definition of $E$ and $F$ from our solution to Example 1.3 and obtain

$$\mathbb{P}[E|F] = \frac{\mathbb{P}[E \cap F]}{\mathbb{P}[F]} = \frac{1/36}{3/36} = \frac{1}{3}$$

Observe that the probability of event $E$ increased considerably when we added the information "the product of the outcomes is 4" because, out of the 36 outcomes, we are saying that only 3 are possible. □

In the example above, we saw that knowing that event $F$ occured changed the probability of event $E$ happening. In other words, the extra information is useful to predict the outcome of the dice. However, this is not always the case.

**Definition 1.3.** *The events $E$ and $F$ are independent if*

$$\mathbb{P}[E \cap F] = \mathbb{P}[E] * \mathbb{P}[F]$$

*or, equivalently,*

$$\mathbb{P}[E|F] = \mathbb{P}[E]$$

For example, the events $E$ and $F$ above are not independent. On the other hand, if $E$ represents obtaining tails when tossing a coin and $F$ represents obtaining an odd number when rolling a die, then $E$ and $F$ are independent. You may verify with the definition as an exercise.

The importance of conditional probability goes beyond detecting if two events are independent or not. Indeed, when we have events that are not independent, we can use conditional probability to our favor. Formally, we have the following key results.

**Theorem 1.2** (Law of total probability)**.** *Let the events $F_1, \dots, F_n$ be a partition of the state space $\Omega$, that is, $\cup_{i=1}^n F_i = \Omega$ and $F_i \cap F_j = \emptyset$ for all $i \neq j$. Then,*

$$\mathbb{P}[E] = \sum_{i=1}^{n} \mathbb{P}[E|F_i]\,\mathbb{P}[F_i]$$

*or, equivalently,*

$$\mathbb{P}[E] = \sum_{i=1}^{n} \mathbb{P}[E \cap F_i]$$

In words, the law of total probability says that if we want to compute the probability of an event $E$, we can separate in cases (the events $F_i$). However, we must make sure that the cases cover all the possibilities ($\cup_{i=1}^n F_i = \Omega$) and that we don't count the same case more than once ($F_i \cap F_j = \emptyset$). Let's see an example.

**Example 1.5.** *Suppose an urn contains 7 green balls and 5 purple balls. We draw two balls from the urn without replacement. Assuming that each ball in the urn is equally likely to be drawn,*

*(a) What is the probability that both drawn balls are green?*

*(b) How does your answer change if the two balls are drawn with replacement?*

**Solution.**

(a) We first define our events:

$$E = \text{"Both balls are green"}$$
$$F_1 = \text{"First ball is green"}$$
$$F_2 = \text{"First ball is purple"}$$

Since we draw balls without replacement, the probability that the second ball is green depends on whether the first ball we draw is green or purple. Then, we can split in two cases (color of the first ball) and use the law of total probability as follows.

Observe that $F_1, F_2$ is a partition because it covers all the cases, and there is no way that a ball has two colors so $F_1 \cap F_2 = \emptyset$. Then, we have

$$\begin{aligned}\mathbb{P}[E] &= \mathbb{P}[E|F_1]\,\mathbb{P}[F_1] + \mathbb{P}[E|F_2]\,\mathbb{P}[F_2] \\ &= \frac{6}{11} * \frac{7}{12} + \frac{0}{11} * \frac{5}{12} = \frac{42}{132}\end{aligned}$$

(b) If we replace the first ball we draw, then the color of the first ball does not influence the probability that the second ball is green. Specifically, we obtain

$$\begin{aligned}\mathbb{P}[E] &= \mathbb{P}[E|F_1]\,\mathbb{P}[F_1] + \mathbb{P}[E|F_2]\,\mathbb{P}[F_2] \\ &= \frac{7}{12} * \frac{7}{12} + \frac{0}{11} * \frac{5}{12} = \frac{49}{144}\end{aligned}$$

□

In other situations, we know $\mathbb{P}[E|F]$, but we need to compute $\mathbb{P}[F|E]$. Let's see how we do it and a concrete example.

**Theorem 1.3** (Bayes Rule)**.** *Consider two events $E$ and $F$. Then,*

$$\mathbb{P}[F|E] = \frac{\mathbb{P}[E|F] * \mathbb{P}[F]}{\mathbb{P}[E]}$$

Observe that we may use the law of total probability to compute $\mathbb{P}[E]$. The classic example on Bayes' rule is computing the probability that a person is ill given that they tested positive. Let's do a different example.

**Example 1.6.** *Consider two urns. The first urn contains 2 green balls and 7 purple balls. The second urn contains 5 green and 6 purple balls. We flip a fair coin and then draw a ball from the first or the second urn, depending on whether the outcome was heads or tails, respectively. What is the probability that the outcome of the toss was heads given that a green ball was selected?*

**Solution.** We first define our events. Let

$$\begin{aligned}H &= \text{“Coin comes up heads”} \\ G &= \text{“Draw a green ball”}\end{aligned}$$

Then, we know:

$$\mathbb{P}[G|H] = \frac{2}{9}, \quad \mathbb{P}[G|H^c] = \frac{5}{11}$$

and we need to compute $\mathbb{P}[H|G]$. We use Bayes' rule:

$$\begin{aligned}\mathbb{P}[H|G] &= \frac{\mathbb{P}[G|H] * \mathbb{P}[H]}{\mathbb{P}[G]} \qquad \text{(Bayes' rule)} \\ &= \frac{\mathbb{P}[G|H] * \mathbb{P}[H]}{\mathbb{P}[G|H] * \mathbb{P}[H] + \mathbb{P}[G|H^c] * \mathbb{P}[H^c]} \qquad \text{(law of total probability in denominator)} \\ &= \frac{2/9 * 1/2}{2/9 * 1/2 + 5/11 * 1/2} \\ &= \frac{2/9}{2/9 + 5/11} = \frac{22}{67}\end{aligned}$$

□

# 2 Random variables

So far we've been discussin experiments that may or may not have numerical values. For example, when we roll a die, the outcomes are integer numbers from 1 to 6. However, if we toss a coin, the outcomes are heads or tails. We can certainly associate a number to each outcome, but it's not intrinsic to the experiment. In this section we focus on experiments with numerical outcomes.

**Definition 2.1.** *A random variable (rv) is a real-valued function of the outcome of the experiment.*

*(i) A discrete rv can take on at most countably many possible values*

*(ii) A continuous rv can take on an uncountable set of possible values*

*We frequently use capital letters to represent random variables, and lower case letters to represent their values.*

For example, the outcome of a die is a random variable, which we may represent by $X$. Then, the possible values are $x = 1, 2, 3, 4, 5, 6$. In the case of tossing a coin, we need to define a function to associate a number with each outcome. We may define a random variable $Y$ such that $Y(heads) = 1$ and $Y(tails) = 0$, which has $y = 0, 1$ as its possible values.

The distinction between discrete and continuous random variables depends on whether the possible values are a list or a range. The random variables $X$ and $Y$ defined above are discrete. A simple example of a continuous random variable is the time until something happens. For example, the time until the next question in class can be modeled as a random variable $T$ that can take any value in the interval $[0, 50]$ minutes.

## 2.1 Probabilities and densities

When we compute probabilities of an event defined by a random variable, we must make the distinction between discrete and continuous.

**Definition 2.2.** *Let $X$ be a discrete random variable. The probability mass function (pmf) of $X$ is defined by*

$$p_X(x) = \mathbb{P}[X = x], \quad \text{where } \sum_x p_X(x) = 1$$

*The cumulative distribution function (cdf) of $X$ is defined by*

$$F_X(x) = \mathbb{P}[X \le x] = \sum_{i \le x} p_X(i)$$

**Example 2.1.** *Common random variables are the Bernoulli, Binomial and Geometric. In all cases, the experiment has two possible outcomes: 1 (success) or 0 (failure), where the probability of success is $p$.*

*(a) A Bernoulli random variable represents the outcome of one experiment. Then, if $X$ is Bernoulli, we have*

$$p_X(0) = 1 - p, \qquad p_X(1) = p$$

*(b) A Binomial random variable represents the number of successful outcomes out of $n$ experiments. Then, if $X$ is Binomial, we have*

$$p_X(x) = \binom{n}{x} p^x (1-p)^{n-x}, \qquad x = 0, 1, \ldots, n$$

*(c) A Geometric random variable represents the number of experiments until the first success. Then, if $X$ is Geometric, we have*

$$p_X(x) = p(1-p)^{x-1}, \qquad x = 1, 2, 3, \ldots$$

For continuous random variables, writing $\mathbb{P}[X = x]$ doesn't make sense because the probability of each individual value is 0. Instead, we compute the probability that $X$ is in a certain interval and we use the density function.

**Definition 2.3.** *The probability density function (pdf) of a continuous random variable $X$ is a nonnegative function $f_X(x)$ such that*

$$\mathbb{P}[a \le X \le b] = \int_a^b f_X(x)\,dx, \quad \text{where } \int_{-\infty}^{\infty} f_X(x)\,dx = 1$$

*The cumulative distribution function (cdf) of a continuous random variable $X$ is defined by*

$$F_X(x) = \mathbb{P}[X \le x] = \int_{-\infty}^{x} f_X(y)\,dy$$

As mentioned above, $\mathbb{P}[X = x]$ is always zero for continuous random variables. However, the density function is closely related. We can think of $f_X(x)\,dx = \mathbb{P}[x \le X \le x + dx]$, that is, $f_X(x)$ represents the probability that $X$ takes a value close to $x$.

Using the fundamental theorem of calculus, notice that

$$f_X(x) = \frac{d}{dx} F_X(x)$$

**Example 2.2.** *Examples of continuous distributions are:*

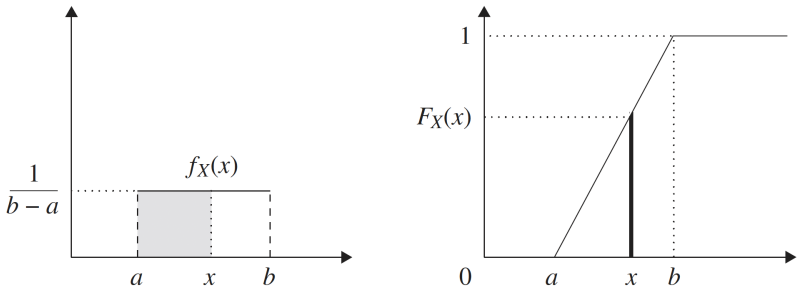
*(a) Uniform distribution in the interval $[a, b]$: All the subintervals of the same length have the same probability. Then, the density function is constant between $a$ and $b$. Specifically,*

$$f_X(x) = \begin{cases} \dfrac{1}{b-a} & , \text{ if } a \le x \le b \\ 0 & , \text{ otherwise} \end{cases}$$

*Then, the cdf is*

$$F_X(x) = \begin{cases} 0 & , \text{ if } x < a \\ \dfrac{x-a}{b-a} & , \text{ if } a \le x \le b \\ 1 & , \text{ if } x > b \end{cases}$$

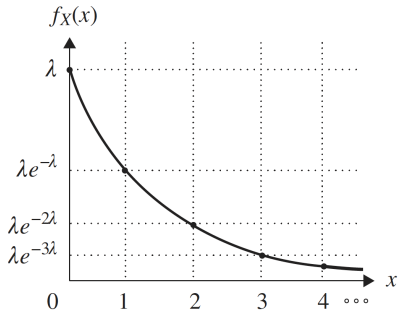
*Pictorially, we have*




*(b) Exponential distribution with rate $\lambda$: The density function is*

$$f_X(x) = \begin{cases} 0 & , \text{ if } x \le 0 \\ \lambda e^{-\lambda x} & , \text{ if } x > 0 \end{cases}$$

*Pictorially,*




*The cumulative function is*

$$F_X(x) = \begin{cases} 0 & , \text{ if } x \le 0 \\ 1 - e^{-\lambda x} & , \text{ if } x > 0 \end{cases}$$

*The exponential distribution is super important in stochastic processes, because it is the only continuous distribution that satisfies the memoryless property, that is, if $X$ has exponential distribution we have*

$$\mathbb{P}[X > t + s | X \geq s] = \mathbb{P}[X > t]$$

## 2.2 Expectation and Variance

The pmf, pdf and cdf contain are functions that completely determine the distribution of a rv and, therefore, contain all the information we need about a rv. However, sometimes we need a summary of information in order to compare performance more intuitively. In such cases, we use measures that show us where is the center and how disperse around the center the distribution is.

**Definition 2.4.**

*(i) The mean or expectation of a random variable $X$ is denoted by $\mathbb{E}[X]$ and is a measure of central tendency of $X$. If $X$ is a discrete random variable, we define*

$$\mathbb{E}[X] = \sum_x x\, p_X(x)$$

*and if $X$ is continuous, we define*

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f_X(x)\, dx$$

*(ii) The variance of a random variable $X$ is denoted by $Var[X]$ and is a measure of dispersion of $X$. It is defined as*

$$Var[X] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

Let's see some examples.

**Example 2.3.** *For each of the following distributions, compute the mean and variance.*

*(a) Bernoulli($p$)*

*(b) Uniform($a, b$)*

**Solution.** We use the definitions above.

(a) Bernoulli is a discrete distribution, where

$$p_X(0) = 1 - p \qquad \text{and} \qquad p_X(1) = p$$

Then, the expected value is

$$\mathbb{E}[X] = 0 * p_X(0) + 1 * p_X(1) = 0 * (1 - p) + 1 * p = p$$

To compute the variance we first compute $\mathbb{E}\left[X^2\right]$. We obtain

$$\mathbb{E}\left[X^2\right] = 0^2 * p_X(0) + 1^2 * p_X(1) = p$$

Then, the variance is

$$\text{Var}[X] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 = p - p^2 = p(1 - p)$$

(b) Uniform is a continuous distribution. Then, we compute the integral. We obtain

$$\begin{aligned}\mathbb{E}[X] &= \int_{-\infty}^{\infty} x f_X(x)\, dx \\ &= \int_a^b x \left(\frac{1}{b - a}\right) dx\end{aligned}$$

$$= \left(\frac{1}{b-a}\right)\left(\frac{b^2}{2} - \frac{a^2}{2}\right)$$
$$= \left(\frac{1}{b-a}\right)\left(\frac{(b+a)(b-a)}{2}\right)$$
$$= \frac{a+b}{2}$$

Similarly to part (a), to compute the variance we start computing $\mathbb{E}\left[X^2\right]$. We obtain

$$\begin{aligned}
\mathbb{E}\left[X^2\right] &= \int_{-\infty}^{\infty} x^2 f_X(x)\,dx \\
&= \int_a^b x^2 \left(\frac{1}{b-a}\right) dx \\
&= \left(\frac{1}{b-a}\right)\left(\frac{b^3}{3} - \frac{a^3}{3}\right) \\
&= \left(\frac{1}{b-a}\right)\left(\frac{(b-a)(b^2+ab+a^2)}{3}\right) \\
&= \frac{b^2+ab+a^2}{3}
\end{aligned}$$

Then, the variance is

$$\begin{aligned}
\mathrm{Var}\left[X\right] &= \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2 \\
&= \frac{b^2+ab+a^2}{3} - \left(\frac{a+b}{2}\right)^2 \\
&= \frac{b^2+ab+a^2}{3} - \frac{a^2+2ab+b^2}{4} \\
&= \frac{4b^2+4ab+4a^2-(3a^2+6ab+3b^2)}{12} \\
&= \frac{b^2-2ab+a^2}{12} \\
&= \frac{(b-a)^2}{12}
\end{aligned}$$

□

Let's summarize some of the main properties of the expectation and variance.

**Theorem 2.1.** *If $X$ and $Y$ are random variables, and $a, b$ are real numbers, we have*

*(i) Linearity of expectation:*

$$\mathbb{E}\left[aX+bY\right] = a\mathbb{E}\left[X\right] + b\mathbb{E}\left[Y\right]$$

*(ii) Expectation of a function of a random variable:*
*If $X$ is a discrete random variable,*

$$\mathbb{E}\left[g(X)\right] = \sum_x g(x)p_X(x)$$

*and if $X$ is a continuous random variable,*

$$\mathbb{E}\left[g(X)\right] = \int_{-\infty}^{\infty} g(x)f_X(x)\,dx$$

*(iii) Variance of a linear combination of random variables:*

$$Var\left[aX+bY\right] = a^2\,Var\left[X\right] + b^2\,Var\left[Y\right] + 2\,Cov\left[X,Y\right],$$

*where $Cov\left[X,Y\right]$ is the covariance between $X$ and $Y$, and is defined by*

$$Cov\left[X,Y\right] = \mathbb{E}\left[XY\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right]$$

*(iv) If $X$ and $Y$ are independent random variables, then*

$$\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$$

*and*

$$Cov[X, Y] = 0$$

*The opposite is not true, that is, if $Cov[X, Y] = 0$, $X$ and $Y$ may not be independent.*

Let's see some examples.

**Example 2.4.** *Compute the mean and variance of a binomial random variable with parameters $n$ and $p$.*

**Solution.** Recall that a binomial random variable is the sum of $n$ independent Bernoulli random variables. Then, we define $X$ as our binomial random variable and $X_i$ as the $i^{\text{th}}$ Bernoulli random variable. We obtain

$$\begin{aligned}
\mathbb{E}[X] &= \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathbb{E}[X_i] \qquad \text{(linearity of expectation)} \\
&= \sum_{i=1}^{n} p \\
&= np
\end{aligned}$$

From property (iii) above, observe that if the random variables are independent, the variance of the sum becomes the sum of the variances. Then, we obtain

$$\begin{aligned}
\text{Var}[X] &= \text{Var}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \text{Var}[X_i] \qquad \text{(independence)} \\
&= \sum_{i=1}^{n} p(1-p) \\
&= np(1-p)
\end{aligned}$$

□

## 2.3 Probabilities and Expectations Via Conditioning

The definition of conditional probability for events is naturally extended to random variables. For discrete random variables, we simply replace the event $E$ by $X = x$, and $F$ by $Y = y$. The same idea can be used for continuous random variables, but instead of probability mass function we use the probability density function. Let's see an example.

**Example 2.5.** *If $X$ and $Y$ are independent Poisson random variables with means $\lambda_1$ and $\lambda_2$, compute the conditional expectation of $X$ given that $X + Y = n$.*

**Solution.** We need to compute $\mathbb{E}[X|X + Y = n]$. Maybe the easiest way to approach this problem is computing the probability mass function before. By definition of conditional probability, we have

$$\begin{aligned}
\mathbb{P}[X = k \,|\, X + Y = n] &= \frac{\mathbb{P}[X = k \cap X + Y = n]}{\mathbb{P}[X + Y = n]} \\
&= \frac{\mathbb{P}[X = k \cap Y = n - k]}{\mathbb{P}[X + Y = n]}
\end{aligned}$$

$$= \frac{\mathbb{P}[X=k] * \mathbb{P}[Y=n-k]}{\mathbb{P}[X+Y=n]} \qquad \text{(independence)}$$
$$= \frac{\left(\frac{\lambda_1^k e^{-\lambda_1}}{k!}\right)\left(\frac{\lambda_2^{n-k} e^{-\lambda_2}}{(n-k)!}\right)}{\frac{(\lambda_1+\lambda_2)^n e^{-(\lambda_1+\lambda_2)}}{n!}} \qquad \text{(since sum of Poisson rv is also Poisson)}$$
$$= \binom{n}{k}\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\right)^k \left(\frac{\lambda_2}{\lambda_1+\lambda_2}\right)^{n-k}$$

which corresponds to a binomial distribution with $n$ experiments and probability of success $p = \frac{\lambda_1}{\lambda_1+\lambda_2}$. Therefore,

$$\mathbb{E}[X|X+Y=n] = np = \frac{n\lambda_1}{\lambda_1+\lambda_2}$$

□

**Example 2.6.** *A supercomputer center runs large parallel jobs for scientists from all over the country. To charge users appropriately, jobs are grouped into different bins based on the number of CPU hours they require, each with a different price. Suppose that job durations are Exponentially distributed with mean 1000 processor-hours. Further, suppose that all jobs requiring less than 500 processor-hours are sent to bin 1, and all remaining jobs are sent to bin 2.*

*(a) What is the probability that a random job is sent to bin 1?*

*(b) What is the probability that the duration of a job sent to bin 1 is below 200?*

*(c) What is the conditional density of the duration $X$, $f_{X|Y}(x)$, where $Y$ is the event that the job is sent to bin 1?*

*(d) What is the conditional expectation of the job duration, given that it is sent to bin 1?*

**Solution.**

(a) Based on the notation from the problem, we need:

$$\mathbb{P}[Y] = \mathbb{P}[X<500] = F_X(500) = 1 - e^{-\frac{1}{1000}*500} = 1 - e^{\frac{1}{2}}$$

(b) In this case, we need:

$$\mathbb{P}[X<200|X<500] = \frac{\mathbb{P}[X<200 \cap X<500]}{\mathbb{P}[X<500]} = \frac{\mathbb{P}[X<200]}{\mathbb{P}[X<500]} = \frac{1-e^{-\frac{1}{5}}}{1-e^{-\frac{1}{2}}}$$

(c) We use the definition of conditional density (similar to conditional probability) and obtain:

$$f_{X|Y}(x) = \begin{cases} \frac{f_X(x)}{\mathbb{P}[Y]} & \text{if } x<500 \\ 0 & \text{otherwise} \end{cases}$$
$$= \begin{cases} \frac{\frac{1}{1000}e^{-\frac{x}{1000}}}{1-e^{-\frac{1}{2}}} & \text{if } x<500 \\ 0 & \text{otherwise} \end{cases}$$

(d) We use the density computed in the last part. We obtain

$$\mathbb{E}[X|Y] = \int_0^{500} x\frac{\frac{1}{1000}e^{-\frac{x}{1000}}}{1-e^{-\frac{1}{2}}}\,dx \approx 229$$

□

We can also use conditioning to compute expectations directly, that is, without computing the probability mass function or probability density function of the conditional random variable.

**Definition 2.5.** *Let $X$ and $Y$ be random variables. Then, the conditional expectation $\mathbb{E}[X|Y]$ is a function of the random variable $Y$ that takes the value $\mathbb{E}[X|Y=y]$ when $Y=y$. Hence, $\mathbb{E}[X|Y]$ is also a random variable.*

An extremely important property is stated below.

**Theorem 2.2** (Tower property)**.** *For all random variables $X$ and $Y$,*

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$$

*That is, if $Y$ is a discrete random variable,*

$$\mathbb{E}[X] = \sum_y \mathbb{E}[X|Y=y]\,\mathbb{P}[Y=y]$$

*and if $Y$ is continuous, we have*

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} \mathbb{E}[X|Y=y]\, f_Y(y)\, dy$$

Observe that the tower property is similar to the law of total probability. The difference is that now we use it to compute an expected value instead of a probability. Let's see an example for a discrete random variable.

**Example 2.7.** *Use conditional expectation to compute the mean of a geometric random variable.*

**Solution.** Let $N$ be the geometric random variable, that is, the number of experiments until the first success of a Bernoulli random variable $Y$. Then, we condition on the first outcome and obtain:

$$\mathbb{E}[N] = \mathbb{E}[N|Y=1]\,\mathbb{P}[Y=1] + \mathbb{E}[N|Y=0]\,\mathbb{P}[Y=0]$$

When $Y=1$, the number of experiments is 1, and when $Y=0$, the number of experiments until the first success has the same distribution as $N$. Then, we obtain

$$\mathbb{E}[N] = 1 * \mathbb{P}[Y=1] + (1 + \mathbb{E}[N]) * \mathbb{P}[Y=0]$$

Observe that we obtained a recursive equation. Using that $\mathbb{P}[Y=1] = p$ and $\mathbb{P}[Y=0] = 1-p$, we obtain

$$\begin{aligned} &\mathbb{E}[N] = p + (1 + \mathbb{E}[N])(1-p) \\ \implies &p\mathbb{E}[N] = p + (1-p) \\ \implies &\mathbb{E}[N] = \frac{1}{p} \end{aligned}$$

□

**Example 2.8.** *Let $X_1, X_2, \ldots$ be independent and identically distributed random variables, and $N$ be a discrete random variable independent of $X_i$'s. Compute*

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right]$$

This example looks theoretical (and fun, for sure). However, it has many practical applications. For example, $N$ can the the number of customers in a supermarket and $X_i$ can be the amount of $ they are buying from the store.

**Solution.** The first thing I'd like to do when I see this problem is using linearity of expectation. However, we cannot do it right away because the number of elements in the sum is random. Then, if we do

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = N\mathbb{E}[X_1]$$

we obtain a random variable instead of a number.

