Lecture notes Stochastic Optimization

http://www.math.vu.nl/obp/edu/so

Ger Koole

Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands

http://www.math.vu.nl/∼koole, koole