However, we are not too far away from the right path. All we need to do is conditioning on the value of $N$ before we apply the linearity property. We have:

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{N} X_i \middle| N\right]\right]$$

When we compute $\mathbb{E}\left[\sum_{i=1}^{N} X_i | N\right]$, we are treating $N$ as a number and, hence, we can apply linearity of expectation to obtain

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] &= \mathbb{E}\left[N\mathbb{E}\left[X_1|N\right]\right] \\ &= \mathbb{E}\left[N\mathbb{E}\left[X_1\right]\right] \qquad \text{(because } X_1 \text{ and } N \text{ are independent)} \\ &= \mathbb{E}\left[X_1\right]\mathbb{E}\left[N\right] \qquad \text{(linearity of expectation)}\end{aligned}$$

□

# 3 Discrete-Time Markov Chains

So far, we've been discussing random variables without thinking what happens next in each application. For example, we computed the mean of a geometric random variable, but we haven't considered the possibility that after the first success we might want to start the process again. In this section we include time as a variable of interest, and we study a model that will help us analyze the evolution of a probabilistic system. We start with a definition.

**Definition 3.1.** *A stochastic process $\{X(t) : t \in \mathcal{T}\}$ is a collection of random variables indexed by $t$ (which usually represents time), that is, for each $t \in \mathcal{T}$, $X(t)$ is a random variable.*

*The set $\mathcal{T}$ is called the index set of the process, and depending on its elements the stochastic process can be discrete or continuous.*

- *If $\mathcal{T}$ has countably many elements, $\{X(t) : t \in \mathcal{T}\}$ is called a discrete-time process*
- *If $\mathcal{T}$ has uncountably many elements, $\{X(t) : t \in \mathcal{T}\}$ is called a continuous-time process*

For example,

- The position of a frog that randomly jumps among 10 lily pads after $n$ jumps is a discrete-time stochastic process
- The number of customers in a store during Black Friday is a continuous-time stochastic process
- The number of patients in a hospital unit **at every time** $t$ is a continuous-time stochastic process
- The number of patients in a hospital unit **every day at 8 am** is a discrete-time stochastic process

As hinted by the examples above, most of the time the stochastic processes are not intrinsically continuous or discete-time processes. Instead, we make a decision to model them and study them in the most appropriate way. In this part of the course we will focus on discrete-time processes. Specifically, Markov chains.

**Definition 3.2.** *A Discrete-Time Markov Chain (DTMC) is a stochastic process $\{X_n : n \in \mathbb{Z}_+\}$, where $X_n$ denotes the state at time step $n$, that satisfies:*

*(i) Markov property:*
*For all $n \in \mathbb{Z}_+$, all $i, j \in \Omega$ and all $i_0, i_1, \ldots, i_{n-1} \in \Omega$ we have*

$$\mathbb{P}\left[X_{n+1} = j \,|\, X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\right] = \mathbb{P}\left[X_{n+1} = j \,|\, X_n = i\right]$$

*(ii) Stationarity:*
*For all $n \in \mathbb{Z}_+$ and all $i, j \in \Omega$,*

$$\mathbb{P}\left[X_{n+1} = j \,|\, X_n = i\right] = P_{i,j},$$

*where $P_{i,j}$ is independent of $n$.*

The first property states that the probability mass function of the future state $X_{n+1}$ conditioned on all the history, only depends on the latest information we have, that is, on the present state $X_n$. The slogan of the Markov property is:

*The future depends on the past only through the present state.*

The stationarity property indicates that all we care about when computing the probability mass function of $X_{n+1}$ is the state $X_n$, but not the time index. That is, we care about where we are now, but not what time it is. The notation we introduced in the definition hints the use of a matrix to represent these probabilities.

**Definition 3.3.** *The transition probability matrix asociated with a DTMC is a matrix $P$ whose element $(i, j)$ represents the probability of going from state $i$ to $j$ in one step.*

When we are in state $i$, we must be at some state after one step. Then,

$$\sum_j P_{i,j} = 1 \qquad \forall i$$

Therefore, the $i^{\text{th}}$ row of the transition matrix $P$ represents the conditional probability mass function of the next step given that the current state is $i$.

**Example 3.1.** *In the following situations, determine if the process described can be modeled as a DTMC. If yes, determine the state space and the transition probability matrix of the DTMC.*

*(a) A machine is either working or in the repair center. If it is working today, then there is a 95% chance that it will be working tomorrow. If it is at the repair center today, then there is a 40% chance that it will be working tomorrow. We are interested in studying what fraction of time does the machine spend in the repair center.*

*(b) Suppose that whether it rains or not today depends on previous weather conditions through the last two days. Specifically, suppose that if it has rained for the past two days, then it will rain tomorrow with probability 0.7; if it rained today but not yesterday, then it will rain tomorrow with probability 0.5; if it rained yesterday but not today, then it will rain tomorrow with probability 0.4; if it has not rained in the past two days, then it will rain tomorrow with probability 0.2.*

*(c) Let's consider a single-server queue modeled in discrete time. Every one minute, the probability that a customer enters a store is $\lambda$, and the probability that a customer leaves the store is $\mu$ (unless is empty). For simplicity, let's assume that no more than 1 customer can arrive to the store or leave the store within 1 minute. Additionally, arrivals and departures in each minute are independent of each other, and independent of what happened previously. We are interested in the number of customers at the store.*

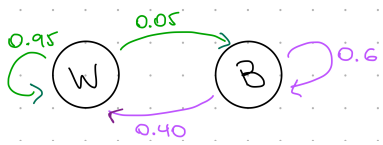
*(d) Consider again the single-server queue in discrete time described above. What changes if we are now interested in the number of customers waiting in line?*

**Solution.**

(a) The state of the machine is either working or not broken, and the future state only depends on the current state. Then, the process is a DTMC. The state space is $\Omega = \{W, B\}$, then, the transition matrix is

$$P = \begin{bmatrix} \mathbb{P}[X_1 = W | X_0 = W] & \mathbb{P}[X_1 = B | X_0 = W] \\ \mathbb{P}[X_1 = W | X_0 = B] & \mathbb{P}[X_1 = B | X_0 = B] \end{bmatrix} = \begin{bmatrix} 0.95 & 0.05 \\ 0.40 & 0.60 \end{bmatrix}$$

We might as well represent the transitions with a graph, where the nodes are the states and the edges represent the transition probabilities. In this case, we have



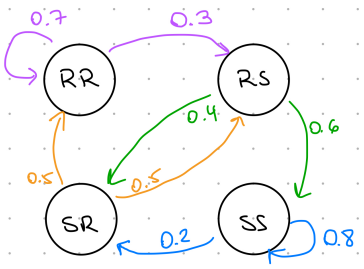
(b) If we model the states as whether it rains or not today, then the process described is not a DTMC because the future state depends on the present and the past. However, we can define two-dimensional states such that the current state says whether it rains today and yesterday or not. Specifically, our states can be:

$$RR, RS, SR, SS$$

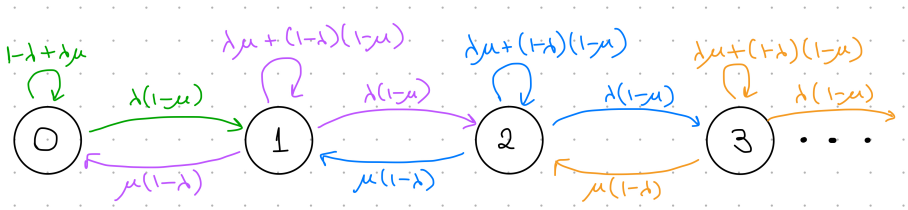
Then, both properties of Markov chains are satisfied and we can compute the transition matrix. Observe that the future state contains the weather today and tomorrow, and the current state contains the weather yesterday and today. Then, the second element of the current state must match the first element of the future state to obtain a meaningful transition. Hence, we have

$$P = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 \end{bmatrix}$$

and graphically, we have




(c) If we know the current number of customers in the system, we have all the information about the state and we can compute the probability that this number increases by 1, decreases by 1, or remains the same. Further, these probabilities do not depend on the number of transitions so far. Hence, the current number of customers in the system can be represented by a DTMC. The state space is the set of nonnegative integers, and the diagram is




and the transition probabilities can be thought of as follows:

- From one ste to next, the number of customers in the system can only change by 1 or stay the same
- If we are in state 0 (empty system), there can be an arrival with probability $\lambda$. If there is an arrival and no departure, we advance to state 1 and if not, we stay at 0.
- If we are currently in state 1, there can be an arrival, a departure or both. We move to state 2 if there is an arrival and no departure, and this happens with probability $\lambda(1-\mu)$ because the arrivals are independent from departures. Similarly, we move to state 0 if there is a departure and no arrival, and this happens with probability $\mu(1-\lambda)$. Finally, we stay in state 1 if there is an arrival and a departure, or if there is no arrival and no departure. Then, the probability of staying in state 1 is $\lambda\mu+(1-\lambda)(1-\mu)$.
- For all states $i \geq 1$, we use the same logic as in state 1.

Since there are infinitely many states, the transition matrix has infinitely many rows and columns. We can write it as a matrix (with many dots), or we can define it by cases, as follows:

$$P_{0,0} = 1 - \lambda + \lambda\mu$$

$$
\begin{aligned}
P_{i,i+1} &= \lambda(1-\mu) \qquad \forall i \geq 0 \\
P_{i,i-1} &= \mu(1-\lambda) \qquad \forall i \geq 1 \\
P_{i,i} &= \lambda\mu + (1-\lambda)(1-\mu) \qquad \forall i \geq 1 \\
P_{i,j} &= 0 \qquad \forall i,j : |j-i| \geq 2
\end{aligned}
$$

(d) If we only look at the waiting customers, we expect the evolution from state to state to be the same as in part (c), except when the queue is empty. If there are no waiting customers and we only look at the queue, we don't know if there is one customer in service or not. If there is a customer in service, then the probability of jumping to state 1 is $\lambda(1-\mu)$. However, if there is no customer in service, the same probability becomes 0. Hence, the number of waiting customers is **not** a DTMC because the state does not completely determine the transition probabilities.

□

## 3.1 $n$-step transition matrix

Using the law of total probability, observe that the probability mass function of a DTMC after one step is

$$
\mathbb{P}[X_1 = j] = \sum_i \mathbb{P}[X_1 = j | X_0 = i]\, \mathbb{P}[X_0 = i] \qquad \forall j
$$

Then, if we use $\boldsymbol{p}^{(1)}$ a vector such that $p_j^{(1)} = \mathbb{P}[X_1 = j]$ we have

$$
\boldsymbol{p}^{(1)} = P^T \boldsymbol{p}^{(0)} \tag{1.1}
$$

and we can use the same argument to obtain

$$
\begin{aligned}
\boldsymbol{p}^{(2)} &= P^T \boldsymbol{p}^{(1)} \\
&= (P^2)^T \boldsymbol{p}^{(0)} \qquad \text{(using (1.1))}
\end{aligned}
$$

If we continue in this fashion, we obtain

$$
\boldsymbol{p}^{(n)} = \left(P^n\right)^T \boldsymbol{p}^{(0)} \qquad \forall n \geq 1, \tag{1.2}
$$

that is, the $n$-step transition matrix equals $P^n$. Let's see a simple example.

**Example 3.2.** *Consider the example of the machine that either works or it is at the repair center, that is, Example 3.1 part (a). What is the probability that the machine is working on Friday, given that it was working on Monday?*

**Solution.** There are 4 transitions between Monday and Friday. Then, we compute

$$
\boldsymbol{p}^{(4)} = \left(P^4\right)^T \boldsymbol{p}^{(0)}
$$

Since the machine is working on Monday, we have

$$
\boldsymbol{p}^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}
$$

Next, we use some software to compute $P^4$ and obtain

$$
P^4 = \begin{bmatrix} 0.90 & 0.10 \\ 0.81 & 0.19 \end{bmatrix}
$$

Then,

$$
\boldsymbol{p}^{(4)} = \begin{bmatrix} 0.90 & 0.81 \\ 0.10 & 0.19 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.10 \end{pmatrix}
$$

Hence, the probability that the machine is working on Friday is 0.90. □

## 3.2 Limiting distribution

The analysis above is useful and simple when we have finite state space DTMC's and want to study what's going to happen a few steps ahead of time. However, we might be interested in long-term questions. For example, in the machine example, the original question was what's the proportion of time that the machine works. Additionally, when we have DTMCs with infinite state space (such as the single-server queue example), matrix multiplication becomes considerably harder and we might want to study the long-term behavior to get some insights.

When we say long-term behavior, we mean that $n \to \infty$ in (1.2). Specifically, does $\lim_{n\to\infty} P^n$ exist? If yes, does $\lim_{n\to\infty} \boldsymbol{p}^{(n)}$ depend on $\boldsymbol{p}^{(0)}$? The answer to both questions is "it depends."

Let's continue with the machine example to understand the meaning of $\lim_{n\to\infty} P^n$. If we look at the 10-step transition matrix, we obtain

$$P^{10} = \begin{bmatrix} 0.88917 & 0.11083 \\ 0.88664 & 0.11336 \end{bmatrix}$$

and if we compute the 30-step transition matrix we obtain

$$P^{30} = \begin{bmatrix} 0.88889 & 0.11111 \\ 0.88889 & 0.11111 \end{bmatrix}$$

Observe that, as we increase the number of days ahead, the rows of $P^n$ become more and more similar. Recall that the $i^{\text{th}}$ row of $P^n$ is the probability mass function of the $n^{\text{th}}$ step conditioned on $X_0 = i$. Then, if the rows become similar, the probability mass function after $n$ steps becomes independent of the state where we start and, hence, independent of $i$.

**Definition 3.4.** *Assuming that the limit is independent of $i$, define*

$$\pi_j = \lim_{n\to\infty} P^n_{ij}, \tag{1.3}$$

*where $P^n_{ij}$ is the element $(i, j)$ of $P^n$. Then, $\pi_j$ represents the limiting probability that the DTMC is in state $j$ in the long term. Let $\boldsymbol{\pi}$ be a vector with elements $\pi_j$. Then,*

$$\sum_j \pi_j = 1$$

For the time being, let's assume that the limit

$$\lim_{n\to\infty} P^n_{i,j}$$

exists and it is independent of $i$. We will focus on conditions for its existence later. We now learn how to compute the limit.

Computing the limit of the powers of a matrix can be very hard. Instead of following that path, we will learn an alternative that uses the meaning of limiting distribution. As $n$ grows, we should expect that $P^n$ becomes very similar to $P^{n+1}$ and all the rows become the same. Indeed, you can verify that in the machines example, we have $P^{30} = P^{31}$ and the rows of the matrices are equal to each other. Then, we should also have $\boldsymbol{p}^{(n)} \approx \boldsymbol{p}^{(n+1)}$. Further, since the limit of $P^n$ does not depend on the initial condition, we should expect to observe $\boldsymbol{\pi} = \boldsymbol{p}^{(n)} = \boldsymbol{p}^{(n+1)}$. With this motivation, we have the following definition.

**Definition 3.5.** *A probability mass function $\boldsymbol{\pi}$ is said to be stationary if*

$$\boldsymbol{\pi} = P^T\boldsymbol{\pi}, \quad \text{and} \quad \sum_j \pi_j = 1$$

The first equation represents that one transition does not change the probability mass function; hence the name stationary. Further, if the limit (1.3) exists, the limiting and stationary distribution are the same.

Let's see a few examples before we find the conditions on the existence of the limit (1.3).

**Example 3.3.** *Let's revisit Example 3.1 part (a). We already know that $P^{30}$ satisfies the limiting condition, so we suspect that a limiting distribution exists and equals*

$$\boldsymbol{\nu} = \begin{pmatrix} 0.89 \\ 0.11 \end{pmatrix}$$

*Compute the stationary distribution and verify that it equals $\boldsymbol{\nu}$.*

**Solution.** We solve the system of equations. We have:

$$\boldsymbol{\pi} = P^T\boldsymbol{\pi},\ \sum_j \pi_j = 1$$

$$\iff \begin{cases} \pi_1 = 0.95\pi_1 + 0.40\pi_2 \\ \pi_2 = 0.05\pi_1 + 0.60\pi_2 \\ \pi_1 + \pi_2 = 1 \end{cases}$$

Observe that the first two equations are redundant. Then, we omit one of them and use that $\pi_1 + \pi_2 = 1$. From the first equation, we obtain:

$$\begin{aligned} & 0.05\pi_1 = 0.40\pi_2 \\ \implies & \pi_1 = 8\pi_2 \end{aligned}$$

Then, using this result in the last equation, we obtain:

$$\begin{aligned} & \pi_1 + \pi_2 = 1 \quad \& \quad \pi_1 = 8\pi_2 \\ \implies & 9\pi_2 = 1 \\ \implies & \pi_2 = \frac{1}{9} \approx 0.11111 \quad \& \quad \pi_1 = \frac{8}{9} \approx 0.88889 \end{aligned}$$

□

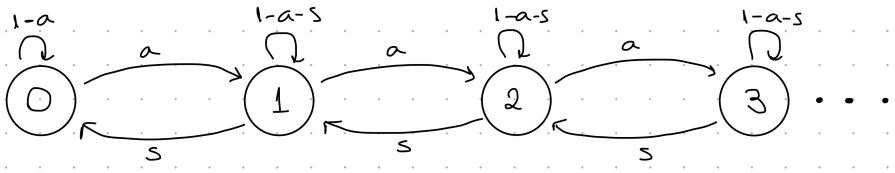
Before we go to the conditions, let's compute the stationary distribution of a single-server queue.

**Example 3.4.** *Let's revisit Example 3.1 part (c). Compute the stationary distribution of customers in the system.*

**Solution.** To simplify notation, let's define

$$a = \lambda(1-\mu) \quad \text{and} \quad b = \mu(1-\lambda)$$

Then, the diagram becomes




The next step is writing the system of equations $\boldsymbol{\pi} = P^T\boldsymbol{\pi}$. One way to think about the right-hand side of the $j^{\text{th}}$ equation is to consider all the paths that lead to state $j$. In this case, we obtain

$$\begin{aligned} \pi_0 &= (1-a)\pi_0 + s\pi_1 \\ \pi_1 &= a\pi_0 + (1-a-s)\pi_1 + s\pi_2 \\ &\vdots \\ \pi_j &= a\pi_{j-1} + (1-a-s)\pi_j + s\pi_{j+1} \end{aligned}$$

From the first equation, we obtain

$$\pi_1 = \frac{a}{s}\pi_0$$

Then, using this result in the second equation and reorganizing terms yields

$$\pi_2 = \left(\frac{a}{s}\right)^2 \pi_0$$

If we follow the same steps with the rest of the equations, we observe

$$\pi_j = \left(\frac{a}{s}\right)^j \pi_0 \qquad \forall j$$

Next, we plug this result in $\sum_j \pi_j = 1$ and obtain:

$$\sum_{j=0}^{\infty} \pi_j = \sum_{j=0}^{\infty} \left(\frac{a}{s}\right)^j \pi_0 = \frac{\pi_0}{1 - \frac{a}{s}}$$

where the last equality holds if $a < s$, which translates to $\lambda < \mu$. Therefore, we obtain

$$\pi_0 = 1 - \frac{a}{s} \quad \text{and} \quad \pi_j = \left(\frac{a}{s}\right)^j \left(1 - \frac{a}{s}\right)$$

In this computation we included the assumption $\lambda < \mu$, which means that the probability of an arrival is smaller than the probability of a departure. Otherwise, we would not be able to obtain the stationary distribution. This requirement makes sense because, if $\lambda > \mu$, then the server would not be able to manage all the incoming customers and, hence, the queue length would grow to infinity. In such case, talking about a distribution does not make sense. The case $\lambda = \mu$ is a bit tricky, and we'll discuss it later. □

## 3.3 Classification of states

The goal of this section is to understand the properties of the states and transition probabilities that ensure the existence of a limit distribution. We already saw two examples where it exists and matches the stationary distribution. To motivate this section, let's see two examples where these are not true.

**Example 3.5.** *In the following two examples we provide the state space and the transition matrix of a DTMC. In both cases, the initial state is $X_0 = 0$. Discuss the existence of a stationary and a limiting distribution.*

*(a) The state space is $\Omega = \{0, 1\}$ and the transition matrix is*

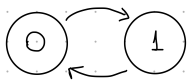
$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

*(b) The state space is $\Omega = \{0, 1, 2\}$ and the transition matrix is*

$$P = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Solution.** In each case, we analyze the limit and compare with the stationary distribution.

(a) The diagram of this chain is




Observe that in each step, we know exactly what's the next state. Since we are told that the chain starts at state $X_0 = 0$, we know that

$$X_n = \begin{cases} 1 & \text{, if } n \text{ is odd} \\ 0 & \text{, if } n \text{ is even} \end{cases}$$

Indeed, computing the powers of the transition matrix $P$ we observe

$$P^{2n-1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{and} \quad P^{2n} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \forall n \geq 1$$

Therefore, $\lim_{n\to\infty} P^n$ does not exist. You can think of it as computing $\lim_{n\to\infty} (-1)^n$.

Let's see what happens if we try to compute the stationary distribution. The system of equations $\boldsymbol{\pi} = P^T \boldsymbol{\pi}$ yields

$$\pi_0 = \pi_1$$

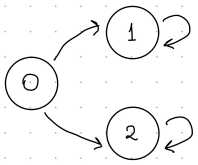
$$\pi_1 = \pi_0$$

Then, plugging this result in $\pi_0 + \pi_1 = 1$ we obtain

$$\pi_0 = \pi_1 = \frac{1}{2}$$

**How come the stationary distribution exists if the limiting distribution does not exist?**
Well, the stationary distribution represents the proportion of time that we spent in every state. It is not a limit. In this case, since we jump from 0 to 1 and from 1 to 0 with probability 1, the proportion of time spent in each state in the long run is indeed $\frac{1}{2}$. However, that does **not** mean that the **probability** of being in one state or the other is $\frac{1}{2}$. Actually, in odd steps we are at state 1 with probability 1, and in even steps, at state 0 with probability 1.

(b) The diagram of this chain is




To study the limiting distribution we compute powers of the transition matrix $P$. Observe

$$P^2 = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = P$$

Then,

$$P^3 = P^2 P = PP = P^2 = P$$

where we used that $P^2 = P$ twice. Therefore, $P^n = P$ for all $n$ and the limit is

$$\lim_{n\to\infty} P^n = \lim_{n\to\infty} P = P = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So it exists. However, the rows of the limiting matrix are not equal to each other. Therefore, the probability of transitioning to state 1 or 2 depends on the initial state; even after $n \to \infty$ transitions. Intuitively, this happens because once we enter state 1 (or 2), it is impossible to go to any other state. This example shows the importance of $\lim_{n\to\infty} P_{ij}^n$ being independent of $i$.

Let's now compute the stationary distribution. From $\boldsymbol{\pi} = P^T \boldsymbol{\pi}$ we obtain

$$\begin{aligned} \pi_0 &= 0 \\ \pi_1 &= 0.5\pi_0 + \pi_1 \\ \pi_2 &= 0.5\pi_0 + \pi_2 \end{aligned}$$

Using that $\pi_0 = 0$ in the other two equations we obtain the redundant equations $\pi_1 = \pi_1$ and $\pi_2 = \pi_2$. Using these results in $\pi_0 + \pi_1 + \pi_2 = 1$ we obtain

$$\pi_1 + \pi_2 = 1,$$

which does not have a unique solution. Therefore, the stationary distribution does not exist.

□

These examples illustrate the importance of understanding the structure of the transition matrix to detect whether we can analyze the DTMC's in the long run or not. In the rest of this section we formalize these conditions. We start with a definition that will help us detect the first problem above, that is, when the limiting distribution does not exist.

**Definition 3.6.** *The period of state $j$ is the greatest common divisor (GCD) of the set of integers $n$ such that $P_{jj}^n > 0$.*

*(i) If the period of a state is 1, then the state is called aperiodic. If all the states in a DTMC are aperiodic, we say that the chain is aperiodic too.*

*(ii) If the period of a state is larger than 1, then the state is called periodic*

For example, in Example 3.5(a), both states have period 2. When a state has a self-loop, it is always aperiodic. However, there are some cases where there is no self loop and it is still aperiodic. From this example and analysis, we conclude that aperiodic chains are more likely to have a limiting distribution.

Now let's discuss what happened in the second case.

**Definition 3.7.**

*(i) A state $j$ is accessible from state $i$ if $P_{ij}^n > 0$ for some $n \geq 1$.*

*(ii) States $i$ and $j$ communicate if $i$ is accessible from $j$ and $j$ is accessible from $i$.*

*(iii) Two states that communicate are said to be in the same class.*

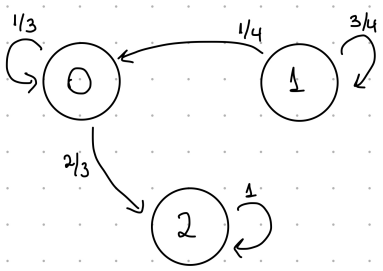
*(iv) A DTMC is irreducible if it has only one class, that is, if all its states communicate with each other.*

For example, in Example 3.5(a), the states 0 and 1 communicate and the chain is irreducible. In Example 3.5(b), instead, states 1 and 2 are accessible from state 0, but not viceversa. Indeed, there are two classes: state 1 alone and state 2 alone.

**Example 3.6.** *Consider DTMC's with state space $\Omega = \{0, 1, 2\}$ and the transition matrices specified below. Determine the period of each state, the classes, and whether they are irreducible.*

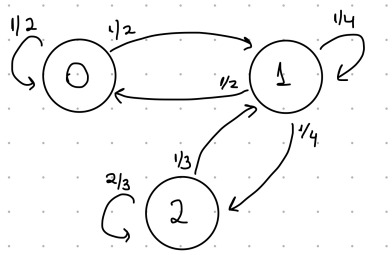
$$P = \begin{bmatrix} 1/3 & 0 & 2/3 \\ 1/4 & 3/4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 1/4 & 1/4 \\ 0 & 1/3 & 2/3 \end{bmatrix}$$

**Solution.** In each case, we first draw the diagram. For the transition matrix $P$, we obtain




The three states are aperiodic because they have a self-loop. Regarding classes, each of the three states only communicates with itself. Then, we have three classes: $C_1 = \{0\}$, $C_2 = \{1\}$ and $C_3 = \{2\}$. Further, since there are three classes, the chain is not irreducible.

For the transition matrix $Q$, we have the following diagram:



Again, all the states have a self loop, so they are aperiodic. Additionally, all the states communicate with each other. Therefore, they belong to the same class and the chain is irreducible. □

The last feature we need to measure is the probability that the DTMC comes back to a state after a finite period of time. For example, in Example 3.6 we observe that the matrix $P$ yields a chain for which the probability of coming back to state 0 or 1 is strictly smaller than 1. That is, there is a chance that the chain will never come back to states 0 and 1 because they are not accessible from state 2. In the DTMC associated to $Q$, instead, all states are accessible from all states. Hence, there is a high probability of revisiting all the states.

**Definition 3.8.** *Let $f_j$ be the probability that a chain in state $j$ ever returns to state $j$. Mathematically,*

$$f_j \triangleq \mathbb{P}\left[X_n = j \text{ for some } n \geq 1 \,|\, X_0 = j\right]$$

*(i) If $f_j < 1$, then $j$ is a transient state*

*(ii) If $f_j = 1$, then $j$ is a recurrent state*

*(iii) If a state $j$ is recurrent and the expected time between visits to $j$ is finite, then the state is positive recurrent*

*(iv) If a state $j$ is recurrent and the expected time between visits to $j$ is infinite, then the state is null recurrent*

Matrix $P$ in Example 3.6 has two transient states (0 and 1) and one recurrent state (state 2). Further, the time between visits to state 2 equals 1 step, so the state is positive recurrent. Matrix $Q$, instead, yields a DTMC where all the states are recurrent. Since there are finitely many states, the probability that the time between visits to a specific state is infinite is negligible. Hence, all the states are positive recurrent.

We end this section establishing some properties of the classifications of states, and we later establish when the limiting distribution exists and matches the stationary distribution.

**Theorem 3.1.**

*(i) The period, transience, null recurrence and positive recurrence are class properties, that is, all the states of a class satisfy the same properties.*

*(ii) In finite-state DTMCs, the classes can be either transient or positive recurrent.*

*(iii) In an irreducible finite-state DTMC, all the states are positive recurrent.*

Hence, when analyzing DTMC's, the first step is identifying the classes. We close this chapter with a limiting theorem and the notion of expected time between visits.

**Theorem 3.2.** *An irreducible, aperiodic DTMC belongs to one of the following classes:*

*(i) All the states are transient. In this case,*

$$\pi_j = \lim_{n\to\infty} P_{ij}^n = 0 \qquad \forall j$$

*and there does not exist a stationary distribution.*

*(ii) All the states are null recurrent. In this case,*

$$\pi_j = \lim_{n\to\infty} P_{ij}^n = 0 \qquad \forall j$$

*and there does not exist a stationary distribution.*

*(iii) All the states are positive recurrent. Then, the limiting distribution $\boldsymbol{\pi}$ exists and there is a positive probability of being in each state. Here,*

$$\pi_j = \lim_{n\to\infty} P_{ij}^n > 0 \qquad \forall j$$

*is the limiting probability of being in state $j$. Further, $\boldsymbol{\pi}$ is the unique stationary distribution, that is, solves $\boldsymbol{\pi} = P^T\boldsymbol{\pi}$, $\sum_j \pi_j = 1$.*

*Further, defining $m_{jj}$ as the mean number of steps between visits to state $j$, we have*

$$m_{jj} = \frac{1}{\pi_j} \qquad \forall j$$

**Example 3.7.** *Each morning an individual leaves his house and goes for a run. He is equally likely to leave either from his front or back door. Upon leaving the house, he chooses a pair of running shoes (or goes running barefoot if there are no shoes at the door from which he departed). On his return he is equally likely to enter, and leave his running shoes, either by the front or back door. Suppose he owns a total of 2 pairs of running shoes.*

*(a) Model the situation as a DTMC and specify the transition probabilities.*

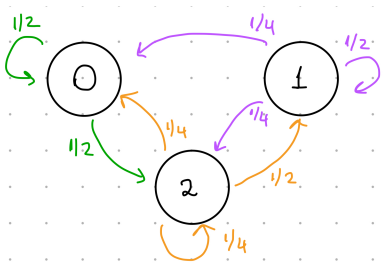
*(b) Determine the classes of the DTMC and classify each class in transient/null recurrent/positive recurrent, and periodic/aperiodic. Is the DTMC irreducible?*

*(c) Does a limiting distribution exist? Explain. If yes, compute it.*

*(d) If the individual runs barefooted today, what is the expected time until he runs barefooted again?*

**Solution.**

(a) The key to solve this problem easily is to define the state strategically. In this case, let $X_n$ be the number of pairs of shoes at the door where the individual starts his run in the morning of day $n$. Then, the diagram is




and the transition matrix is

$$P = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

To compute these numbers, we use the following steps:

- If $X_n = 0$, it means the runner ran barefooted this morning. Then, he won't change the number of shoes in each door when he comes back. However, the next morning he has the same probability of choosing the same door (with no shoes) or the other door (with 2 shoes).
- If $X_n = 1$, the runner will come back to the same door where he left with probability $1/2$, or to the other one with probability $1/2$ too. If he comes to the same door, the next morning there is 1 pair of shoes in each door. Hence, the probability of transitioning from state 1 to 1 is $1/2$. If he changes the door when he comes back, there will be two pairs of shoes in one door, and none in the other door. Then, he will have probability $1/2 * 1/2$ of finding no shoes at the door, and probability $1/2 * 1/2$ of finding two shows.
- Similarly, if $X_n = 2$, the runner will come back to the same door or the other one with probability $1/2$. If he comes back to the same door, the next morning one of the doors will have 2 pairs of shoes, and the other one none. Then, he will have probability $1/2 * 1/2$ of finding 0 or 2 shoes in the door he picks. If he comes back with the other door, each door will have one pair of shoes the next morning. Hence, the probability of finding 1 pair of shoes at the door the next morning is $1/2$.

(b) There is only one class. All the states are positive recurrent and aperiodic.

(c) According to Theorem 3.2, the limiting distribution exists and matches the stationary distribution. To compute it, we solve

$$\boldsymbol{\pi} = P^T \boldsymbol{\pi}$$

$$\iff \begin{cases} \pi_0 = \frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 + \frac{1}{4}\pi_2 \\ \pi_1 = \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2 \\ \pi_2 = \frac{1}{2}\pi_0 + \frac{1}{2}\pi_1 + \frac{1}{4}\pi_2 \end{cases}$$

Reorganizing terms, we obtain

$$\pi_0 = \pi_1 = \pi_2$$

Then, using that $\pi_0 + \pi_1 + \pi_2 = 1$, we obtain

$$\pi_0 = \pi_1 = \pi_2 = \frac{1}{3}$$

(d) Observe that the probability that the individual runs barefooted today is $\pi_0$. Then, the expected number of days until he runs barefooted again is

$$m_{00} = \frac{1}{\pi_0} = 3$$

□

The last example is very special because the columns of the transition matrix $P$ also add up to 1. A matrix whose rows and columns add up to 1 is called a doubly stochastic matrix, and it can be proved that a DTMC with $M$ states and a doubly stochastic transition matrix always satisfies

$$\pi_j = \frac{1}{M} \qquad \forall j$$

# Chapter 2

# Reinforcement Learning

Reinforcement Learning (RL) is an area of machine learning where an *agent* takes *actions* and interacts with an *environment* to fulfill a goal. To discover which actions help to reach the goal, the agent receives *rewards* that encourage good actions and discourage bad actions.

RL is different from supervised and unsupervised learning. In supervised learning, there is a training set of labeled data, and it is used to extrapolate responses to situations outside the training set; in unsupervised learning, there is a data without any label, and the goal is to find patterns. RL, instead, seeks to maximize the reward to reach a specific goal. In that sense, it is similar to human and animal behavior. The reward must be structured so that the agent can learn which actions help while it interacts with the environment.

There are many contexts where RL can be used, but here are three examples with their link to a YouTube video:

- Cart pole: A cart with an attached pole can move to the left or right to lift the pole and balance it pointing upwards.
- Escape room: An artificial intelligence must figure out how to escape a room, by finding the exit, opening the door and going though it. There are several rooms, and as the AI moves on, the difficulty increases.
- Self-driving cars: A car that goes forward must stay in its lane while turning right and then left.

To have a meaningful RL problem, the agent needs a clear goal and needs to actively interact with the environment. Then, it must be able to monitor the environment and react accorgingly. Depending on the problem, the actions the agent take may affect the immediately next step, or steps in the future too. The whole goal is that the agent *learns*, that is, uses its experience to improve performance over time.

One of the key challenges in RL (and machine learning in general) is the trade-off between exploration and exploitation. Exploitation means that the agent should prefer actions that are known to give a high reward. Exploration, instead, means that the agent should take some random actions to learn if they are better than the already known actions. Then, to maximize the reward the agent must try a variety of actions (exploration) while progressively favoring actions that appear to be the best (exploitation). There is no unique answer to this trade-off, and a good equilibrium between both depends on each problem.

Before we get into more technical knowledge, let's identify the key elements of RL.

**Definition 3.9.**

*(i) A policy is a mapping from states to actions to be taken. In other words, a policy tells us which action to take at every possible state. Policies might be deterministic or stochastic.*

*(ii) A reward signal defines the objective function, by evaluating how good each action is at each step. It represents the immediate feedback that the agent perceives after every action.*

*(iii) The value function is a function of the state, and represents the long-term value of each state. In other words, the expected future reward starting from the current state.*

*(iv) A model of the environment tells us the probabilities of transitioning from one state to another. It might be known or unknown.*

The following table shows the agent, environment, possible actions, possible policies and rewards of the three examples we saw before.

| Example | Agent | Environment | Actions | Policies | Rewards |
|---|---|---|---|---|---|
| Cart pole | The cart | Horizontal bar, gravity | Move left or right, and at which speed | If the bar is low, move fast. Otherwise, move slowly and to the opposite direction of the tip | Height of the tip |
| Escape room | Albert | Each room, the green square, the door | Go left, right, forward, backward, jump | If the door is closed, go to the green square. If it's open, go through it | Distance to the door, some reward for opening the door |
| Self-driving car | Car | The lane and the grass around | Turn right, turn left and how much. Maybe increase or decrease speed. | If approaching end of lane on the left, turn right. If approaching end of lane on right, turn left. Decrease speed while turning. | Negative reward for being too close to the grass, and a more negative value if touches the grass. |

In the next few chapters we will learn some examples of RL tabular methods, that is, where we can explicitly write the value function and the rewards of every state/action pair.

# 4 Multi-Armed Bandits

In this section we learn one of the most fundamental problems in RL. The beauty of this problem is that, even though is a simple problem, illustrates well the need for exploration.

In RL, the agent must be trained to evaluate its actions without any instruction, and detect which actions will help the most to maximize the reward. We will learn some basic learning methods that serve as foundation of more complex RL methods.

## 4.1 A $k$-armed bandit problem

The name of this problem is inspired in a gambler at a row of slot machines. The gambler is allowed to use $k$ machines, wants to bring home as much money as possible and is allowed to play at most $T$ times. Below is a more generic definition.

**Definition 4.1.** *A k-armed bandit problem repeatedly faces the choice among $k$ actions. After each action, the agent receives a reward that comes from a stationary distribution that only depends on the action selected. The long-term goal is to maximize the total reward over a fixed number of actions $T$.*

From the definition, we immediately see that the agent needs to concentrate its actions on the best levers, that is, the ones that will likely give higher rewards. Let's introduce some notation to study this problem.

**Definition 4.2.**

*(i) The value of an action is the expected reward given that that action is selected. If we use $A_t$ do denote the action selected at time step $t$, and $R_t$ the reward received at time step $t$, then the expected reward can be defined as*

$$q_*(a) = \mathbb{E}\left[R_t \mid A_t = a\right]$$

*(ii) The estimated value of action $a$ at time step $t$ represents our estimation of $q_*(a)$ with the information we have up to time step $t$, and is denoted by $Q_t(a)$.*

Of course, we would like $Q_t(a)$ to be close to $q_*(a)$ and, further, we would like to improve the estimate as $t$ increases. At any step $t$, we have an estimate of each of the possible actions $a$ and, as a consequence, we can order the actions from best to worst according to their value.

**Definition 4.3.** *A greedy action at time step $t$ is an action whose estimated value is the greatest. That is, an action $\bar{a}$ such that for all other actions $a$, we have $Q_t(\bar{a}) \geq Q_t(a)$.*

When we select a greedy action, we are *exploiting* our current knowledge. If we select a nongreedy action instead, we say that we are *exploring* because this choice helps us improve our estimation of the actions' value. Exploitation is the right thing to do to maximize the expected reward on one step, but exploration may produce a greater total reward in the long run. It is impossible to explore and exploit with the same action, so we must choose what to do in each step.

Whether is better to explore or exploit in each step depends on many factors, such as remaining time, precise values of the estimates, and the inherent uncertainties of each problem. There are sophisticated mathematical methods to deal with this trade-off, but they involve strong assumptions that frequently impractical. In this course, we don't worry about these sophisticated methods; instead, we only try to balance at all. In the next section we learn one of the simplest methodologies to deal with this issue.

## 4.2 Action-value methods

We start with the definition of these methods.

**Definition 4.4.** *Action-value methods estimate the values of actions and use the estimates to make action selection decisions.*

The simplest action-value method is averaging the rewards received when taking each action. Mathematically, we use the estimate

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \mathbb{1}_{\{A_i=a\}}}{\sum_{i=1}^{t-1} \mathbb{1}_{\{A_i=a\}}} \tag{2.1}$$

where $\mathbb{1}_{\{A_i=a\}}$ is the indicator function of taking action $a$ at time period $i$, that is,

$$\mathbb{1}_{\{A_i=a\}} = \begin{cases} 1 & \text{, if at time step } i \text{ we chose action } a \\ 0 & \text{, otherwise} \end{cases}$$

Then, the estimate $Q_t(a)$ is the sample average of the rewards received when action $a$ has been chosen because it adds up the total rewards received when we took action $a$, and divides by the number of times we used action $a$.

Recall that we defined the greedy action as the action with the highest estimated value. Mathematically, we can define the greedy action as

$$A_t = \arg\max_a Q_t(a)$$

where arg max represents the maximizer, that is, the value of the argument that maximizes the function.

Selecting the greedy action always exploits current knowledge to maximize reward. However, we already know that this behavior may not be optimal. In this section we propose a simple way to balance exploration and exploitation, with the following policy.

**Definition 4.5.** *An $\epsilon$-greedy policy chooses a greedy action with probability $1-\epsilon$ and a randomly selected action with probability $\epsilon$ in each time step.*

An advantage of the $\epsilon$-greedy policy is that, no matter how small is $\epsilon$, in the long run all the actions will be visited infinitely many times. Hence, under $\epsilon$-greedy, we have

$$\lim_{t\to\infty} Q_t(a) = q_*(a) \qquad \forall a$$

**Example 4.1** (Example 2.1 from the textbook)**.** *In $\epsilon$-greedy action selection, for the case of 2 actions and $\epsilon = 0.5$, what is the probability that the greedy action is selected?*

**Solution.** Let's call the actions $a_1$, $a_2$ and suppose that $a_1$ is the greedy action at time step $t$, that is,

$$Q_t(a_1) > Q_t(a_2)$$

We select action $a_1$ in two cases:

- We decide to select the greedy action, which happens with probability $1 - \epsilon = 0.5$
  or
- We decide to select a random action, and we select action $a_1$. We select a random action with probability $\epsilon$, and the selected action is $a_1$ with probability 0.5. Hence, in this case we select action $a_1$ with probability 0.25

Therefore, we select the greedy action with probability $0.5 + 0.25 = 0.75$. □

**Example 4.2.** *Consider a 4-armed bandit, that is, a multi-armed bandit as described above with $k = 4$. We apply an $\epsilon$-greedy action selection, sample-average action-value estimates, and initial estimates*

$$Q_0(a) = 0 \qquad \forall a \in \{1, 2, 3, 4\}$$

*Suppose the initial sequence of actions and rewards is*

| *Iteration* | *Action* | *Reward* |
|---|---|---|
| $t = 1$ | $A_1 = 1$ | $R_1 = -1$ |
| $t = 2$ | $A_2 = 2$ | $R_2 = 1$ |
| $t = 3$ | $A_3 = 2$ | $R_3 = -2$ |
| $t = 4$ | $A_4 = 2$ | $R_4 = 2$ |
| $t = 5$ | $A_5 = 3$ | $R_5 = 0$ |

*On some of these steps the $\epsilon$ case may have occurred, causing an action to be selected at random. On which steps did this definitely occur?*

**Solution.** First, observe that the "$\epsilon$ case" refers to an exploration step. We can be sure that this occured when the action taken is not one of the greedy actions. Let's evaluate each iteration.

- Step $t = 0$: All the estimated action values are $Q_0(a) = 0$, so all the actions are greedy.
- Step $t = 1$: We selected $A_1 = 1$, which is one of the greedy actions. Then, this is not necessarily an exploration step.

  After taking action $A_1 = 1$ we can update the action-value estimate of $a = 1$. Using the sample average estimation, we obtain

  $$Q_1(1) = \frac{R_1}{1} = \frac{-1}{1} = -1$$

  We don't modify the action-value estimate of the remaining actions. Then, we have

  $$Q_1(a) = 0 \qquad \forall a \in \{2, 3, 4\}$$

  Therefore, the greedy actions are $a = 2, 3, 4$.
- Step $t = 2$: The action taken is $A_2 = 2$, which is one of the greedy actions. Hence, we don't know if this step was exploration or exploitation.

  After taking action $A_2$, we can update our action-value estimate of $a = 2$ as follows:

  $$Q_2(2) = \frac{R_2}{1} = \frac{1}{1} = 1$$

  The remaining actions maintain their estimated values from step 1, that is, we have

  $$Q_2(1) = -1, \qquad \text{and} \qquad Q_2(3) = Q_2(4) = 0$$

  Therefore, the greedy action is $a = 2$.

- Step $t = 3$: Here we select $A_3 = 2$, which is the greedy action. Then, we cannot be sure whether this is an exploration step or not.

  The updated action-value estimates are:

  $$Q_3(2) = \frac{R_2 + R_3}{2} = \frac{1 + (-2)}{2} = -\frac{1}{2}$$

  and the other actions maintain their action-values:

  $$Q_3(1) = -1, \qquad \text{and} \qquad Q_3(3) = Q_3(4) = 0$$

  Then, the new greedy actions are $a = 3, 4$.

- Step $t = 4$: Again, we select $A_4 = 2$. However, this time $a = 2$ is not a greedy action. Hence this must be an exploration step.

  We update the action-value estimates as follows:

  $$Q_4(2) = \frac{R_2 + R_3 + R_4}{3} = \frac{1 + (-2) + 2}{3} = \frac{1}{3}$$

  and

  $$Q_4(1) = -1, \qquad \text{and} \qquad Q_4(3) = Q_4(4) = 0$$

  This step makes $a = 2$ the greedy action again.

- Step $t = 5$: We select $A_5 = 3$, which is not a greedy action. Then, this is another exploration step.

  We update the action-value estimates as follows:

  $$Q_5(3) = \frac{R_5}{1} = \frac{0}{2} = 0$$

  and

  $$Q_5(1) = -1, \qquad Q_5(2) = \frac{1}{3}, \qquad Q_5(4) = 0$$

  Again, $a = 2$ is the greedy action.

□

## 4.3 A 10-armed bandit example

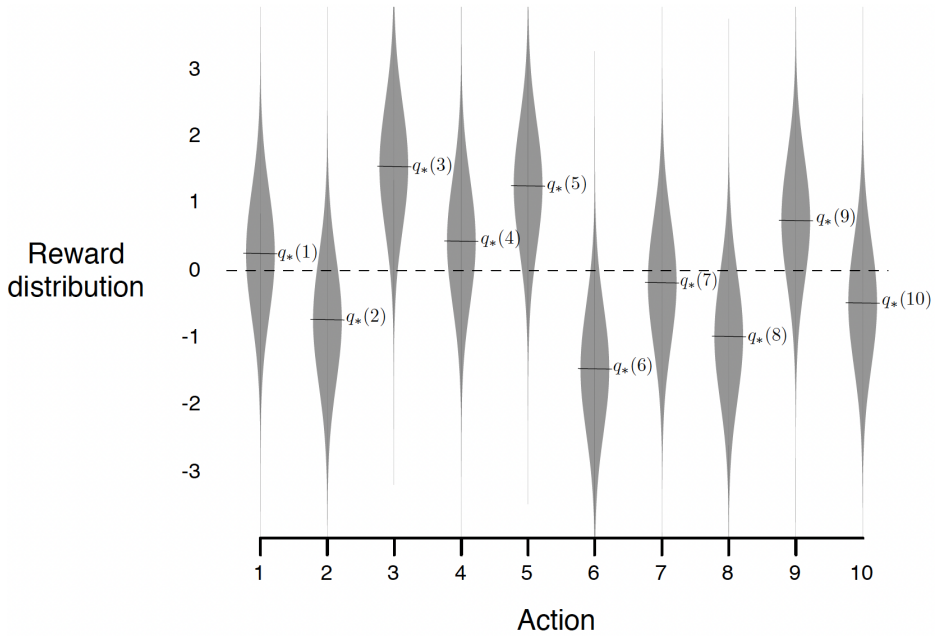
In this section we study a simulated example of the $k$-armed bandit problem we studied in the previous section. Our goal is to numerically study the effects of exploration in maximizing the reward.

We consider $k = 10$, and will study the rewards after 2000 randomly generated replicas of the bandit problem. In each replica,

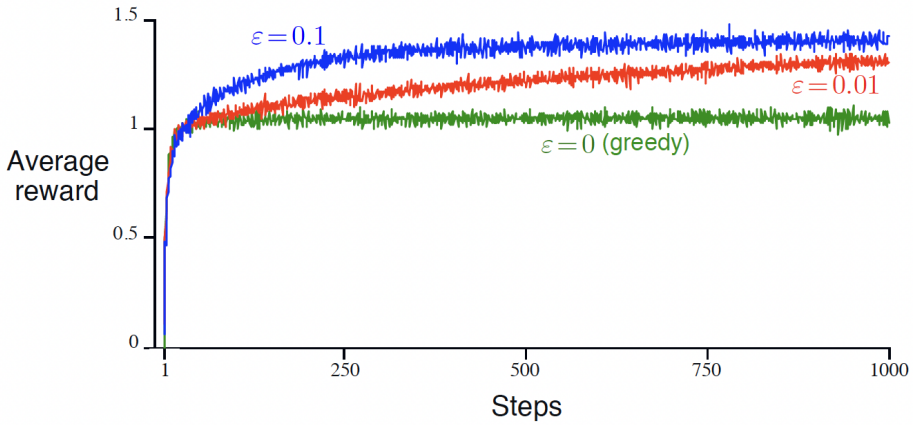
- The true action values $q_*(a)$ for $a \in \{1, 2, \ldots, 10\}$ are i.i.d. normally distributed with mean 0 and variance 1.
- In each step $t$, an action $A_t$ is taken and the reward $R_t$ received is normally distributed with mean $q_*(A_t)$ and variance 1.
- Action-value estimates are computed in each step using the running average.
- Each replica (or run) plays for $T = 1000$ time steps

Then, in each of the 2000 replicas, the rewards of each action are represented by the following picture[1]

[1] Figure 2.1 from Sutton and Barto's textbook



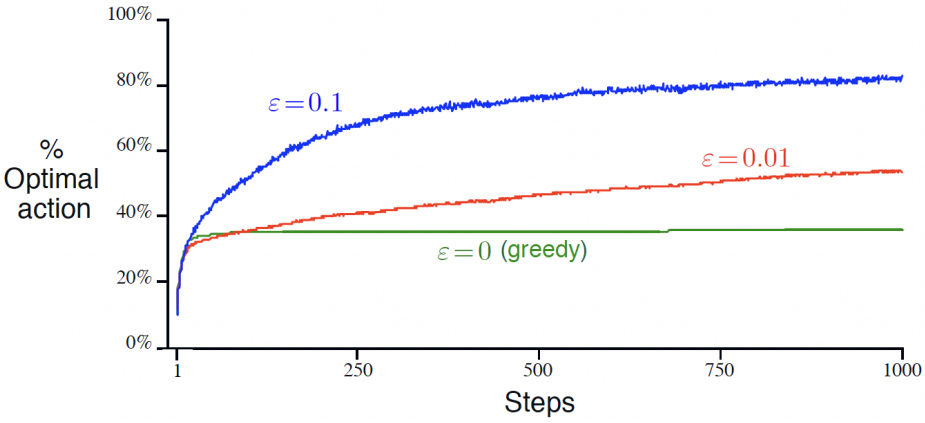

In the following figure[2] we show the average reward with respect to time steps of a greedy algorithm, vs. $\epsilon$-greedy with $\epsilon = 0.01$ and $\epsilon = 0.1$.




Observe that $\epsilon = 0.1$ obtains the highest average reward, and the greedy algorithm the lowest. However, in the first few steps, the greedy algorithm grows a bit faster. The greedy algorithm didn't do well in the long run because it often gets stuck in suboptimal actions.

In the following figure[3] we show the percentage of times where each of the policies found the optimal action.

[2] Upper graph of Figure 2.2 from Sutton and Barto's textbook
[3] Lower graph of Figure 2.2 from Sutton and Barto's textbook




Again, the greedy algorithm performed worse than the $\epsilon$-greedy algorithms, and $\epsilon = 0.1$ has the best performance.

The only case where the greedy algorithm outperforms $\epsilon$-greedy is when the rewards are deterministic and stationary. If they are deterministic, but change with time (as often happens in real applications), some exploration is needed to update the greedy actions.

On the other hand, the more variance of the rewards received per action, the more exploration is needed to find an optimal action.

## 4.4 Incremental Implementation

We've discussed that a good estimate of the action-values is averaging the rewards we receive each time we take a determined action. To compute this estimate we use Equation (2.1), where we add all the corresponding rewards and divide by the number of times we've taken the action. As $t$ increases, we need to use more rewards to compute these estimates and, hence, we need to be able to store all the historical rewards. Therefore, this estimates need increasing memory, and the computation time increases too. In this section, we learn how to compute these averages efficiently, with constant memory and constant per-time-step computation.

To simplify the notation, let's focus on a single action. Specifically, let $R_i$ denote the reward received the $i^{\text{th}}$ time we took action $i$, and $Q_n$ be the estimate of this action value after being selected $n-1$ times. Then, Equation (2.1) becomes

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

We'll manipulate the expression above to create a recursive formula. We obtain

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) && \text{(separating the "new" reward from the sum)} \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) && \text{(multiplying and dividing by } n-1 \text{ the sum)} \\
&= \frac{1}{n}\left(R_n + (n-1)Q_n\right) && \text{(using the definition of } Q_n\text{)} \\
&= Q_n + \frac{1}{n}\left(R_n - Q_n\right) && \text{(reorganizing terms)} \qquad (2.2)
\end{aligned}$$

Hence, we can compute $Q_{n+1}$ only using the value of $Q_n$ from the memory, and the new reward.
Equation (2.2) is of a form that frequently occurs in RL. The general form is

$$\text{New estimate} \leftarrow \text{Old estimate} + \text{Step size}\,(\text{Target} - \text{Old estimate})$$

The term (Target - Old estimate) is an error in the estimate. In this case, the target corresponds to the reward (the quantity we are trying to estimate).

## 4.5 Tracking a nonstationary problem

Computing the action-value estimates with a simple average makes sense when the rewards do not change over time, that is, in stationary problems. However, we often have cases where the rewards are a function of time. In such case, it is better to compute the action-value estimates giving a higher weight to most recent observed rewards. In this section we learn how.

As pointed at the end of the previous section, the term $\frac{1}{n}$ in Equation (2.2) is a step size. If instead of $\frac{1}{n}$ we consider a constant step size $\alpha \in (0, 1)$, we update the action-value estimates as follows:

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha(R_n - Q_n) \\ &= \alpha R_n + (1-\alpha)Q_n \end{aligned}$$

where the newest reward has weight $\alpha$, and the old estimate has weight $1-\alpha$. Now, we can recursively use the update of action-value estimates to write $Q_{n+1}$ in function of $\alpha$ and the historical rewards, as follows:

$$\begin{aligned} Q_{n+1} &= \alpha R_n + (1-\alpha)Q_n \\ &= \alpha R_n + (1-\alpha)(\alpha R_{n-1} + (1-\alpha)Q_{n-1}) \\ &= \alpha R_n + \alpha(1-\alpha)R_{n-1} + (1-\alpha)^2 Q_{n-1} \qquad \text{(reorganizing terms)} \\ &= \alpha R_n + \alpha(1-\alpha)R_{n-1} + (1-\alpha)^2(\alpha R_{n-2} + (1-\alpha)Q_{n-2}) \\ &= \alpha R_n + \alpha(1-\alpha)R_{n-1} + \alpha(1-\alpha)^2 R_{n-2} + (1-\alpha)^3 Q_{n-2} \end{aligned}$$

If we continue inductively working on the recursion, we will obtain the following expression:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \tag{2.3}$$

Since $\alpha$ is a parameter between 0 and 1, $(1-\alpha)^{n-i}$ increases with $i$, that is, the newest rewards receive more weight than the older rewards. Since the weight of each reward $R_i$ decays exponentially with the number of steps ahead we are currently visiting, the expression (2.3) is sometimes called exponential recency-weighted average.

Further,

$$(1-\alpha)^n + \alpha \sum_{i=1}^{n} (1-\alpha)^{n-i} = 1$$

Therefore, Equation (2.3) is a weighted average of the historic rewards.

In the rest of this chapter, we will learn three ways to incentive exploration (besides $\epsilon$-greedy).

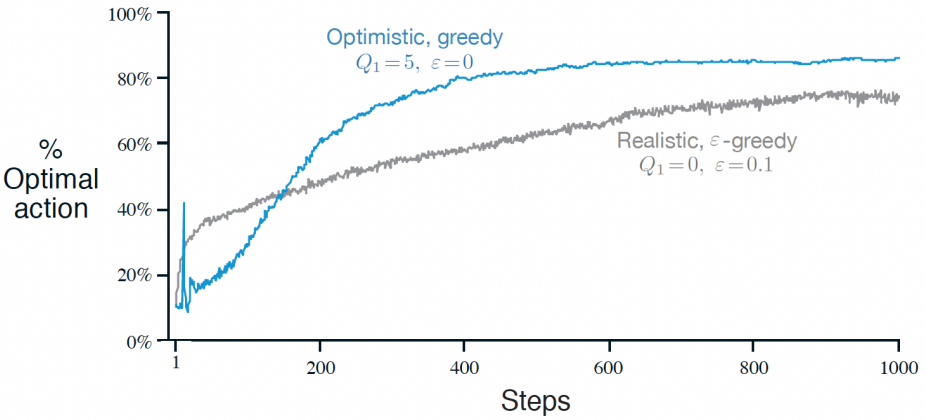
## 4.6 Optimistic Initial Values (OIV)

With the average (or weighted average) action-value methods, our very first estimate is a parameter of our choice. In the examples we've seen so far, we initialized all the action-value estimates at 0, that is, the mean of the true values. An alternative is to initialize all them at a value that will encourage exploration. In this case, we initialize them at a very high value and use the greedy action in each step.

The result is that all the actions are tried several times before their action-value estimates are close to their values $q_*(a)$. Then, the system does a considerable amount of exploration, even when taking a greedy action in each step.

As an example to show its effects, suppose we consider the example from Section 4.3 with initial values $Q_1(a) = 5$ for all $a \in \{1, \dots, 10\}$. Recall that the action values $q_*(a)$ are drawn from a standard normal distribution. Hence, $Q_1(a) = 5$ is five standard deviations above the mean. We show the result and compare it with $\epsilon$-greedy in the following figure[4]:

[4] Figure 2.3 from Sutton and Barto's textbook




Observe that $\epsilon$-greedy is better at finding the optimal action in early steps, but OIV becomes better in the long run. The result in the figure is an average over 2000 replications, so it is interesting that in early stages (close to step 1), OIV has a markes spike. A possible reason is that, while exploring, an optimal action was found and used several times. However, since all the actions are initialized at a high value, the optimal action stops being greedy when its estimated value improves. Then, for some time the system uses currently greedy actions, until the estimates improve and only the truly greedy actions are perceived as greedy.

A drawback of this methos is that it only encourages exploration in early steps. Then, if the problem is nonstationary, it won't find the optimal actions in the long run. Another drawback is that, when we initialize the action values, we need to have a notion of what is a high action value. However, such notion is not always clear.

## 4.7 Upper-Confidence-Bound Action Selection (UCB)

As pointed at the end of last section OIV only encourages exploration at the beginning of the time horizon. Hence, it may not be appropriate for nonstationary problems. In such cases, $\epsilon$-greedy is better. A drawback from $\epsilon$-greedy is that the exploration steps can choose any action, including the greedy action(s) and actions that have low probability of being optimal. However, exploration on actions that are potentially optimal would be a better use of resources (and time).

With UCB, in every step we choose actions according to their potential to being optimal while considering the uncertainties. Mathematically,

$$A_t \in \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right\},$$

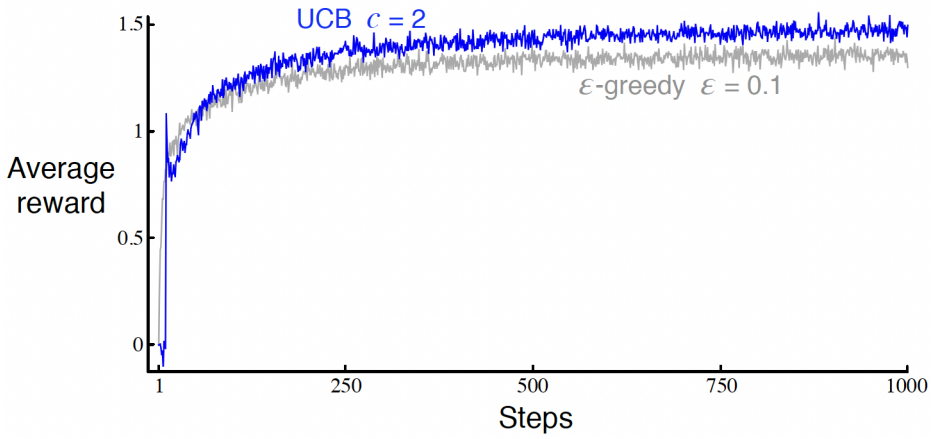
where $N_t(a)$ is the number of times action $a$ has been selected prior to time $t$, and $c$ is a parameter that controls the degree of exploration. Observe that UCV is greedy with respect to the objective function $Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$, which considers the best actions according to their estimated value $Q_t(a)$ and the uncertainty $c\sqrt{\frac{\ln t}{N_t(a)}}$. Then, by modifying the objective function, we do both: exploration and exploitation.

Intuitively, $\frac{\ln t}{N_t(a)}$ measures the uncertainty in the estimate of $q_*(a)$ because:

- Each time action $a$ is selected, $N_t(a)$ increases by 1 and, therefore, the uncertainty measure decreases
- If action $a$ is not selected, then $N_t(a)$ remains constant and $t$ increases. Then, the uncertainty increases.
- Since the numerator depends on $\ln t$, the uncertainty increases more for smaller values of $t$ than for large values of $t$.

Therefore, UCB selects with more frequency actions with high estimated value and high uncertainty. The figure[5] below shows UCB's performance in comparison to $\epsilon$-greedy in the 10-armed bandit example.

[5] Figure 2.4 from Sutton and Barto's textbook




We observe that, by selecting the right parameters, the performance of UCB and $\epsilon$-greedy is similar. However, UCB is more complicated and harder to generalize for nonstationary scenarios and more general problems. Further, the intuition behind the parameter $\epsilon$ is more intuitive than the intuition for $c$.

## 4.8 Grandient Bandit Algorithms

In this section we introduce a different way to decide which actions are preferable to others. In $\epsilon$-greedy and OIV, we preferred actions that have high estimated values. In UCB, we favor actions with high estimated value and high uncertainty. In this section we introduce the notion of preference function. Let $H_t(a)$ denote the preference for action $a$ at time step $t$.

Based on the preference for each action at each time slot, GBA selects an action $a$ with probability $\pi_t(a)$, defined as

$$\pi_t(a) = \mathbb{P}\left[A_t = a\right] = \frac{e^{H_t(a)}}{\sum_{i=1}^{k} e^{H_t(i)}}$$

Then, actions with higher preference get higher probability of being selected. Observe that this method naturally encourages exploration because we are only providing the probability of choosing an action in each time step.

To encourage exploitation, we need to make sure that the preference for actions with high estimated values is large. In this case, we compute the preference at each time step recursively. Let $\overline{R}_t$ be the average reward received so far with all the actions, that is,

$$\overline{R}_t = \frac{1}{t}\sum_{i=0}^{t-1} R_i$$

Then, we compute the preference number by

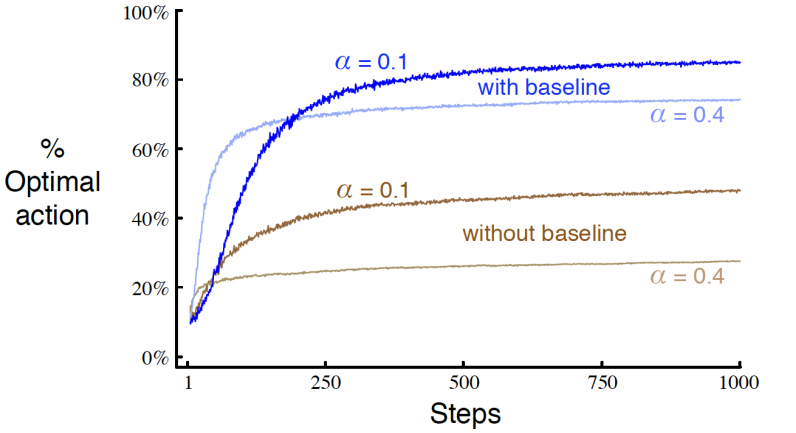
$$H_1(a) = 1 \qquad \forall a \in \{1, \dots, k\}$$

$$H_{t+1}(a) = \begin{cases} H_t(a) + \alpha(R_t - \overline{R}_t)(1 - \pi_t(a)) & \text{, if } A_t = a \\ H_t(a) - \alpha(R_t - \overline{R}_t)\pi_t(a) & \text{, if } A_t \neq a \end{cases} \qquad \forall t \geq 1,\ \forall a \in \{1, \dots, k\}$$

where $\alpha > 0$ is a step-size parameter. Observe that $\overline{R}_t$ acts as a baseline to increase or decrease the preference of each action: If we choose action $a$ and the reward we receive is larger than the average reward so far, we increase the preference of action $a$. If the reward is smaller than the average reward, we decrease the preference for action $a$. For other actions, we do the opposite.

An alternative is to update the preferences without using $\overline{R}_t$ as a baseline. However, this modification considerably harms the performance of the algorithm. In the following figure[6] we compare the performance of the algorithm for two values of $\alpha$, with and without baseline, in the 10-armed bandit example.

[6] Figure 2.5 from Sutton and Barto's textbook




Observe that the baseline makes a huge difference in the percentage of times the algorithm finds an optimal action. Additionally, larger values of $\alpha$ are more effective in the short term, but are outperformed by smaller values of $\alpha$ in the long run.

# 5 Markov Decision Processes

In this chapter we study a general model which is often used in RL. In the previous section, we studied multi-armed bandits, where the agent takes an action in each step and the action brings an immediate reward. This is called an *evaluative* problem because all we do is evaluating how good is an action.

More general RL problems also involve an *associative* aspect, where each action is evaluated according to the state. Then, in different states we might have a different set of available actions, and an optimal action at a given state may not give a high reward at a different state. Then, we need to *associate* good actions to states.

Markov Decision Processes (MDP) are a formalization of sequential decision making, where the actions influence the possible future states, the immediate reward, and the future rewards. MDPs are sometimes considered an idealized form of the RL problem.

## 5.1 Key definitions and notation

Let's start with some definitions.

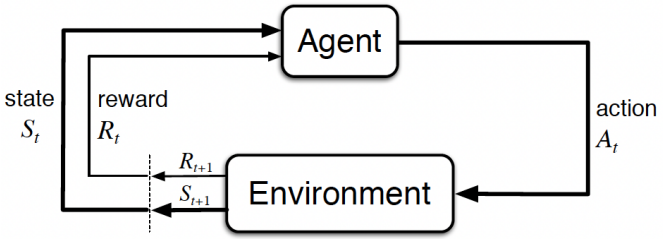
**Definition 5.1.**

*(i) The learner or decision maker is called the agent*

*(ii) The things that interact with the agent, comprising everything outside the agent, is called the environment*

The agent and the environment interact with each other in a sequence of discrete time steps $t = 1, 2, \ldots$. At each time step, the agent receives information about the environment's state $S_t \in \mathcal{S}$, and takes an action $A_t \in \mathcal{A}(S_t)$. Observe that the set of possible actions depends on the current state. One time step later, the agent receives a reward $R_{t+1} \in \mathcal{R}$ and finds itself in a new state $S_{t+1} \in \mathcal{S}$.

The following figure depicts this sequence




and the history of the agent's observations is

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$$

At time $t$, the action taken influences the reward received and the next state. Specifically, the transition probabilities are defined by

$$p(s', r|s, a) = \mathbb{P}\left[S_t = s', R_t = r \,|\, S_{t-1} = s, A_{t-1} = a\right] \qquad \forall s, s' \in \mathcal{S},\ a \in \mathcal{A}(s),\ r \in \mathcal{R}$$

In words, the probability of receiving a certain reward and transitioning to each state, depends on the current state and the action selected. If we link the function above with DTMC's, $p(s', r|s, a)$ represents the transition matrix. Indeed,

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1 \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

The function $p$ defined above completely determines the dynamics of the environment. Then, we must make sure that the state $S_t$ contains all the information that influences the future state of the agent. Then, we say that the state satisfies the **Markov property**.

With a slight abuse of notation, we denote

$$p(s'|s, a) = \mathbb{P}\left[S_t = s' | S_{t-1} = s, A_{t-1} = a\right] = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

We can compute the expected rewards in the next step given the current state an action in terms of the function $p$, as follows:

$$r(s,a) = \mathbb{E}\left[R_t \,|S_{t-1} = s, A_{t-1} = a\right] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s', r|s, a)$$

We may also be interested in the rewards given the transition to a state $s'$. With a slight abuse of notation, we use

$$r(s,a,s') = \mathbb{E}\left[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'\right] = \sum_{r\in\mathcal{R}} r\frac{p(s', r|s, a)}{p(s'|s, a)}$$

Determining the boundary between the agent and the environment can be challenging. In practice, this limit depends on the goals we are pursuing and the level of detail that we need to model. In general, it is easier to determine the states, actions and rewards to identify the decision maker and the environment signals.

**Example 5.1.** *Suppose that RL is used to control the motion of a robot arm in a repetitive pick-and-place task. The goal is to learn movements that are fast and smooth. Identify possible actions, states, and rewards.*

**Solution.**

- Possible actions: Voltage applied to each motor at each joint
- Possible states: Latest readings of joint angles and velocities
- Possible rewards: +1 for each object successfully picked up and placed. To encourage smoothness, give a negative reward for jerkiness in motion.

□

**Example 5.2.** *Consider the problem of driving. You could define your actions in terms of:*

- *Where your body meets the car: Accelerator, steering wheel, and brake, or*
- *Where the car meets the road: Tire torques, or*
- *Where your brain meets your body: Muscle twitches to control your limbs*
- *Very general task: Where to drive*

*What is the right level, the right place to draw the line between the agent and the environment? Why would you prefer one level or another one?*

**Solution.** The answer to all those questions is "it depends." Before defining an appropriate model we need to decide what is the decision we want to learn, and who is the decision maker. □

**Example 5.3.** *A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin. The robot runs on a rechargeable battery.*

*High level decisions about how to search for cans are made by a RL agent based on the current level of battery. If the battery level is high, the agent can decide to actively search for a can for a time step, or to remain stationary and wait for someone to bring a can. If the battery level is low, the agent might search, wait, or go to recharge. The rewards are zero most of the time, but become 1 if the robot secures a can, or -3 if the battery runs all the way down. Let $r_s$ and $r_w$ be the expected reward after a period of search and wait, respectively. We can safely assume that $r_s > r_w$.*

*The best way to find cans is to actively search for them, but this runs down the battery, whereas waiting does not. Whenever the robot is searching, there is a chance that its battery will become depleted. In this case, the robot must shut down and wait to be rescued. When the robot is rescued, the battery is then charged back to high. No cans can be collected if the battery is depleted, or if the robot is at the charging station.*

*A period of searching that begins with high battery, leaves the energy in high with probability $\alpha$, and low with probability $1-\alpha$. On the other hand, a period of searching that begins with with low energy, ends with low energy with probability $\beta$, and depleted with probability $1-\beta$.*

*Model this problem as an MDP, indicating the transition probabilities and the expected rewards.*

**Solution.** The state represents the level of battery of the robot. Then, the state space is $\mathcal{S} = \{L, H\}$, where $L$ represents low and $H$ represents high.

The possible actions are

$$\mathcal{A}(L) = \{R, S, W\}, \quad \mathcal{A}(H) = \{S, W\},$$

where $R$ represents recharge, $S$ represents search and $W$ represents wait.

In this case we don't have a distribution of the rewards. Then, we compute $p(s'|s, a)$ and $r(s, a, s')$ for each $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. We obtain

| $s$ | $a$ | $s'$ | $p(s'\|s,a)$ | $r(s,a,s')$ |
|---|---|---|---|---|
| $H$ | $S$ | $H$ | $\alpha$ | $r_s$ |
| $H$ | $S$ | $L$ | $1-\alpha$ | $r_s$ |
| $H$ | $W$ | $H$ | 1 | $r_w$ |
| $H$ | $W$ | $L$ | 0 | |
| $L$ | $S$ | $H$ | $1-\beta$ | -3 |
| $L$ | $S$ | $L$ | $\beta$ | $r_s$ |
| $L$ | $W$ | $L$ | 1 | $r_w$ |
| $L$ | $W$ | $H$ | 0 | |
| $L$ | $R$ | $H$ | 1 | 0 |
| $L$ | $R$ | $L$ | 0 | |

□

## 5.2 Returns and Episodes

In general, we are interested in maximizing the total reward received by the agent. However, the sum of all the future rewards grows to infinity unless the MDP has a finite number of steps. Hence, we instead maximize the present value of future rewards, which we call returns. Let $G_t$ be the returns from future steps. Then,

$$G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{2.4}$$

where $\gamma \in [0, 1]$ is the discount rate and $T$ is the time of termination. We allow $T = \infty$ and $\gamma = 1$, but not both at the same time. Otherwise, $G_t = \infty$. We additionally assume that the sequence of rewards is finite, that is, $R_\tau < \infty$ for all $\tau$.

If $\gamma = 0$ the agent is *myopic* because it only cares about the immediate reward, and as $\gamma$ approaches 1, the agent becomes more *farsighted*.

Observe that we can also write the returns recursively, as follows:

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{2.5}$$

The definition of $G_t$ above includes the cases where $T$ is finite, and where $T$ is infinity. When $T$ is finite, we usually run the learning process more than once. For example, in a game. However, most of the time we probabilistically analyze only one episode (or replica). Hence, without loss of generality, we work with the definition of returns provided in Equation (2.4).

Let's do an example to better understand the returns and how to compute them.

**Example 5.4.** *Suppose $\gamma = 0.5$, and the following sequence of rewards is received: $R_1 = -1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are $G_0, G_1, G_2, G_3, G_4, G_5$?*

**Solution.** Probably the easiest way to compute the returns is using the recursion from Equation (2.5). Observe that we need $G_{t+1}$ to compute $G_t$. Then, we compute them backwards. We obtain

$$\begin{aligned} G_5 &= R_6 = 0 \\ G_4 &= R_5 + \gamma G_5 = 2 + 0.5 * 0 = 2 \\ G_3 &= R_4 + \gamma G_4 = 3 + 0.5 * 2 = 4 \end{aligned}$$

$$G_2 = R_3 + \gamma G_3 = 6 + 0.5 * 4 = 8$$
$$G_1 = R_2 + \gamma G_2 = 2 + 0.5 * 8 = 6$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.5 * 6 = 2$$

□

## 5.3 Policies and Value Functions

We already learned that the value function represents how good is for the agent to be in a certain state, considering the available actions and the expected future rewards it will receive. In this section, we formalize this idea and provide an equation.

Recall that a policy is a mapping from states to actions, that indicates which actions we prefer in each state. A policy can be deterministic or probabilistic. We use

$$\pi(a|s) = \mathbb{P}\left[A_t = a \,|\, S_t = s\right]$$

**Example 5.5.** *If the current state is $S_t = s$, and actions are selected according to a stochastic policy $\pi$, then what is the expectation of $R_{t+1}$ in terms of $\pi$ and $p(s', r|s, a)$?*

**Solution.** We need to compute $\mathbb{E}\left[R_{t+1}|S_t = s\right]$. However, the return depends on the action selected, and the action selected depends on the policy. Hence, we condition. We obtain

$$\mathbb{E}\left[R_{t+1}|S_t = s\right] = \sum_{r\in\mathcal{R}}\sum_{s'\in\mathcal{S}}\sum_{a\in\mathcal{A}(s)} r\, p(s', r|s, a)\pi(a|s)$$

□

Our next goal is connecting the returns with the policies. Specifically, we need to know the expected returns under a specific policy $\pi$.

**Definition 5.2.**

*(i) The value function of a state $s$ under policy $\pi$ is denoted by $v_\pi(s)$ and is formally defined as*

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t|S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,\middle|\, S_t = s\right] \qquad \forall s \in \mathcal{S}$$

*(ii) The action-value function represents the value of taking action $a$ in state $s$ under policy $\pi$. It is denoted by $q_\pi(s, a)$ and formally defined as*

$$q_\pi(s, a) = \mathbb{E}_\pi\left[G_t|S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,\middle|\, S_t = s, A_t = a\right] \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Observe that

$$v_\pi(s) = \sum_{a\in\mathcal{A}(a)} q_\pi(s, a)\pi(a|s)$$

Solving a RL problem means finding a policy $\pi$ that maximizes $v_\pi(s)$. To find this solution, a classic method is solving a recursion called the **Bellman equation**. In the rest of this section we compute the Bellman equation. Then, in the next chapter, we will learn some methods to solve it.

Using the recursion from Equation (2.5) and the law of total probability, we obtain

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left[G_t \,|\, S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s\right] \\
&= \sum_{a,s',r} \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r\right] \mathbb{P}\left[A_t = a, S_{t+1} = s', R_{t+1} = r|S_t = s\right] \\
&= \sum_{a,s',r} \left(r + \gamma\mathbb{E}\left[G_{t+1} \,|\, S_{t+1} = s'\right]\right) \mathbb{P}\left[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\right] \mathbb{P}\left[A_t = a|S_t = s\right]
\end{aligned}$$

$$= \sum_{a,s',r} (r + \gamma v_\pi(s'))\, p(s', r|s, a)\pi(a|s)$$

We close the section with the definition of an optimal policy.

**Definition 5.3.** *A policy $\pi$ is better than or equal to a policy $\pi'$, denoted $\pi \geq \pi'$, if and only if*

$$v_\pi(s) \geq v_{\pi'}(s) \qquad \forall s \in \mathcal{S}$$

*A policy that is better than or equal to all other policies is called an optimal policy. All the optimal policies share the optimal state-value function, denoted by $v_*$ and defined as*

$$v_*(s) = \max_\pi v_\pi(s) \qquad \forall s \in \mathcal{S}$$

*Optimal policies also share the optimal action-value function, denoted by $q_*$ and defined as*

$$q_*(s, a) = \max_\pi q_\pi(s, a) \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Observe that, by definition, we have that the optimal Bellman equations of the value function are

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} [r + \gamma v_*(s')]\, p(s', r|s, a) \qquad \forall s \in \mathcal{S}$$

**Example 5.6.** *Write the Bellman equations for the recycling robot of Example 5.3 for a given policy $\pi(a|s)$, and the optimality Bellman equations.*

**Solution.** Recall that the state space is $\mathcal{S} = \{H, L\}$ with $H$ denoting high battery, and $L$ denoting low battery. The actions can be search (denoted $S$), wait ($W$) and recharge ($R$) and we can only go recharge in low battery state.

We first write the Bellman equations for each state $s \in \{H, L\}$. We start with state $H$. By definition, we have

$$v_\pi(H) = \sum_{a,s',r} (r + \gamma v_\pi(s'))\pi(a|H)p(s', r|H, a)$$

Observe that we don't know the specific rewards of each action-state pair. Instead, we know the expected value. Then, we reorganize the sums to make the expectations show up. We first open the sums depending on the action taken. We obtain:

$$\begin{aligned} v_\pi(H) &= \pi(S|H) \left[\sum_{s',r} (r + \gamma v_\pi(s'))p(s', r|H, S)\right] + \pi(W|H) \left[\sum_{s',r} (r + \gamma v_\pi(s'))p(s', r|H, W)\right] \\ &= \pi(S|H)\left[r_s + \gamma v_\pi(H)\alpha + \gamma v_\pi(L)(1 - \alpha)\right] + \pi(W|H)\left[r_w + \gamma v_\pi(H)\right] \end{aligned}$$

Now we write the Bellman equations for the state $L$. In this case, the rewards depend on the future state because if the battery dies out while searching, the rewards are -3. This case is representing by the future state $H$. We split the sum depending on the action taken. We obtain

$$\begin{aligned} v_\pi(L) &= \pi(S|L)\left[(-3 + \gamma v_\pi(H))p(H|L, S) + (r_s + \gamma v_\pi(L))p(L|L, S)\right] + \pi(W|L)(r_w + \gamma v_\pi(L)) + \pi(R|L)\gamma v_\pi(H) \\ &= \pi(S|L)\left[(-3 + \gamma v_\pi(H))(1 - \beta) + (r_s + \gamma v_\pi(L))\beta\right] + \pi(W|L)\left[r_w + \gamma v_\pi(L)\right] + \pi(R|L)\gamma v_\pi(H) \end{aligned}$$

Now we compute the optimality Bellman equations. In this case, instead of using the policy $\pi$ to determine the probability of taking each action in each state, we maximize over actions. Then, we obtain

$$\begin{aligned} v_*(H) &= \max\left\{r_s + \gamma v_*(H)\alpha + \gamma v_*(L)(1 - \alpha),\ r_w + \gamma v_*(H)\right\} \\ v_*(L) &= \max\left\{(-3 + \gamma v_*(H))(1 - \beta) + (r_s + \gamma v_*(L))\beta,\ r_w + \gamma v_*(L))\right\},\ \gamma v_*(H)\} \end{aligned}$$

□

Frequently, when we face an episodic MDP (that is, with finite $T$), we need to keep track of how time passes in the Bellman equations. In such case, we solve them using backward induction. There is a terminal reward $r_T(s, a)$ and the Bellman equations under policy $\pi$ become:

$$v_{T,\pi}(s) = r_T(s) \qquad \forall s \in \mathcal{S}$$

$$v_{t,\pi}(s) = \sum_{a,s',r} \left(r + \gamma v_{\pi,t+1}(s')\right) p(s', r|s, a)\pi(a|s) \qquad \forall s \in \mathcal{S}, t \in \{1, \ldots, T-1\}$$

Similarly, the optimality Bellman equations are:

$$v_{T,*}(s) = r_T(s) \qquad \forall s \in \mathcal{S}$$
$$v_{*,t}(s) = max_{a \in \mathcal{A}(s)} \sum_{s',r} \left[r + \gamma v_{*,t+1}(s')\right] p(s', r|s, a) \qquad \forall s \in \mathcal{S}, t \in \{1, \ldots, T-1\}$$

# 6 Dynamic Programming

In this section we learn some methods to solve finite MDPs with a perfect model, that is, MDPs where the sets of states, actions and rewards are finite, and where we know the function $p(s', r|s, a)$ for all $s, s' \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$. We start with a definition.

**Definition 6.1.** *Dynamic Programming (DP) is a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as an MDP.*

In other words, DP solves the Bellman equations to some MDPs. DP limits the number of problems we can solve in RL because we need to assume that we completely know the model and that we have enough computational power to solve the equations. However, it provides the foundation to understanding many algorithms to approximately solve RL problems.

The main idea of DP is using the value functions to organize and structure the search for good policies. In particular, we use the Bellman equations as update rules for improving the approximations of the desired value functions. Before we start with the first algorithm, recall the Bellman equations:

$$v_*(s) = \max_a \sum_{s',r} [r + \gamma v_*(s')]\, p(s', r|s, a) \qquad \forall s \in \mathcal{S} \tag{2.6}$$

$$q_*(s, a) = \sum_{s',r} \left[r + \gamma \max_{a'} q_*(s', a')\right] p(s', r|s, a) \qquad \forall s \in \mathcal{S},\ a \in \mathcal{A}(s) \tag{2.7}$$

## 6.1 Policy evaluation

The first step to find an optimal policy is being able to compute the value function $v_\pi(s)$ of every state $s \in \mathcal{S}$ for a specific policy $\pi$. Recall the Bellman equation for $v_\pi(s)$:

$$v_\pi(s) = \sum_{a,s',r} [r + \gamma v_\pi(s')]\, p(s', r|s, a)\pi(a|s) \qquad \forall s \in \mathcal{S}$$

The policy evaluation algorithm simply establishes that we should use the Bellman equations as an update rule and compute $v_\pi$ sequentially, that is, we set some arbitrary initial values for $v_0$ and

$$v_{k+1}(s) = \sum_{a,s',r} [r + \gamma v_k(s')]\, p(s', r|s, a)\pi(a|s) \qquad \forall s \in \mathcal{S},\ k \geq 1 \tag{2.8}$$

Observe that $v_k = v_\pi$ is a fixed point of the update rule. Further it can be proved that the sequence $\{v_k : k \in \mathbb{Z}_+\}$ converges to $v_\pi$ as $k \uparrow \infty$.

To code the sequential computation (2.8) we have two options. The first option is maintaining two arrays: one representing $v_k$ and the other one representing $v_{k+1}$. The second one is computed using (2.8). The second option is maintaining only one array and using the updated values of $v_{k+1}(s)$ immediately after computing them by overwriting the old values. In the last case, we can think of the updates as being done in sweep through the state space.

The following pseudocode uses the second approach and tests the quantity

$$\max_{s \in \mathcal{S}} |v_{k+1}(s) - v_k(s)|$$

to determine if $v_k \approx v_{k+1}$, that is, to determine if a solution close to $v_\pi$ has been computed.

**Algorithm 6.1** (Policy Evaluation)**.**

*0.* ***Inputs:***

- *Policy to be evaluated:* $\pi$
- *Accuracy parameter:* $\theta > 0$

*1.* ***Initialization:*** *Provide an arbitrary value for* $V(s)$ *for all* $s \in \mathcal{S}$*.*

*2.* ***Loop:*** *Do While* $\Delta > \theta$*:*

*2.1. Set* $\Delta \leftarrow 0$.

*2.2. For each* $s \in \mathcal{S}$ *do:*

*2.2.1.* $v \leftarrow V(s)$

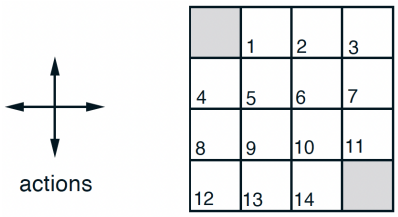
*2.2.2.* $V(s) \leftarrow \sum_{a,s',r} [r + \gamma V(s')]\, \pi(a|s) p(s', r|s, a)$

*2.2.3.* $\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$

3. ***Output:*** *Value function* $V(s)$ *for all* $s \in \mathcal{S}$.

Let's see a quick example.

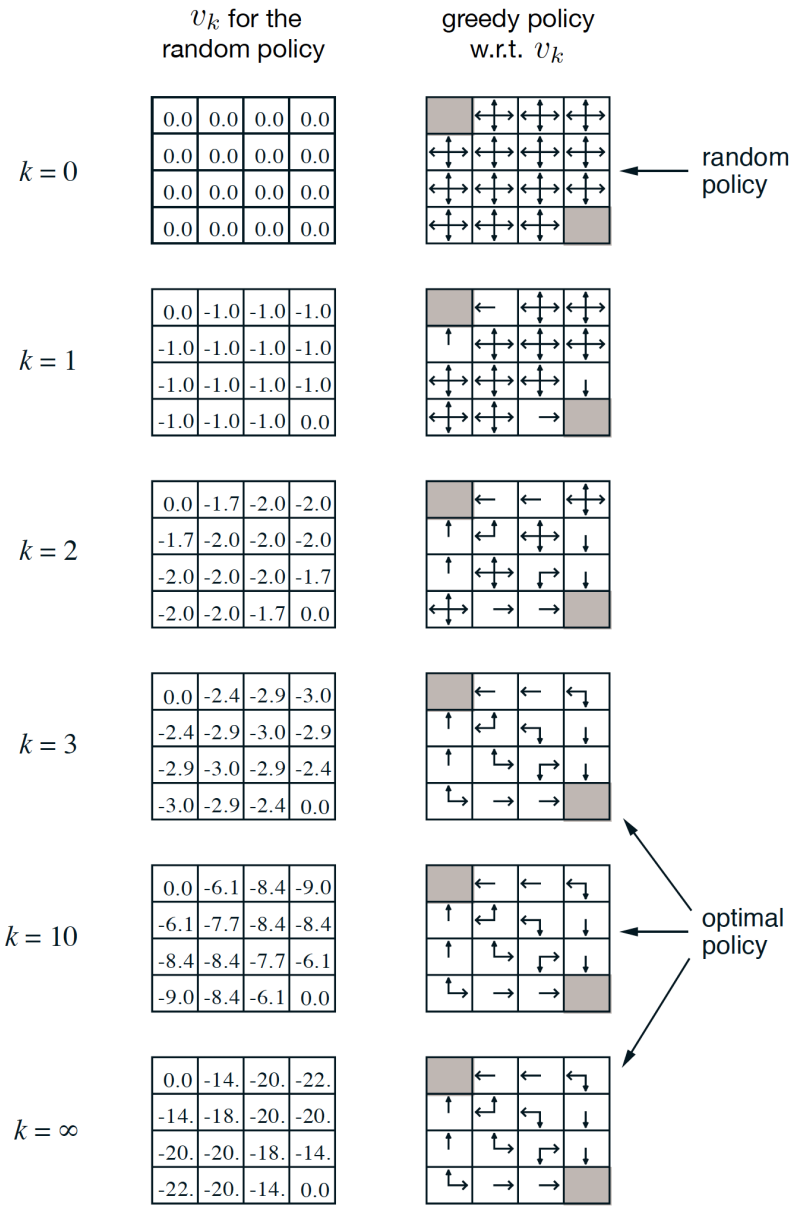
**Example 6.1.** *Consider the* $4 \times 4$ *grid shown below.*




*The shaded squares represent the terminal state (that is, the end of the game) and the numbered squares represent the states* $\mathcal{S}$ *where an action can be made. The possible actions in each of the numbered states are up, down, left or right and they result in the agent moving one square in the corresponding direction. The only exceptions are the actions that would take the agent outside the grid, and in these cases, the agent does not move. For example, if* $S_t = 4$ *and we take the action "up" we go to the terminal state, and if we take the action "left," then* $S_{t+1} = 4$.

*The reward for reaching a numbered state is -1, and a terminal state, 0.*

*The following picture shows the result of policy evaluation of the random policy for different values of* $k$*, and the greedy policy with respect to the estimation of* $v_\pi$ *for the given* $k$*'s.*




*Observe that, for $k = \infty$, the values of $v_\pi(s)$ represents the expected number of steps until the agent reaches a terminal state.*

## 6.2 Policy Improvement

The purpose of evaluating the policies in the previous section is being able to compare two policies $\pi$ and $\pi'$. In this section we learn how to compute a policy $\pi'$ that is better than a given policy $\pi$. Specifically, given a policy $\pi$ we would like to know whether or not for some state $s$ we should consider an action $a \neq \pi(s)$.

To do that, we consider selecting action $a$ in the current state $s$ and following the existing policy $\pi$ thereafter. Specifically, we evaluate

$$q_\pi(s, a) = \sum_{s', r} [r + \gamma v_\pi(s')]\, p(s', r|s, a)$$

and compare it with $v_\pi(s)$. If $q_\pi(s, a) \geq v_\pi(s)$, then taking action $a$ in state $s$ is better than taking action $\pi(s)$. Then, we expect that action $a$ is always better than action $\pi(s)$. Indeed, this is a special case of the following theorem.

**Theorem 6.1** (Policy improvement theorem)**.** *Let $\pi$ and $\pi'$ be any pair of deterministic policies such that for all $s \in \mathcal{S}$*

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{2.9}$$

*Then, the policy $\pi'$ must be as good as or better than $\pi$. That is, for all $s \in \mathcal{S}$,*

$$v_{\pi'}(s) \geq v_\pi(s) \tag{2.10}$$

*Moreover, if there is a strict inequality of (2.9) for some $s$, then there must be a strict inequality of (2.10) for the same state $s$.*

The policy improvement algorithm establishes that we should choose the action that appears best according to $q_\pi(s, a)$ and we will always obtain a better policy. Specifically, we consider the greedy policy $\pi'$ defined by

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s, a) \\ &= \arg\max_a \sum_{s',r} [r + \gamma v_\pi(s')]\, p(s', r|s, a)\end{aligned}$$

With this policy improvement methodology we can iteratively improve our policy $\pi$ until we obtain $v_\pi = v_{\pi'}$. In such case, the improvement step becomes the Bellman equation and, hence, we can assure that we found an optimal policy.

All the ideas in this section were developed for deterministic policies, but can be easily extended to stochastic policies. We won't do the details here.

## 6.3 Policy iteration

We now know how to evaluate the quality of a policy, and how to improve upon it. Then, if we put these two steps together, we have an algorithm that can find an optimal policy. Specifically, we can obtain a sequence of monotonically improving policies and value functions as follows:

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

This process is called policy iteration, and we provide a pseudocode below.

**Algorithm 6.2** (Policy Iteration)**.**

*0.* ***Input:*** *Parameter $\theta > 0$*

*1.* ***Initialization:*** *Provide an arbitrary value for $V(s)$ and $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$*

*2.* ***Policy Evaluation:*** *Do While $\Delta > \theta$:*

*2.1. Set $\Delta \leftarrow 0$*

*2.2. Loop for all $s \in \mathcal{S}$:*

*2.2.1. $v \leftarrow V(s)$*

*2.2.2. $V(s) \leftarrow \sum_{s',r,a} [r + \gamma V(s')]\, \pi(a|s) p(s', r|s, a)$*

*2.2.3. $\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$*

*3.* ***Policy Improvement:***

*3.1. Policy-Stable $\leftarrow$ True*

*3.2. Loop for all $s \in \mathcal{S}$:*

*3.2.1. Old-Action $\leftarrow \pi(s)$*

*3.2.2. $\pi(s) \leftarrow \arg\max_a \sum_{s',r} [r + \gamma V(s')]\, p(s', r|s, a)$*

*3.2.3. If Old-Action $\neq \pi(s)$, then Policy-Stable $\leftarrow$ False.*

*3.3. If Policy-Stable, then STOP. Return $v_* \approx V$ and $\pi^* = \pi$. Otherwise, go back to step 2.*

4. ***Output:*** *Policy $\pi$ approximately optimal.*

Let's do an example.

**Example 6.2.** *Consider a model with $\mathcal{S} = \{s_1, s_2, s_3\}$, $\mathcal{A}(s_1) = \{a_{11}, a_{12}\}$, $\mathcal{A}(s_2) = \{a_{21}, a_{22}\}$, and $\mathcal{A}(s_3) = \{a_{31}\}$. The rewards are*

$$\begin{aligned} &r(s_1, a_{11}) = r(s_1, a_{12}) = 0 \\ &r(s_2, a_{21}) = 3, \quad r(s_2, a_{22}) = 4 \\ &r(s_3, a_{31}) = 4 \end{aligned}$$

*and the transition probabilities are*

$$\begin{aligned} &p(s_1|s_1, a_{11}) = p(s_2|s_1, a_{11}) = \frac{1}{2} \\ &p(s_1|s_1, a_{12}) = \frac{2}{3}, \quad p(s_3|s_1, a_{12}) = \frac{1}{3} \\ &p(s_1|s_2, a_{21}) = 1 \\ &p(s_3|s_2, a_{22}) = 1 \\ &p(s_1|s_3, a_{31}) = 1 \end{aligned}$$

*Suppose the objective is to maximize the infinite horizon discounted reward with discount factor $\gamma = 0.8$. Perform one iteration of the policy iteration.*

**Solution.** Before starting policy iteration, observe that the rewards are deterministic. Then, we use the function $p(s'|s, a)$ instead of $p(s', r|s, a)$.

To run an iteration of policy iteration, we start with the random policy $\pi_0$, that is:

$$\begin{aligned} &\pi_0(a_{1j}|s_1) = \frac{1}{2} \qquad \forall j \in \{1, 2\} \\ &\pi_0(a_{2j}|s_2) = \frac{1}{2} \qquad \forall j \in \{1, 2\} \\ &\pi_0(a_{31}|s_3) = 1 \end{aligned}$$

We first go through the policy evaluation step, that is, we compute $v_\pi(s_j)$ for $j \in \{1, 2, 3\}$ and we use the Bellman equations. For state $s_1$, we obtain:

$$\begin{aligned} v_{\pi_0}(s_1) &= \sum_{j=1}^{3} \sum_{i=1}^{2} \left[ r(s_1, a_{1i}) + \gamma v_{\pi_0}(s_j) \right] \pi(a_{1i}|s_1) p(s_j|s_1, a_{1i}) \\ &= \sum_{j=1}^{3} \sum_{i=1}^{2} \gamma v_{\pi_0}(s_j) \pi(a_{1i}|s_1) p(s_j|s_1, a_{1i}) \qquad \text{(since } r(s_1, a_{11}) = r(s_1, a_{12}) = 0\text{)} \\ &= \sum_{j=1}^{3} \gamma v_{\pi_0}(s_j) \left[ \pi_0(a_{11}|s_1) p(s_j|s_1, a_{11}) + \pi_0(a_{12}|s_1) p(s_j|s_1, a_{12}) \right] \\ &= \sum_{j=1}^{3} \frac{\gamma}{2} v_{\pi_0}(s_j) [p(s_j|s_1, a_{11}) + p(s_j|s_1, a_{12})] \qquad \text{(by definition of } \pi_0\text{)} \\ &= \frac{\gamma}{2} \left[ v_{\pi_0}(s_1) \left( \frac{1}{2} + \frac{2}{3} \right) + v_{\pi_0}(s_2) \left( \frac{1}{2} \right) + v_{\pi_0}(s_3) \left( \frac{1}{3} \right) \right] \end{aligned}$$

Reorganizing terms, we obtain the following Bellman equation for $s_1$:

$$v_{\pi_0}(s_1) = \frac{7\gamma}{12} v_{\pi_0}(s_1) + \frac{\gamma}{4} v_{\pi_0}(s_2) + \frac{\gamma}{6} v_{\pi_0}(s_3) \tag{2.11}$$

Following a similar approach, we obtain the following Bellman equations for $s_2$ and $s_3$:

$$v_{\pi_0}(s_2) = (3 + \gamma v_{\pi_0}(s_1)) \left(\frac{1}{2}\right) + (4 + \gamma v_{\pi_0}(s_3)) \left(\frac{1}{2}\right) \tag{2.12}$$

$$v_{\pi_0}(s_3) = 4 + \gamma v_{\pi_0}(s_1) \tag{2.13}$$

With $\gamma = 0.8$, we solve the system of equations (2.11), (2.12), (2.13). We can use policy evaluation to solve it, but in this case we solve it using software for simplicity. We obtain

$$v_{\pi_0}(s_1) = \frac{1585}{68}, \quad v_{\pi_0}(s_2) = \frac{1600}{68}, \quad v_{\pi_0}(s_3) = \frac{1540}{68}$$

We now go through the policy improvement step. For state $s_1$ we have:

$$\begin{aligned}
\pi_1(s_1) &= \arg\max_a \left\{ \begin{array}{ll} a = a_{11}: & [r(s_1, a_{11}) + \gamma v_{\pi_0}(s_1)]\, p(s_1|s_1, a_{11}) + [r(s_1, a_{11}) + \gamma v_{\pi_0}(s_2)]\, p(s_2|s_1, a_{11}) \,, \\ a = a_{12}: & [r(s_1, a_{12}) + \gamma v_{\pi_0}(s_1)]\, p(s_1|s_1, a_{12}) + [r(s_1, a_{12}) + \gamma v_{\pi_0}(s_2)]\, p(s_2|s_1, a_{12}) \end{array} \right\} \\
&= \arg\max_a \left\{ \begin{array}{ll} a = a_{11}: & 0.8 * \dfrac{1585}{68} * \dfrac{1}{2} + 0.8 * \dfrac{1600}{68} * \dfrac{1}{2} \,, \\ a = a_{12}: & 0.8 * \dfrac{1585}{68} * \dfrac{2}{3} + 0.8 * \dfrac{1540}{68} * \dfrac{1}{3} \end{array} \right\} \\
&= \arg\max_a \{a = a_{11} : 18.73,\ a = a_{12} : 18.47\} \\
&= a_{11}
\end{aligned}$$

Similarly, for state $s_2$ we obtain

$$\begin{aligned}
\pi_1(s_2) &= \arg\max_a \left\{ \begin{array}{ll} a = a_{21}: & [r(s_2, a_{21}) + \gamma v_{\pi_0}(s_1)] p(s_1|s_2, a_{21}) \,, \\ a = a_{22}: & [r(s_2, a_{22}) + \gamma v_{\pi_0}(s_3)] p(s_3|s_2, a_{22}) \end{array} \right\} \\
&= \arg\max_a \left\{ \begin{array}{ll} a = a_{21}: & \left(3 + 0.8 * \dfrac{1585}{68}\right) \,, \\ a = a_{22}: & \left(4 + 0.8 * \dfrac{1540}{68}\right) \end{array} \right\} \\
&= \arg\max_a \{a = a_{21} : 21.65,\ a = a_{22} : 22.12\} \\
&= a_{22}
\end{aligned}$$

Finally, in state $s_3$ there is only one action, so $\pi_1(s_3) = a_{31}$. Hence, the improved policy is

$$\pi_1(s_1) = a_{11}, \quad \pi_1(s_2) = a_{22}, \quad \pi_1(s_3) = a_{31}$$

□

## 6.4 Value iteration

A drawback of policy iteration is that we need to do policy evaluation in every step, and this requires multiple sweeps through the entire state set. Then, for large state spaces, we'd be spending a considerable amount of time updating our estimates of the value function.

An alternative approach is stopping policy evaluation after just one sweep, that is, only one update of each state. This is exactly what value iteration does. More formally, we iteratively compute the optimal value function by turning the optimal Bellman equations into an update rule, as follows. Take an arbitrary value of $v_0$ and then

$$v_{k+1}(s) = \max_a \sum_{s', r} (r + \gamma v_k(s'))\, p(s', r|s, a) \qquad \forall s \in \mathcal{S}$$

The termination rule is that if $v_k \approx v_{k+1}$, then $v_k$ approximately solves the optimal Bellman equations. Using similar ideas as in policy evaluation, we obtain the following algorithm.

**Algorithm 6.3** (Value iteration)**.**

*0.* ***Input:*** *Parameter* $\theta > 0$

*1.* ***Initialization:*** *Arbitrary values of* $V(s)$ *for* $s \in \mathcal{S}$

*2.* ***Loop:*** *Do While* $\Delta > \theta$*:*

*2.1. Set* $\Delta \leftarrow 0$

*2.2. Loop for all* $s \in \mathcal{S}$*:*

*2.2.1.* $v \leftarrow V(s)$

*2.2.2.* $V(s) \leftarrow \max_a \sum_{s',r} (r + \gamma V(s'))\, p(s', r|s, a)$

*2.2.3.* $\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$

*3.* ***Output:*** *Policy* $\pi$ *approximately optimal, with*

$$\pi(s) = \arg\max_a \sum_{s',r} (r + \gamma V(s'))\, p(s', r|s, a)$$

Value iteration combines one sweep of policy evaluation and one sweep of policy improvement in a single sweep. However, it still converges very slowly. Frequently, we get faster convergence with multiple sweeps of policy evaluation between policy improvement sweeps.

## 6.5 Asynchronous Dynamic Programming

A major drawback of DP is the high number of operations needed in each iteration of either algorithm. Several applications have large state spaces and, then, computing the optimal policy using the algorithms described before becomes too slow. This is known as the *curse of dimensionality*.

A popular alternative is updating the values of states once in a while (making sure we update all of them). This method does not necessarily mean faster convergence, but at least we don't get stuck with too many computations for policies that we know are suboptimal.

# 7 Temporal-Difference (TD) Learning

As discussed earlier, a disadvantage of DP is that we need a perfect model, that is, we need to know the function $p(s', r|s, a)$ for all $s, s' \in \mathcal{S}$, $a \in \mathcal{A}(s)$ and $r \in \mathcal{R}$. In this chapter we learn the basics of Temporal-Difference (TD) learning, a method that can learn from experience without a model of the environment's dynamics.

Similarly to the previous chapter, we start with the prediction problem (evaluating the current policy) and, then, we focus on the control problem (finding the optimal policy).

## 7.1 TD prediction

When we think of using experience to solve for the value function $v_\pi(s)$, the most immediate way is using the actual returns and update $V(s)$ using the information of an entire episode. This approach is called Monte Carlo, and a simple update rule with step size $\alpha$ is

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

Observe the similarity of this update rule with the efficient implementation of the running average. In this case, we are using the value $V(S_t)$ to estimate the future returns starting from state $S_t$, that is, to estimate $G_t$. This is called constant-$\alpha$ Monte Carlo method. However, to update the value of $V(S_t)$ we need the actual value of $G_t$, which represents all the future discounted returns. Therefore, we need to wait until the episode ends before we can learn the value of $V(S_t)$ under the current policy.

With TD, instead, we only need to wait until the next time step. Specifically, we use the following update rule

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

where we only need to wait one time step to observe the reward $R_{t+1}$ and the future state $S_{t+1}$. This method is called $TD(0)$ and is a special case of $TD(\lambda)$. We will learn about $TD(\lambda)$ soon.

With this update, we obtain the following prediction algorithm:

**Algorithm 7.1** ($TD(0)$)**.**

*0.* ***Input:*** *Policy $\pi$ to be evaluated and algorithm parameter $\alpha > 0$*

*1.* ***Initialization:*** *Choose an arbitrary value of $V(s)$ for all $s \in \mathcal{S}$.*

*2.* ***Loop for each episode:***

*2.1. Initialize $S$*

*2.2. Loop for each step of episode:*

*2.2.1. $A \leftarrow \pi(S)$*

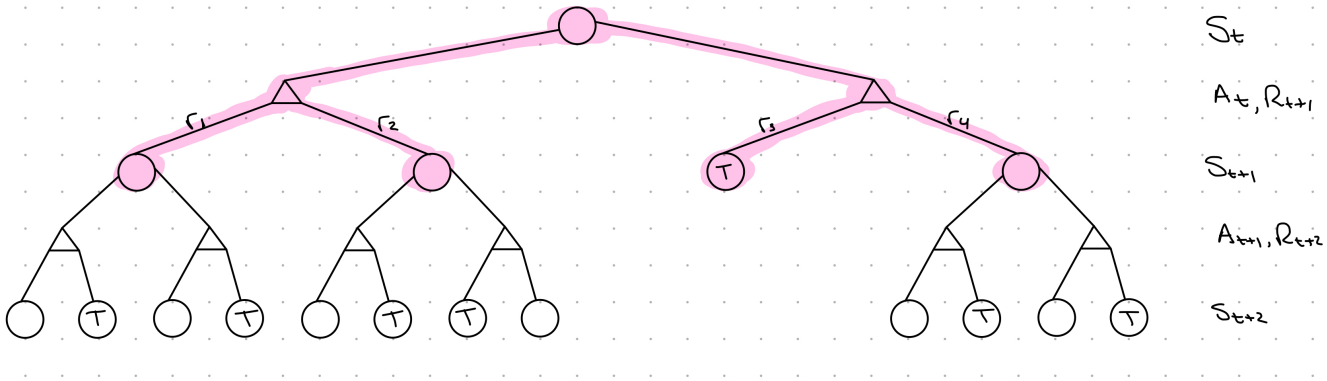
*2.2.2. Take action $A$, observe reward $R$ and next state $S'$*

*2.2.3. $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$*

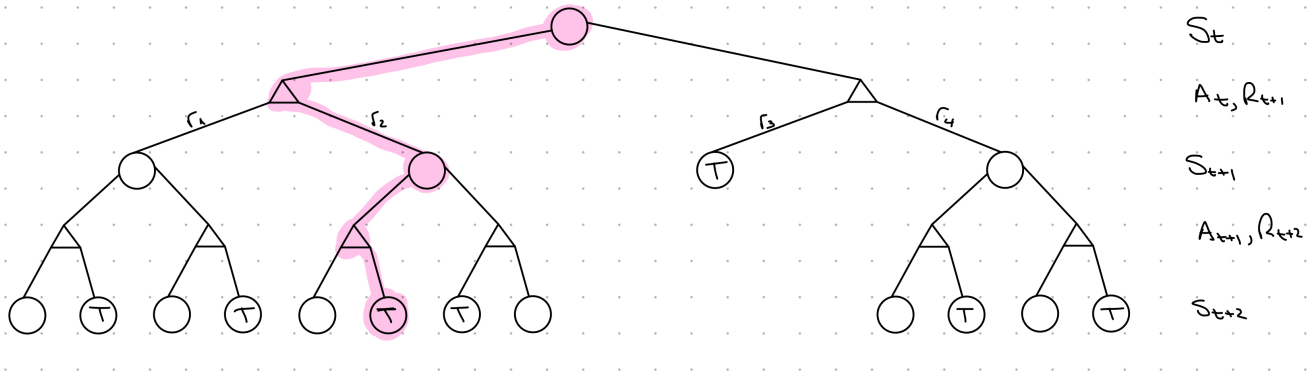
*2.2.4. $S \leftarrow S'$*

DP, Monte-Carlo and TD(0) are three methods with the same goal of computing a function $V(s)$ for all $s \in \mathcal{S}$ that approximately solves the Bellman equations, that is, such that $V(s) \approx v_\pi(s)$ for all $s \in \mathcal{S}$. However, they involve different targets and use different approaches to compute it.

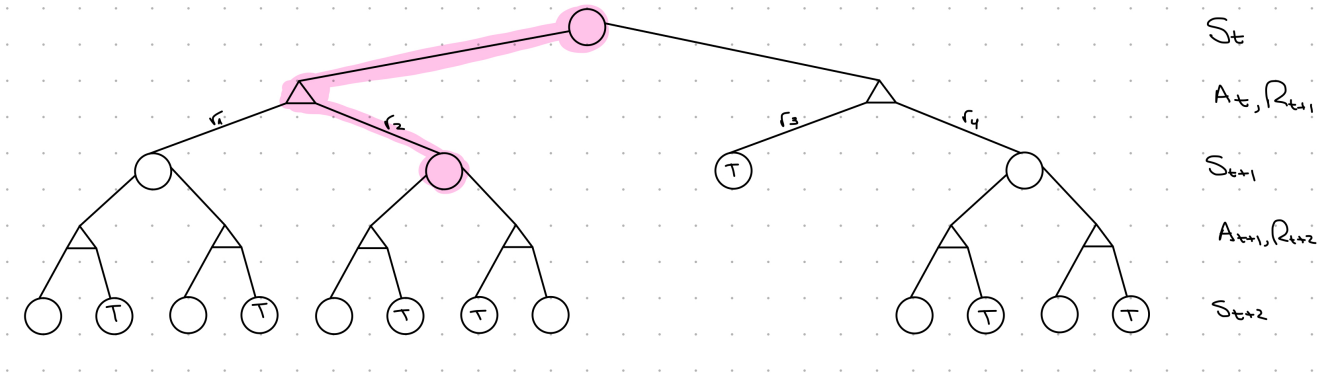
- In DP, the target is $\mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right]$ and we use $V(S_{t+1})$ to compute it. Observe that the future state $S_{t+1}$ is random and to compute the expected value in the target we need to consider all the possible cases and take the weighted average. In a simple diagram where circles represent states, triangles decisions, and circles with a $T$ represent terminal states (the end of an episode), DP can be represented by the area shaded in pink:




- In Monte Carlo simulation, the target is $\mathbb{E}_\pi [G_t | S_t = s]$. In this case we don't have a model and, hence, we cannot compute the expectation analytically. Instead, we sample the returns starting at each time $t$ and use each observation to estimate the expectation. In a diagram similar to the one above, Monte Carlo can be represented as follows:




- In TD(0) our target is $\mathbb{E}_\pi [R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$ and we compute it by sampling $R_{t+1}$. In some sense, TD(0) is between DP and Monte Carlo because it uses information about only one step ahead (as DP), and samples the values of interest in the expectation (as in Monte Carlo). A diagram for TD is as follows:




**Example 7.1.** *Each day as you drive home from school, you try to predict how long it will take you to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant.*

*Say on this Friday you are leaving from school at 6 o'clock, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. Traffic is often slower in the rain, so you re-estimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later, you*

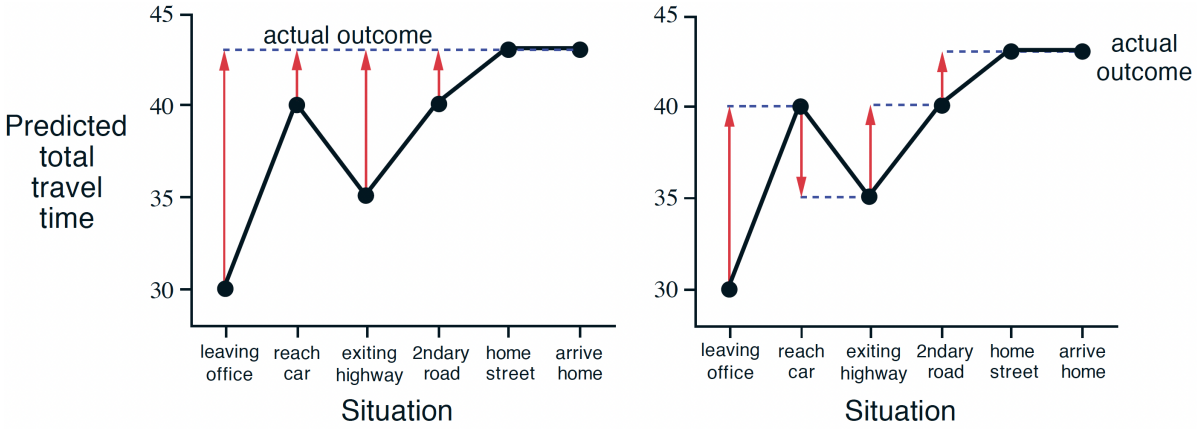
*have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes. Unfortunately, at this point you get stuck behind a slow truck, and the road is too narrow to pass. You end up having to follow the truck until you turn onto the side street where you live at 6:40. Three minutes later, you are home.*

*The sequence of states, times, and predictions is thus as follows:*

| *State* | *Elapsed time (min)* | *Predicted time to go (min)* | *Predicted total time(min)* |
|---|---|---|---|
| *Leaving office, 6 o'clock* | *0* | *30* | *30* |
| *Reach car, raining* | *5* | *35* | *40* |
| *Exiting highway* | *20* | *15* | *35* |
| *Secondary road, behind truck* | *30* | *10* | *40* |
| *Entering home street* | *40* | *3* | *43* |
| *Arrive home* | *43* | *0* | *43* |

*The rewards in this example are the elapsed time on each leg of the journey and we are not discounting ($\gamma = 1$). Then, the value of each state is the expected time to go.*

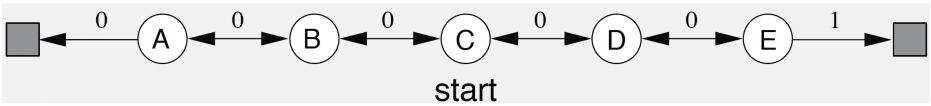
*In the following figure, the red arrows represent the changes in predictions recommended by constant-$\alpha$ Monte Carlo methods (left) and $TD(0)$ (right).*




*Observe that the error in the Monte Carlo method is larger because we need to wait until the end of the episode (until we get home) to be able to update our travel-time estimate. In $TD(0)$, instead, we only need to wait until the next step to update our estimate. Then, each error is proportional to the change over time of the prediction, that is, to the temporal differences of the predictions.*

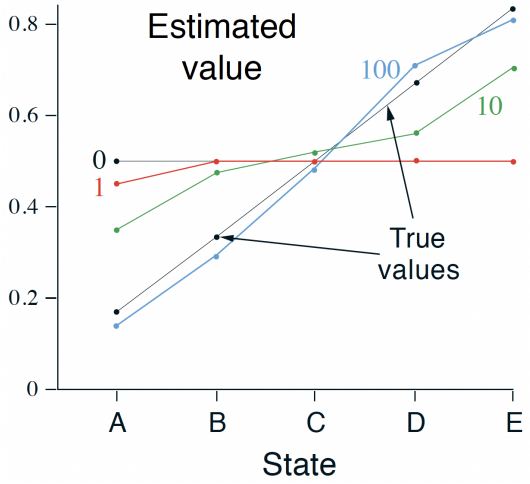
**Example 7.2** (Markov Reward Process)**.** *A Markov Reward Process (MRP) is an MDP without actions. Then, the agent receives a reward depending on the states it visits, and the future state is determined by a DTMC.*

*Let's consider a simple MRP as shown in the following figure:*




*This is an episodic MDP, and all the episodes start at $C$. Then, they proceed either left or right with equal probability. Episodes terminate either on the extreme left or the extreme right. When an episode terminates on the right, a reward of $+1$ occurs; all other rewards are zero. For example, a typical episode might consist of the following state-and-reward sequence: $C, 0, B, 0, C, 0, D, 0, E, 1$. Assume $\gamma = 1$ and $\alpha = 0.1$.*

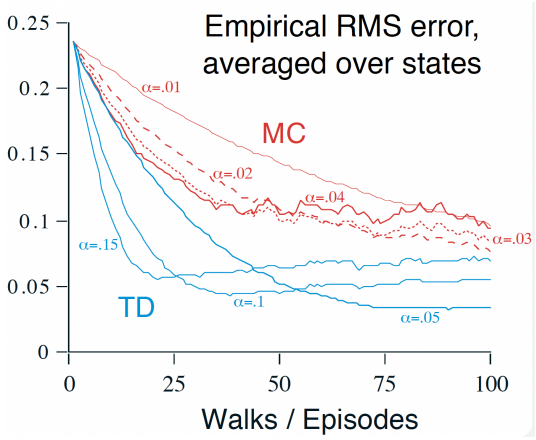
*To learn the values of $v_\pi(s)$ using TD(0), we run many episodes starting from $C$, observe the rewards and update the estimates we have for each state. For 0, 1, 10 and 100 episodes and initial values $V(s) = 0.5$ for all states $s$, we obtain the following result:*




*(a) Provide a method to compute the true values $v_\pi(s)$ for all $s \in \mathcal{S} = \{A, B, C, D, E\}$*

*(b) Interpret the graph. In particular, observe that it appears that the first episode only changed the value of $V(A)$. What does this tell you about what happened in the first episode?*

*(c) In the following graph we show the root mean-squared (RMS) error with respect to the number of episodes of TD(0) for three values of $\alpha$, and Monte-Carlo methods. What can you observe?*




**Solution.**

(a) There are multiple ways to solve this question. In this case, we solve the Bellman equations.

Let $L$ be the terminal state on the left and $R$ the terminal state on the right. Since in MRP there aren't any actions, we omit the subindex of the policy in the Bellman equation. Additionally, the rewards are deterministic, so we don't need to sum over all the possible rewards. Instead, we use $r(s)$ without a summation. We obtain the following:

$$v(s) = \sum_{s' \in \mathcal{S}} [r(s) + v(s')p(s', r(s)|s)] + [r(s) + v(L)p(L|s)] + [r(s) + v(R)p(R|s)]$$

The value function of the terminal states are the rewards received when we reach them. Then, $v(L) = 0$ and $v(R) = 1$. Additionally, $r(s) = 0$ for all $s \in \mathcal{S}$. Using these values and reorganizing terms, we obtain

$$v(s) = \sum_{s' \in \mathcal{S}} v(s')p(s'|s) + p(R|s)$$

Therefore, we obtain the following system of equations:

$$v(A) = \frac{1}{2} v(B)$$

$$v(B) = \frac{1}{2}v(A) + \frac{1}{2}v(C)$$
$$v(C) = \frac{1}{2}v(B) + \frac{1}{2}v(D)$$
$$v(D) = \frac{1}{2}v(C) + \frac{1}{2}v(E)$$
$$v(E) = \frac{1}{2}v(D) + \frac{1}{2}$$

Sequentially plugging the result from each of the equations in the consecutive equation, we obtain:

$$v(A) = \frac{1}{2}v(B), \quad v(B) = \frac{2}{3}v(C), \quad v(C) = \frac{3}{4}v(D), \quad v(D) = \frac{4}{5}v(E), \quad v(E) = \frac{5}{6}$$

Then, using the value of $v(E)$ backwards in these equations, we obtain

$$v(A) = \frac{1}{6}, \quad v(B) = \frac{2}{6}, \quad v(C) = \frac{3}{6}, \quad v(D) = \frac{4}{6}, \quad v(E) = \frac{5}{6}$$

(b) Exactly as we expected, as the number of episodes increases, the approximation $V(s)$ of the value function $v(s)$ improves.

Regarding the first episode, the first observation is that the value of $v(A)$ decreased with respect to the initial value $V(A) = 0.5$. Hence, the terminal state was $L$.

To illustrate what TD(0) is doing, let's track the iterations according to Algorithm 7.1. Observe that in each iteration, we only update the value of the state $S$ that is being visited. The simplest sequence of states to reach the terminal state on the left is $C, B, A, L$. Using this sequence of states, we have:

Step 0. Initialization:

$$V(A) = V(B) = V(C) = V(D) = V(E) = 0.5, \quad V(L) = V(R) = 0$$

We initialize the terminal states at 0 because there is no future after we reach them.

Step 1. Initialize $S = C$ and observe reward 0 and next state $S' = B$. Then, we obtain

$$V(C) \leftarrow V(C) + 0.1[V(B) - V(C)] = 0.5 + 0.1[0.5 - 0.5] = 0.5$$

Step 2. $S = B$, observe reward 0 and next state $S' = A$. Then, we obtain

$$V(B) \leftarrow V(B) + 0.1[V(A) - V(B)] = 0.5 + 0.1[0.5 - 0.5] = 0.5$$

Step 3. $S = A$, observe reward 0 and next state $S' = L$. Then, we obtain

$$V(A) \leftarrow V(A) + 0.1[V(L) - V(A)] = 0.5 + 0.1[0 - 0.5] = 0.45$$

The episode ends because we have reached a terminal state. Hence, our estimates of the value function at each state are:

$$V(A) = 0.45, \quad V(B) = V(C) = V(D) = V(E) = 0.5, \quad V(L) = V(R) = 0$$

(c) The first observation is that TD seems to be considerably better than Monte Carlo for this problem. Additionally, within TD runs, the smaller $\alpha$, the better the approximation of the value function we obtain.

□

## 7.2 Sarsa: On-policy TD Control

We now move to solving the control problem, that is, improving the policy. We will go through two methods: on-policy and off-policy. On-policy methods evaluate and improve the policy that is being used to make the decisions. Off-policy methods, instead, evaluate or improve a policy different from the one being used.

We start with Sarsa, an on-policy method. The main idea is to learn the action-value functions in each step of an episode. In TD prediction, our goal was learning the value function (or the value of a state). In this case, we use the same idea to learn the value of action-state pairs, that is, to learn $q_\pi(s,a)$ for each $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. Formally, we use the following update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

In every step, we must choose an action $A_t$ to evaluate and, therefore, we need to make sure that we balance exploration and exploitation. For example, we may use $\epsilon$-greedy to choose the action for each state $s$.

Observe that we update the value of $Q(S_t, A_t)$ in function of the random variables $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$. These five variables give the name to the algorithm.

A pseudocode for the Sarsa algorithm is provided below.

**Algorithm 7.2** (Sarsa on-policy for TD control)**.**

*0.* ***Input:*** *Step size $\alpha \in (0, 1]$*

*1.* ***Initialization:*** *Initialize $Q(s,a)$ at arbitrary values for $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.*
** Terminal states $s_T$ must be initialized at $Q(s_T, a) = 0$*

*2. Loop for each episode:*

*2.1. Initialize $S$*

*2.2. Choose action $A$ from $S$ using policy derived from $Q$*

*2.3. Loop for each step of episode:*

*2.3.1. Take action $A$, observe $R$ and $S'$*

*2.3.2. Choose action $A'$ from state $S'$ using policy derived from $Q$*

*2.3.3. Update estimate: $Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma Q(S', A') - Q(S,A)\right]$*

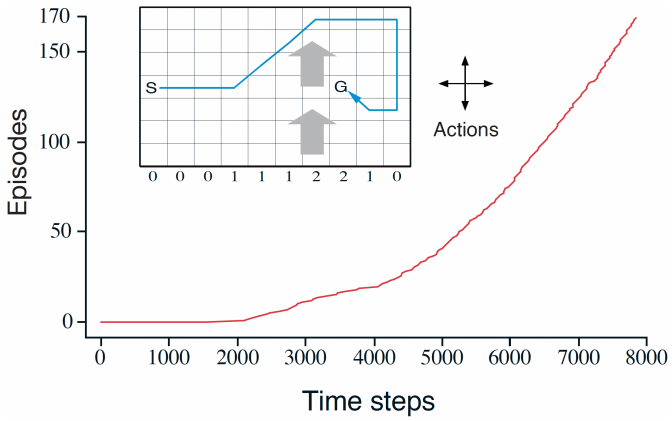
*2.3.4. Update state and action: $S \leftarrow S'$, $A \leftarrow A'$.*

Observe that we must choose an action in steps 2.2. and 2.3.2.. This choice of actions must ensure a balance between exploration and exploitation. For example, we might use $\epsilon$-greedy.

Let's close this section with an example in which Sarsa was used.

**Example 7.3.** *The figure represents a grid game with a start node $S$ and a goal node $G$. The actions are the standard: up, down, left and right. In the middle region there is an upward "wind" that shifts states. The number of cells that the wind shifts varies from column to column and are given under the grid in the figure. For example, if you are one cell to the right of the node $G$ and choose action left, you end up in the cell above $G$. This is an undiscounted episodic task, with constant rewards of -1 until the goal is reached.*

*The graph shows the results of applying Sarsa with $\epsilon$-greedy to choose the action, where $\epsilon = 0.1$ and $\alpha = 0.5$. The initial values are $Q(s,a) = 0$ for all $s, a$. The horizontal axis shows the cumulative number of steps needed in all the episodes, and the vertical axis shows the number of episodes. Then, the slope of the curve represents the number of steps needed per episode to reach a terminal state.*




*Observe that the curve starts flat at 0 for the first* $\sim 2000$ *time steps, that is, the first episode took nearly 2000 steps. This makes sense because the agent does not know the wind, and needs to learn the task and the conditions of the environment. In further episodes, the slope is strictly positive and increasing. This indicates that the agent is learning what to do because it is reducing the number of steps needed to complete an episode. Around step 6000, the slope stops increasing and stabilizes. This means that we cannot keep improving the number of steps. A rough calculation indicates that the slope is* $\sim 20$ *steps per episode after step* $6500$*. As a reference, the optimal strategy in blue in the grid has 15 steps. Then, it makes sense that the slope stabilizes roughly around 20.*

## 7.3 $Q$-learning: Off-policy TD control

The main idea of $Q$-learning is to learn directly an approximation of the optimal action-value function $q_*$, independent of the policy being used. The update is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

In this case, we compare the current estimate $Q(S_t, A_t)$ to the maximum future rewards by choosing the greedy action for state $S_{t+1}$. The policy still has a small effect, but only when selecting which state-action pairs are visited.

With the update above, we have the following pseudocode.

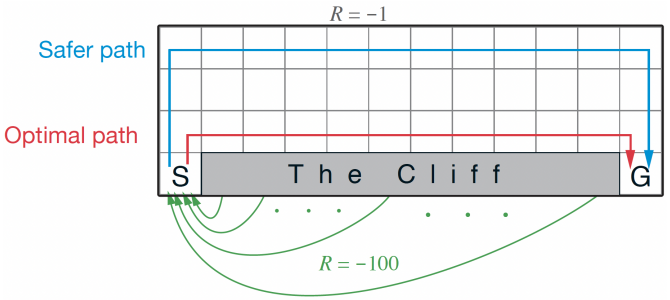
**Algorithm 7.3** ($Q$-learning)**.**

0. ***Input:*** *Step size* $\alpha \in (0, 1]$
1. ***Initialization:*** *Initialize* $Q(s, a)$ *for all* $s \in \mathcal{S}$ *and* $a \in \mathcal{A}(s)$ *at an arbitrary value.*
   * *Terminal states* $s_T$ *must be initialized at* $Q(s_T, a) = 0$
2. *Loop for each episode:*
   - 2.1. *Initialize* $S$
   - 2.2. *Loop for each step of episode:*
     - 2.2.1. *Choose action* $A$ *from* $S$ *using a policy derived from* $Q$
     - 2.2.2. *Take action* $A$*, observe reward* $R$ *and next state* $S'$
     - 2.2.3. *Update estimate:* $Q(S, A) \leftarrow Q(S, A) + \alpha \left[ R + \gamma \max_a Q(S', a) - Q(S, A) \right]$
     - 2.2.4. *Update state:* $S \leftarrow S'$

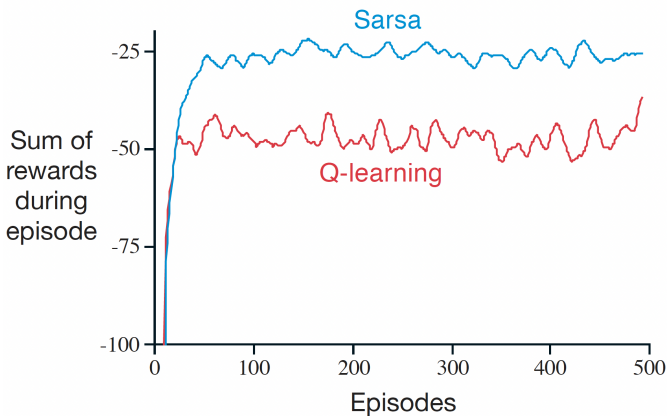
Similarly to Sarsa, we balance exploration and exploitation when we choose the action $A$ to update the state-action value estimate (in this case, step 2.2.1.). One way to do it is using $\epsilon$-greedy.

Let's close with an example in which we compare the outcome of Sarsa and $Q$-learning in a specific setting.

**Example 7.4.** *Consider the grid world shown in the figure. The agent starts at* $S$ *and can take actions up, down, left or right to move through the grid. An episode ends when the agent gets to* $G$*. The rewards are -1 on all transitions, except those to the cliff. Stepping into this region incurs a reward of -100 and the agent is instantly back at cell* $S$*.*




*The following graph shows the performance of the Sarsa and Q-learning methods with $\epsilon$-greedy and $\epsilon = 0.1$.*




*What can you observe?*

**Solution.**

- Both methods have a learning period of 20 episodes, and then the rewards per episode stabilize around a value. For Sarsa, this value is -25 and for $Q$-learning, around -50. Then, based on the rewards, Sarsa is better.
- If we look at the grid above, the optimal path found by Sarsa is longer than the optimal path learned by $Q$-learning. At first, we might think this is contradicting with the performance results from the later graph. However, it makes sense because we are choosing the action from each step using $\epsilon$-greedy, so there is randomness. $Q$-learning stays very close to the cliff, and a random action can be going down. In such case, the reward is immediately -100. Sarsa, instead, has more randomness in the choice of state-action pairs to visit. Then, it encourages paths that are further away from the cliff and, hence, safer.

□

# Chapter 3

# Queueing Theory

We all have experienced waiting in line to get help with a specific request. Classic examples are waiting to check out at a supermarket, waiting for a doctor's appointment, at the DMV to get/renew our driver's license, etc. The longer the wait, the more we wonder why there are 3 empty processing stations while we wait, or how come we had an appointment and we still need to wait 30 minutes. Don't get me wrong; I also wonder why this happens. In this part of the course, we will learn how to analyze different ways to use the resources and how they affect the customers' waiting time.

We start with a quick review of Continuous-Time Markov Chains (CTMCs).

## 8 Continuous-Time Markov Chains

As the name suggests, Continuous-Time Markov Chains (CTMCs) are similar to DTMCs and the main difference is that now time is a real number. In DTMCs we split the timeline in time steps and 'observed' what happens in one time step. In CTMCs we get rid of the notion of steps and, instead, we wait until there is a transition to a different state.

As we will see later, the Markov property implies that the time until the next transition is exponentially distributed and, hence, the number of transitions in a fixed interval of time is a Poisson process. Before we get to CTMCs, let's quickly review these two concepts.

### 8.1 Review the Exponential Distribution

Let's start with the definition of the exponential distribution.

**Definition 8.1.** *A random variable $X$ is exponentially distributed with rate $\lambda$, denoted $X \sim Expo(\lambda)$, if its probability density function is*

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & , \text{ if } x \geq 0 \\ 0 & , \text{ if } x < 0 \end{cases}$$

Some of the most well-known properties of the exponential distribution are:

(1) Cumulative distribution function:

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & , \text{ if } x \geq 0 \\ 0 & , \text{ if } x < 0 \end{cases}$$

(2) Expected value and variance:

$$\mathbb{E}[X] = \frac{1}{\lambda} \qquad \text{Var}[X] = \frac{1}{\lambda^2}$$

Intuitively, an exponentially distributed random variable $X$ represents time until something happens. Then, the rate $\lambda$ represents the expected number of events/transitions per time unit.

(3) Memoryless property:

$$\mathbb{P}[X > t + s | X > s] = \mathbb{P}[X > t] \qquad \forall s, t \geq 0$$

In words, the memoryless property means that if we already 'waited' for $s$ minutes, the probability to wait another $t$ minutes is the same as if we didn't wait anything in the past. That is, the exponential random variable $X$ 'forgets' its value when it comes to how much it can grow.

Further, the only continuous distribution that satisfies the memoryless property is the exponential distribution.

Let's see quick example.

**Example 8.1.** *Suppose the time a customer spends at the bank is exponentially distributed with mean 7 minutes.*

*(a) What is the probability that the customer waits more than 5 minutes?*

*(b) If the customer has already waited 10 minutes, what is the probability that the total time he/she spends in the bank is more than 15 minutes?*

*(c) What is the expected total time that the customer will wait if he/she already waited for 10 minutes?*

**Solution.** The mean of the exponential distribution is 7 minutes. Then, $\lambda = \frac{1}{7}$. Therefore, we have:

(a) We need to compute

$$\mathbb{P}[X > 5] = 1 - F(5) = 1 - \left[1 - e^{-\frac{5}{7}}\right] = e^{-\frac{5}{7}} \approx 0.71$$

(b) Since the customer already waited for 10 minutes, we condition on $X > 10$ and obtain

$$\begin{aligned} \mathbb{P}[X > 15 | X > 10] &= \mathbb{P}[X > 5] \qquad \text{(memoryless property)} \\ &= e^{-\frac{5}{7}} \approx 0.71 \end{aligned}$$

(c) In this case we need to compute $\mathbb{E}[X | X > 10]$. The memoryless property says that $X$ starts over after 10. Then,

$$\mathbb{E}[X | X > 10] = 10 + \mathbb{E}[X] = 10 + \frac{1}{\lambda} = 10 + 7 = 17.$$

□

The following two properties are extremely useful when more than one exponentially distributed time is running. For example, the time until one of two servers finishes processing a request.

(4) Suppose $X_1 \sim Expo(\lambda_1)$, $X_2 \sim Expo(\lambda_2)$ and $X_1, X_2$ are independent random variables. Then, $X = \min\{X_1, X_2\} \sim Expo(\lambda_1 + \lambda_2)$.

(5) Suppose $X_1 \sim Expo(\lambda_1)$, $X_2 \sim Expo(\lambda_2)$ and $X_1, X_2$ are independent random variables. Then,

$$\mathbb{P}[X_1 < X_2] = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

**Example 8.2.** *Suppose a post office has two clerks and there are two customers being served. Customer A is being served by the experienced server, whose processing times are exponentially distributed with mean 10 minutes; and customer B is being processed by the novice clerk, who also has exponentially distributed service times, but with mean 15 minutes.*

*Customer C arrives when both clerks are busy.*

*(a) What is the distribution of customer C's waiting time?*

*(b) What is the probability that customer C is the last one to leave the post office?*

**Solution.**

(a) Customer $C$ waits until the first clerk is free, that is, until the minimum between the processing time of customers $A$ and $B$.

Let $T_A$ be the processing time of customer $A$, and $T_B$ of customer $B$. Then, $T_A \sim Expo\left(\frac{1}{10}\right)$ and $T_B \sim \left(\frac{1}{15}\right)$. By the memoryless property, the remaining processing time has the same distribution as the total processing time.

Using $W$ for the waiting time of customer $C$, we obtain $W = \min\{T_A, T_B\}$. Then, $W \sim Expo\left(\frac{1}{10} + \frac{1}{15}\right) = Expo\left(\frac{1}{6}\right)$.

(b) Let $T_C$ be the processing time of customer $C$. The distribution of $T_C$ depends on which clerk finishes first. If the experienced clerk (processing $A$) finishes first, then $T_C \sim Expo\left(\frac{1}{10}\right)$, and if the novice clerk finishes first, then $T_C \sim Expo\left(\frac{1}{15}\right)$. Therefore,

$$T_C | T_A < T_B \sim T_A \qquad \text{and} \qquad T_C | T_A > T_B \sim T_B$$

Then, the probability that $C$ is the last customer in the post office is

$$\begin{aligned}
\mathbb{P}[C \text{ is last}] &= \mathbb{P}[T_C > T_A | T_A > T_B]\mathbb{P}[T_A > T_B] + \mathbb{P}[T_C > T_B | T_A < T_B]\mathbb{P}[T_A < T_B] \\
&= \mathbb{P}[T_B > T_A]\mathbb{P}[T_A > T_B] + \mathbb{P}[T_A > T_B]\mathbb{P}[T_A < T_B] \\
&= 2\mathbb{P}[T_A < T_B]\,\mathbb{P}[T_B < T_A] \\
&= 2\left(\frac{\frac{1}{10}}{\frac{1}{10}+\frac{1}{15}}\right)\left(\frac{\frac{1}{15}}{\frac{1}{10}+\frac{1}{15}}\right) = \frac{12}{25} = 0.48
\end{aligned}$$

□

## 8.2 Review of Poisson Process

The exponential distribution is a great approximation for interarrival times of customers in many settings. As much as we want to study the time between arrivals, we would like to study the number of arrivals in a specific time interval. To do that, we use the Poisson process. There are several ways to define it, but here we stick with the simplest.

**Definition 8.2.** *A Poisson process with rate $\lambda$ is a counting process $\{N(t) : t \geq 0\}$ such that the time between consecutive events are independent and identically distributed exponential random variables with rate $\lambda$.*

Some of the most important properties of the Poisson distribution are given below:

(1) If $\{N(t) : t \geq 0\}$ is a Poisson process with rate $\lambda$, then $N(t)$ is a Poisson random variable with parameter $\lambda t$, that is,

$$\mathbb{P}[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

(2) Merging independent Poisson processes: Suppose $\{N_1(t) : t \geq 0\}$ and $\{N_2(t) : t \geq 0\}$ are independent Poisson processes, where $N_i(t)$ has rate $\lambda_i$. Let $N(t) = N_1(t) + N_2(t)$. Then, $\{N(t) : t \geq 0\}$ is a Poisson process with rate $\lambda_1 + \lambda_2$.

(3) Poisson splitting: Suppose $\{N(t) : t \geq 0\}$ is a Poisson process with rate $\lambda$. Suppose each event is classified as type 1 with probability $p$, and as type 2 with probability $1-p$. If we use $N_i(t)$ to denote the number of events type $i$ in the interval $(0, t)$, then $\{N_1(t) : t \geq 0\}$ is a Poisson process with rate $\lambda_1 = \lambda p$ and $\{N_2(t) : t \geq 0\}$ is a Poisson process with rate $\lambda_2 = \lambda(1-p)$. Further, $N_1(t)$ and $N_2(t)$ are independent processes.

Let's see some examples.

**Example 8.3.** *If immigrants to area A arrive at a Poisson rate of ten per week, and if each immigrant is of English descent with probability $\frac{1}{12}$, then what is the probability that no people of English descent will emigrate to area A during the month of February?*

**Solution.** Let $N(t)$ be the number of immigrants to area A. Then, $\{N(t) : t \geq 0\}$ is a Poisson process with rate $\lambda = 10$.

If we use $N_E(t)$ to denote the number of immigrants of English descent, then the Poisson splitting property establishes that $\{N_E(t) : t \geq 0\}$ is a Poisson process with rate $\lambda_E = \frac{10}{12}$. Therefore, the probability that no immigrant is from English descent in February is

$$\mathbb{P}\left[N_E(4) = 0\right] = \frac{e^{-\frac{10}{12}*4}\left(\frac{10}{12}*4\right)^0}{0!} = e^{-\frac{10}{12}*4} \approx 0.036$$

□

**Example 8.4.** *An airplane with 300 passengers from Chile just landed in Atlanta, and all the passengers need to be interviewed by the immigration police before entering the US. There are 6 stations working, and each of them takes an exponentially distributed time with mean 2 minutes in processing each passenger. After the immigration control, each passenger takes their luggage instantaneously.*

*Each of these passengers takes the shuttle to the rental-car facility with probability $\frac{1}{3}$. If the shuttle arrives in 10 minutes, what is the expected number of Chilean passengers it will take?*

**Solution.** First, observe that 6 clerks are processing passengers and the processing times are exponential with rate $\lambda = \frac{1}{2}$. Then, the total number of passengers departing the airport $N(t)$ is a Poisson process with rate $5\lambda$. Then, with probability 0.5 they will take the shuttle. Therefore, the number of passengers that take the shuttle $N_S(t)$ is a Poisson process with rate $\lambda_S = \frac{6\lambda}{3} = \frac{6}{6} = 1$. Hence, the expected number of Chileans in the shuttle is

$$\mathbb{E}\left[N_S(10)\right] = 1 * 10 = 10$$

□

## 8.3 Continuous-Time Markov Chains

As the name suggests, Continuous-Time Markov Chains (CTMC's) are the continuous-time analogous of DTMCs. Instead of looking at fixed-time intervals, we will look at the process as it evolves and observe the time until the next transition. Let's see the formal definition.

**Definition 8.3.** *A continuous-time stochastic process $\{X(t) : t \geq 0\}$ with state space $\mathcal{S}$, is a Continuous-Time Markov Chain if for all $s, t \geq 0$ and states $i, j \in \mathcal{S}$ and $x(u) \in \mathcal{S}$ for all $0 \leq u < s$ we have*

*(i) Markov property:*

$$\mathbb{P}\left[X(t+s) = j \,|\, X(s) = i,\, X(u) = x(u)\ \forall 0 \leq u < s\right] = \mathbb{P}\left[X(t+s) = j \,|\, X(s) = i\right]$$

*(ii) Stationarity:*

$$\mathbb{P}\left[X(t+s) = j \,|\, X(s) = i\right] = \mathbb{P}\left[X(t) = j \,|\, X(0) = s\right]$$

Similarly to DTMC's, the Markov property says that the future depends on the past only through the present state, and the stationarity property says that the transition probabilities depend on the length of the interval, but not on the specific times we are looking at.

The Markov and stationarity properties imply that the time that the CTMC spends at a state $i$, denoted by $T_i$, is exponentially distributed. Let $\nu_i$ be the rate of such exponential distribution. After leaving state $i$, the chain visits state $j$ with probability $p_{ij}$ where

$$p_{ii} = 0, \qquad \text{and} \qquad \sum_{j \in \mathcal{S}} p_{ij} = 1$$

**Example 8.5.** *Consider a single-server queue where the arrivals occur according to a Poisson process with rate $\lambda$ and the server processes each customer's requests in an exponentially distributed time with rate $\mu$. This is called an $M/M/1$ queue because there is one server, and the arrival (first $M$) and service (second $M$) processes are Memoryless.*

*Model the number of customers in the system of the $M/M/1$ queue as a CTMC, that is, specify the state space, the rates $\nu_i$ and the transition probabilities $p_{ij}$ for all $i, j$.*

**Solution.** As suggested in the question, the states are defined by

$$X(t) = \text{ number of customers in the system (including server) at time } t$$

Then, the state space is the set of nonnegative integers.

Next we compute the rates $\nu_i$. The only way to leave state 0 is if there is an arrival, which happens in an exponentially distributed time with rate $\lambda$. Then,

$$\nu_0 = \lambda$$

For other states $i \geq 1$, the CTMC transitions to another state whenever there is an arrival or a departure. Indeed, whenever the first of those two happens. Then, the time spent in state $i$, $T_i$, is the minimum of two exponential random variables (time until next arrival and time until next departure) and we have

$$\nu_i = \lambda + \mu \qquad \forall i \geq 1$$

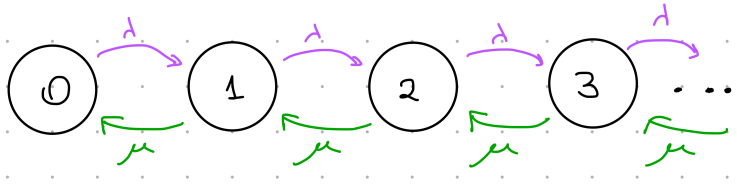
Finally, we need to model the transition probabilities. Again, state 0 is easy and we have

$$p_{0j} = \begin{cases} 1 & \text{, if } j = 1 \\ 0 & \text{, otherwise} \end{cases}$$

For other states $i$, the transition will be to state $i+1$ if there is an arrival and to state $i-1$ if there is a departure. The probability of an arrival is the probability that the time until next arrival is smaller than the time until the next departure. Since both times are exponential, we obtain

$$p_{ij} = \begin{cases} \dfrac{\lambda}{\lambda+\mu} & \text{, if } j = i+1 \\ \dfrac{\mu}{\lambda+\mu} & \text{, if } j = i-1 \\ 0 & \text{, otherwise} \end{cases} \qquad \forall i \geq 1$$

Similarly to DTMC's, we can also draw a diagram with the states and transitions. However, in this case we write the **rate** at which the chain transitions to other states instead of the probability. For an $M/M/1$ queue we have




□

**Example 8.6.** *Now consider a queue with $s$ equal servers. Each of the servers processes jobs in an exponentially distributed time with rate $\mu$, and each job can be processed by at most one server. If there are less jobs in the system than the number of servers, the remaining servers idle. For example, if $s = 5$ and there are 3 jobs in the system, 2 of the servers idle.*

*The arrivals to the system occur according to a Poisson process of rate $\lambda$. If at least one of the servers is idling, a new arrival immediately starts service. Otherwise, it waits in a unique line. This is called an $M/M/s$ queue.*

*Model the number of customers in the system of an $M/M/s$ queue as a CTMC.*

**Solution.** Similarly to an $M/M/1$ queue, the state $X(t)$ represents the number of jobs in the system at time $t$, and the state space is the set of nonnegative numbers.

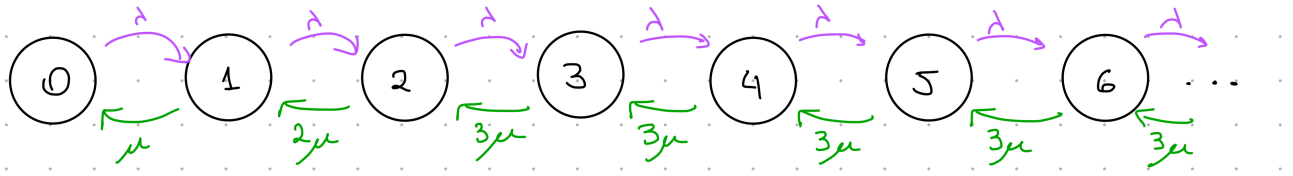
The rate of each state depends on how many servers are working. Specifically, we have

$$\nu_i = \begin{cases} \lambda + i\mu & \text{, if } i \leq s \\ \lambda + s\mu & \text{, if } i > s \end{cases}$$

The transition probabilities become

$$p_{ij} = \begin{cases} \dfrac{\lambda}{\lambda + i\mu} & \text{, if } i \leq s \text{ and } j = i+1 \\ \dfrac{i\mu}{\lambda + i\mu} & \text{, if } i \leq s \text{ and } j = i-1 \\ \dfrac{\lambda}{\lambda + s\mu} & \text{, if } i > s \text{ and } j = i+1 \\ \dfrac{s\mu}{\lambda + s\mu} & \text{, if } i > s \text{ and } j = i-1 \\ 0 & \text{, otherwise} \end{cases} \qquad \forall i, j$$

In this case, the diagram for $s = 3$ becomes




□

In the last two examples, the only possible transitions from any state $i \geq 0$ are to state $i+1$ and $i-1$. This characteristic holds in many applications, and it has a special name.

**Definition 8.4.** *A birth-and-death process is a CTMC where the states represent the number of people in the system at a given time. Whenever there are $n$ people in the system:*

*(i) New arrivals enter the system at an exponential rate $\lambda_n$*

*(ii) People leave the system at an exponential rate $\mu_n$*

*The parameters $\{\lambda_n\}_{n=0}^{\infty}$ are called the birth or arrival rates, and $\{\mu_n\}_{n=1}^{\infty}$ the death or departure rates.*

By definition, in birth-death processes we have the following rates:

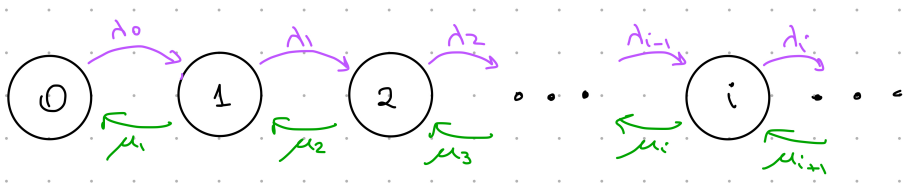
$$\nu_0 = \lambda_0, \qquad \nu_i = \lambda_i + \mu_i \quad \forall i \geq 1$$

the following transition probabilities:

$$p_{01} = 1$$

$$p_{ij} = \begin{cases} \dfrac{\lambda_i}{\lambda_i + \mu_i} & \text{, if } j = i+1 \\ \dfrac{\mu_i}{\lambda_i + \mu_i} & \text{, if } j = i-1 \\ 0 & \text{, otherwise} \end{cases} \qquad \forall i \geq 1$$

and the diagram is



**Example 8.7.** *Use the notation of birth-death processes for the arrival and departure rates in an $M/M/1$ and an $M/M/s$ queue.*

**Solution.** In an $M/M/1$ queue, we have

$$\lambda_i = \lambda \quad \forall i \geq 0, \qquad \mu_i = \mu \quad \forall i \geq 1$$

and in an $M/M/s$ queue, we have

$$\lambda_i = \lambda \quad \forall i \geq 0, \qquad \mu_i = \begin{cases} i\mu & \text{, if } i \leq s \\ s\mu & \text{, if } i > s \end{cases} \quad \forall i \geq 1$$

□

### The transition probability function

When we did DTMC's, the transition probability matrix $P$ with elements $p_{ij}$ represented the probability that the chain goes from state $i$ to $j$ in one time step, and we used $P^n$ to compute the transition probabilities in $n$ steps. In CTMC's we no longer have the notion of a time step. Instead, we need to observe the total amount of time $t$ that has passed to compute the transition probabilities. Specifically, we use

$$p_{ij}(t) = \mathbb{P}\left[X(t+s) = j \,|\, X(s) = i\right],$$

that is, the probability that starting at state $i$, the CTMC is in state $j$ after $t$ time units. Due to the Markov and stationarity properties, the matrix $P(t)$ satisfies the following property.

$$p_{ij}(t+s) = \sum_k p_{ik}(t) p_{kj}(s) \qquad \forall s, t \geq 0. \tag{3.1}$$

These equations are known as Chapman-Kolmogorov equations

We will derive a set of differential equations that the matrix $P(t)$ satisfies but, before that, we introduce one more concept and study its properties.

**Definition 8.5.** *The instantaneous transition rates $q_{ij}$ are defined by*

$$q_{ij} = \nu_i p_{ij}$$

*and represent the rate at which the CTMC goes from state $i$ to $j$.*

By definition of the instantaneous transition rates, we have

$$\nu_i = \sum_j q_{ij} \qquad \text{and} \qquad p_{ij} = \frac{1}{q_{ij}} \sum_k q_{ik} \qquad \forall i, j$$

Additionally, the following limiting properties are satisfied. For all states $i$, we have

$$\lim_{h \to 0} \frac{1 - P_{ii}(h)}{h} = \nu_i \tag{3.2}$$

and for all states $i \neq j$, we have

$$\lim_{h \to 0} \frac{P_{ij}(h)}{h} = q_{ij} \tag{3.3}$$

Now we're ready for the set of differential equations that will allow us to compute $p_{ij}(t)$.

**Theorem 8.1** (Kolmogorov's forward equations)**.** *Under suitable regularity conditions, for all $i, j$ and all $t \geq 0$ we have*

$$p'_{ij}(t) = \sum_{k \neq j} q_{kj} p_{ik}(t) - \nu_j p_{ij}(t)$$

*where the derivative is taken with respect to $t$.*

The proof shows where these (magic) equations come from, so we will go through it.

*Proof.* From the Chapman-Kolmogorov equations, we have

$$\begin{aligned} p_{ij}(t+h) - p_{ij}(t) &= \sum_k [p_{ik}(t)p_{kj}(h)] - p_{ij}(t) \\ &= \sum_{k\neq j} [p_{ik}(t)p_{kj}(h)] + [p_{ij}(t)p_{jj}(h)] - p_{ij}(t) \\ &= \sum_{k\neq j} [p_{ik}(t)p_{kj}(h)] - [1 - p_{jj}(h)]\, p_{ij}(t) \end{aligned}$$

Therefore,

$$\lim_{h\to 0} \frac{p_{ij}(t+h) - p_{ij}(t)}{h} = \lim_{h\to 0} \sum_{k\neq j} p_{ik}(t) \left[\frac{p_{kj}(h)}{h}\right] - \lim_{h\to 0} \left[\frac{1 - p_{jj}(h)}{h}\right] p_{ij}(t)$$

Assuming that we can interchange the sum and the limit (here's where the regularity conditions come to picture), we obtain

$$p'_{ij}(t) = \sum_{k\neq j} p_{ik}(t)q_{kj} - \nu_j p_{ij}(t)$$

□

**Example 8.8.** *Write the forward equations for a birth-death process.*

**Solution.** Based on the rates from Definition 8.4, we have

$$\begin{aligned} q_{01} &= \lambda_0 \\ q_{ij} &= \begin{cases} \lambda_i & \text{, if } j = i+1 \\ \mu_i & \text{, if } j = i-1 \\ 0 & \text{, otherwise} \end{cases} \qquad \forall i \geq 1 \end{aligned}$$

Then, we obtain the following forward equations:

$$\begin{aligned} p'_{i0}(t) &= p_{i1}(t)q_{10} - \nu_0 p_{i0}(t) \\ &= p_{i1}(t)\mu_1 - p_{i0}(t)\lambda_0 \end{aligned}$$

and for $j \geq 1$ we have

$$\begin{aligned} p'_{ij}(t) &= q_{j-1,j}p_{i,j-1}(t) + q_{j+1,j}p_{i,j+1}(t) - \nu_j p_{ij}(t) \\ &= \lambda_{j-1}p_{i,j-1}(t) + \mu_{j+1}p_{i,j+1}(t) - (\lambda_j + \nu_j)p_{ij}(t) \end{aligned}$$

□

### Limiting probabilities

In general, solving the system of differential equations to obtain $P(t)$ exactly is challenging and, in many cases, intractable. In this section we learn how to compute the steady-state probabilities.

Let $\pi_j$ be the proportion of time that the CTMC is in state $j$ in the long run. Then,

$$\sum_j \pi_j = 1$$

The intuitions of $\pi_j$'s are similar to DTMC's, so we directly jump to the result.

**Theorem 8.2.** *Consider a CTMC $\{X(t) : t \geq 0\}$ with state space $\mathcal{S}$. If*

*(i) the chain is irreducible, that is, all the states communicate with each other, and*

*(ii) all the states are positive recurrent, that is, the mean time between visits to a state is finite,*

*then the limiting probabilities exist:*

$$\pi_j = \lim_{t\to\infty} p_{ij}(t)$$

*and* $\lim_{t\to\infty} p'_{ij}(t) = 0$*, that is,*

$$\nu_j \pi_j = \sum_{k\neq j} q_{kj}\nu_k \qquad \forall j \in \mathcal{S} \tag{3.4}$$

The set of equations (3.4) is known as the balance equations, and represent that the rate at which the CTMC leaves state $j$ ($\nu_j\pi_j$) equals the rate at which it enters state $j$ ($\sum_{k\neq j} q_{kj}\nu_k$). Intuitively, this must be true for all states in steady state because, otherwise, the CTMC would get stuck at a single state (or a subset of states), and this situation would contradict irreducibility.

The theorem above plus the condition $\sum_j \pi_j = 1$ allow us to compute the limiting distribution of a CTMC without solving any set of differential equations.

**Example 8.9.** *Compute the limiting distribution $\pi_j$ of a birth-death process.*

**Solution.** The balance equations are:

| State $j$ | Rate to leave state $j$ | = | Rate to enter state $j$ |
|---|---|---|---|
| 0 | $\lambda_0\pi_0$ | = | $\mu_1\pi_1$ |
| 1 | $(\lambda_1+\mu_1)\pi_1$ | = | $\mu_2\pi_2 + \lambda_0\pi_0$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n \geq 1$ | $(\lambda_n+\mu_n)\pi_n$ | = | $\mu_{n+1}\pi_{n+1} + \lambda_{n-1}\pi_{n-1}$ |

Reorganizing terms, we obtain

$$\lambda_j\pi_j = \mu_{j+1}\pi_{j+1} \qquad \forall j \geq 0$$

Hence, using that $\sum_j \pi_j = 1$ we obtain

$$\pi_0 = \left(1 + \sum_{n=1}^{\infty}\prod_{j=1}^{n}\frac{\lambda_{j-1}}{\mu_j}\right)^{-1} = \left(1 + \sum_{n=1}^{\infty}\frac{\lambda_0\cdots\lambda_{n-1}}{\mu_1\cdots\mu_n}\right)^{-1}$$

$$\pi_n = \left(\prod_{j=1}^{n}\frac{\lambda_{j-1}}{\mu_i}\right)\pi_0 = \left(\frac{\lambda_0\cdots\lambda_{n-1}}{\mu_1\cdots\mu_n}\right)\pi_0$$

□

**Example 8.10.** *Use the example above to compute the limiting distribution of an $M/M/1$ queue.*

**Solution.** The $M/M/1$ queue is a birth-death process with

$$\lambda_n = \lambda \;\forall n \geq 0 \qquad \text{and} \qquad \mu_n = \mu \;\forall n \geq 1$$

Hence, we obtain

$$\pi_n = \left(\frac{\lambda}{\mu}\right)^n \pi_0$$

The ration between the arrival and processing rates is called the load of the queue and represents how busy the system is. The higher the load, the busier the server. Typically, the letter $\rho$ is used. In this case, we have

$$\rho = \frac{\lambda}{\mu}$$

Then,

$$\pi_n = \rho^n \pi_0$$

To compute $\pi_0$ we solve the sum. We obtain:

$$\begin{aligned} 1 + \sum_{n=1}^{\infty} \frac{\lambda_0 \cdots \lambda_{n-1}}{\mu_1 \cdots \mu_n} &= 1 + \sum_{n=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^n \\ &= \sum_{n=0}^{\infty} \rho^n \\ &= \frac{1}{1-\rho} \qquad \iff \qquad \rho < 1 \end{aligned}$$

Therefore,

$$\pi_n = (1-\rho)\rho^n \qquad \forall n \geq 0$$

□

# 9 Analysis of Queueing Systems

One of the fundamental results in queueing theory is Little's law, which provides a relationship between the expected number of customers/jobs in the system and their expected waiting time.

**Theorem 9.1** (Little's law)**.** *For an irreducible, aperiodic and positive recurrent system,*

$$\mathbb{E}[N] = \lambda \mathbb{E}[T],$$

*where $N$ represents the number of customers in the system, $\lambda$ is the average arrival rate, and $\mathbb{E}[T]$ is the mean time jobs spend in the system.*

**Example 9.1.** *Compute the expected time that customers spend in an $M/M/1$ queue.*

**Solution.** From Example 8.10, we know that the pmf of the number of jobs in the system is

$$\mathbb{P}[N = j] = \pi_j = (1-\rho)\rho^j,$$

where $\rho = \frac{\lambda}{\mu}$ is the load. Observe that the pmf is Geometric with parameter $p = 1 - rho$. Then, the expected number of jobs in the system is

$$\mathbb{E}[N] = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$$

The $M/M/1$ queue is ergodic if $\lambda < \mu$ (or equivalently, is $\rho < 1$) and this is the same condition for existence of the stationary distribution. Then, we may assume it's satisfied.

Hence, we can use Little's law to compute the mean time in the system. We obtain:

$$\mathbb{E}[T] = \frac{1}{\lambda}\mathbb{E}[N] = \frac{1}{\mu - \lambda}$$

□

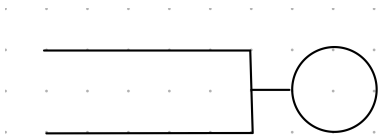
## 9.1 Comparing queue configurations

The goal of this section is comparing the performance of three queue configurations. To do that, we will consider three systems with the same load and will compare the expected number of jobs in the system and the corresponding variance. The difference among systems is the number of servers and the number of waiting lines (queues). Let's first describe each of the three systems.

### The $M/M/1$ queue

This is the simplest system we may have, and we are already familiar with it. The arrival rate is $\lambda$ and, to make the comparison with multi-server queues easier, we will use $n\mu$ to denote the service rate. Hence, we can think of the $M/M/1$ queue as having a *super server* with the capacity of $n$ *regular* servers.

A diagram of this queue is presented below



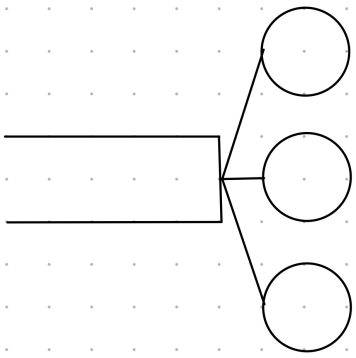
where the circle represents the server, and the open rectangle represents the queue.

In Example 8.10 we computed the stationary distribution of the number of jobs in the system. Then, we can compute the mean and variance analytically. Further, we identified the pmf as Geometric with parameter $\rho = \frac{\lambda}{n\mu}$. Therefore,

$$\mathbb{E}\left[N_{M/M/1}\right] = \frac{\rho}{1-\rho} = \frac{\lambda}{n\mu - \lambda} \qquad \text{and} \qquad \text{Var}\left[N_{M/M/1}\right] = \frac{\rho}{(1-\rho)^2} = \frac{\lambda n\mu}{(n\mu-\lambda)^2}$$

### The $M/M/n$ queue

The $M/M/n$ queue has a single queue, and $n$ identical servers. The arrivals occur according to a Poisson process with rate $\lambda$, and each servers processes jobs in exponentially distributed times with rate $\mu$. A diagram of this queue with $n = 3$ is presented below



The load is the total arrival rate divided by the total service rate. Hence, we also obtain $\rho = \frac{\lambda}{n\mu}$.

To compute the expected number of jobs in the system and the variance, we first compute the pmf in steady state. Recall the formula from Example 8.9

$$\pi_j = \left(\prod_{i=1}^{j} \frac{\lambda_{i-1}}{\mu_i}\right) \pi_0$$

and

$$\lambda_i = \lambda \quad \forall i \geq 0, \qquad \mu_i = \begin{cases} i\mu & \text{, if } i < n \\ n\mu & \text{, if } i \geq n \end{cases}$$

Therefore,

$$\pi_j = \begin{cases} \dfrac{\lambda^j}{j!\mu^j}\pi_0 & \text{, if } j < n \\ \dfrac{\lambda^j}{n!n^{j-n}\mu^j}\pi_0 & \text{, if } j \geq n \end{cases}$$

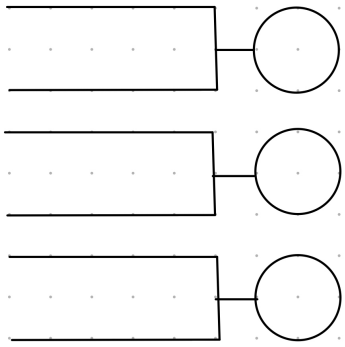
and

$$\begin{aligned}\pi_0 &= \left(1+\sum_{j=1}^{n-1}\frac{\lambda^j}{j!\mu^j}+\sum_{j=n}^{\infty}\frac{\lambda^j}{n!n^{j-n}\mu^j}\right)^{-1}\\ &= \left(1+\sum_{j=1}^{n-1}\frac{\lambda^j}{j!\mu^j}+\frac{\lambda^n}{n!\mu^n}\sum_{k=0}^{\infty}\left(\frac{\lambda}{n\mu}\right)^k\right)^{-1}\\ &= \left(1+\sum_{j=1}^{n-1}\frac{\lambda^j}{j!\mu^j}+\frac{\lambda^n}{n!\mu^n}\frac{\rho}{1-\rho}\right)^{-1}\end{aligned}$$

In this case, the pmf is more complicated and we won't be able to compute closed-form expressions for the expected number of jobs in the system and the variance. Then, we will use simulation.

### Multiple $M/M/1$ queues

In this case we will consider $n$ separate $M/M/1$ queues, each of them with arrival rate $\frac{\lambda}{n}$ and service rate $\mu$. A diagram with 3 queues is presented below



We can think of this system as a single stream of arrivals, which is a Poisson process with rate $\lambda$. Upon arrival, every customer chooses a queue uniformly at random. Then, each queue is selected with probability $\frac{1}{n}$, and this results in an arrival rate of $\frac{\lambda}{n}$ to each queue. Since routing (choice of queue by each customer) is at random, this system behaves as $n$ independent $M/M/1$ queues.

Based on our previous analysis, each of the $n$ queues has mean and variance:

$$\mathbb{E}\left[N_i\right] = \frac{\frac{\lambda}{n}}{\mu-\frac{\lambda}{n}} = \frac{\lambda}{n\mu-\lambda}$$

Then, the expected number of jobs in the system is

$$\mathbb{E}\left[N_{n\,M/M/1}\right] = \frac{n\lambda}{n\mu-\lambda} = n\mathbb{E}\left[N_{M/M/1}\right]$$

so we immediately know that $n$ independent $M/M/1$ queue is worse than a *super* $M/M/1$ queue.

Regarding the variance, since we have $n$ independent $M/M/1$ queue, we have

$$\mathrm{Var}\left[N_{n\,M/M/1}\right] = n\mathrm{Var}\left[N_i\right] = n\frac{\frac{\lambda}{n\mu}}{\left(1-\frac{\lambda}{n\mu}\right)^2} = n\frac{\lambda n\mu}{(n\mu-\lambda)^2} = n\mathrm{Var}\left[N_{M/M/1}\right]$$

### Comparison of systems

We now compare the two systems numerically. We run several cases and in all of them $n\mu = 1$. Then, the load $\rho$ is equal to the total arrival rate to the system $\lambda$. We start comparing the expected total number of customers in the system

| Arrival rate | $n$ | $M/M/1$ | $M/M/n$ | $n\,M/M/1$ |
|---|---|---|---|---|
| 0.5 | 2 | 1 | 1.33 | 2 |
| | 4 | 1 | 2.17 | 4 |
| | 10 | 1 | 5.03 | 10 |
| 0.8 | 2 | 4 | 4.44 | 8 |
| | 4 | 4 | 5.58 | 16 |
| | 10 | 4 | 9.35 | 40 |
| 0.98 | 2 | 48.99 | 49.47 | 97.99 |
| | 4 | 48.99 | 48.45 | 195.99 |
| | 10 | 48.99 | 53.70 | 489.99 |

In all the cases studied above, $n$ independent $M/M/1$ has the largest number of jobs. However, for light loads ($\lambda \leq 0.8$), the *super* $M/M/1$ queue is better than the $M/M/n$ queue, and for high loads ($\lambda = 0.98$), $M/M/n$ is slightly better.

It makes sense that a *super* $M/M1$ queue is better than an $M/M/n$ queue for small loads because, in such cases, the number of customers waiting in line is small. If there are $i < n$ customers in the system, the *super* $M/M/1$ queue is processing jobs at rate $n\mu$, but the $M/M/n$ queue is processing at rate $i\mu$. Then, in light loads, an $M/M/n$ queue 'wastes' part of the service capacity.

Using the same logic, $n$ independent $M/M/1$ queues is the worst because the customers are stuck in the queue they joined, independently of how many servers are empty at a given time. Hence, there might be customers waiting in queue 1, while queues 2 and 3 are empty.

For the variance, we obtain

| Arrival rate | $n$ | $M/M/1$ | $M/M/n$ | $n\,M/M/1$ |
|---|---|---|---|---|
| 0.5 | 2 | 2 | 2.22 | 4 |
| | 4 | 2 | 2.84 | 8 |
| | 10 | 2 | 5.11 | 20 |
| 0.8 | 2 | 20 | 20.28 | 40 |
| | 4 | 20 | 20.88 | 40 |
| | 10 | 20 | 19.71 | 200 |
| 0.98 | 2 | 2449 | 2440 | 4899 |
| | 4 | 2249 | 1990 | 9799 |
| | 10 | 2449 | 2111 | 24,499 |

The behavior is similar to the mean. Again, $n$ independent $M/M/1$ queues gives the worst performance. Also, as the load increases, the variance of the *super* $M/M/1$ queue becomes larger than the variance of the $M/M/n$ queue. This makes sense because with large loads, all the servers are busy. However, if there is a single server and a slow customer, all the waiting customers spend more time in line. However, in an $M/M/n$ queue, a slow customer only 'blocks' one of the $n$ available servers.

## 9.2 The $M/G/1$ queue

We spent some time analyzing $M/M/n$ queues because they are mathematically tractable. In practice, assuming that the arrivals follow a Poisson process is acceptable because usually the arrival of a customer does not influence the arrival of the next customers. The most evident counter-example is rush hours, but in such case one can consider a nonhomogeneous Poisson process where the mean arrival rate is piecewise constant.

The service process, on the other hand, is rarely well represented by exponential times. The sole idea of the memoryless property in service times is disappointing and inaccurate. Hence, a good model for queues is $M/G/1$, where the service times have a general distribution. In this section we learn two approaches to analyze this queue.

### Modeling the number of customers as a DTMC

The number of jobs in the system of an $M/G/1$ queue is clearly not a CTMC because the time spent at states $i \geq 1$ is not exponential. Indeed, it is the minimum between a general distribution and an exponential distribution. Further, if an arrival occurs before a departure, we need to keep track of the amount of time the current customer has been at the server because general distributions are not memoryless.

However, if we define $X_n$ as the number of jobs in the system when the $n^{\text{th}}$ customer leaves the system, we obtain a DTMC. The transition probabilities can be computed if we condition on the number of arrivals that occur during one service time.

### Expected values

As you probably noticed, the analysis becomes considerably more complicated. So, in practice, we work with the expected value of the number of jobs in the system.

One of the most famous results is the Pollaczek-Khintchine formula, that establishes

$$\mathbb{E}[T_Q] = \frac{\lambda \mathbb{E}\left[S^2\right]}{2(1 - \lambda \mathbb{E}[S])},$$

where $T_Q$ is the expected time that a customer spends waiting in line, $\lambda$ is the arrival rate, and $S$ represents the service time of each customer.

Then, one can compute the number of people in the queue using Little's law:

$$\mathbb{E}[N_Q] = \lambda \mathbb{E}[T_Q]$$

Additionally, the time a customer spends in the system is its waiting time plus its processing time (at the server). Hence,

$$\mathbb{E}[T] = \mathbb{E}[T_Q] + \mathbb{E}[S]$$

and the number of customers in the system can also be obtained using Little's law.

Let's see an example.

**Example 9.2.** *Suppose that customers arrive to a single-server system according to a Poisson process with rate $\lambda$, and each customer is one of $r$ types. Suppose that each new arrival is of type $i$ with probability $\alpha_i$, independently of all the customers that arrived before. The time needed to process a customer of type $i$ has distribution $F_i$, with mean $\mu_i$ and variance $\sigma_i^2$.*

*(a) Compute the expected time that a customer type $i$ spends in the system*

*(b) Compute the expected number of customers type $i$ in the system*

**Solution.** The first observation is that this is a special case of an $M/G/1$ customer, where the distribution $G$ is:

$$\begin{aligned} G(x) &= \mathbb{P}[S \leq x] \\ &= \sum_{i=1}^{r} \mathbb{P}[S \leq x \,|\, \text{customer type } i]\, \mathbb{P}[\text{customer type } i] \\ &= \sum_{i=1}^{r} F_i(x)\alpha_i \end{aligned}$$

Then, we compute $\mathbb{E}[S]$ and $\mathbb{E}\left[S^2\right]$ conditioning on the customer type. Specifically, we obtain

$$\mathbb{E}[S] = \sum_{i=1}^{r} \mu_i \alpha_i \qquad \text{and} \qquad \mathbb{E}\left[S^2\right] = \sum_{i=1}^{r} (\mu_i^2 + \sigma_i^2)\alpha_i$$

(a) When a customer arrives to the system, their expected waiting time is independent of the type of customer he/she is. Then, the expected waiting time of a customer type $i$ is

$$\mathbb{E}[T_Q] = \frac{\lambda \mathbb{E}\left[S^2\right]}{2(1 - \lambda \mathbb{E}[S])} = \frac{\lambda \sum_{i=1}^{r} (\mu_i^2 + \sigma_i^2)}{2\left(1 - \sum_{i=1}^{r} \mu_i \alpha_i\right)}$$

and the expected amount of time a customer type $i$ spends in the system becomes

$$\mathbb{E}\left[T^{(i)}\right] = \mu_i + \mathbb{E}\left[T_Q\right]$$

(b) To compute the expected number of customers type $i$ in the system we use Little's law. The arrival rate of customers type $i$ is $\lambda\alpha_i$. Then,

$$\mathbb{E}\left[N^{(i)}\right] = \lambda\alpha_i\mathbb{E}\left[T^{(i)}\right]$$

□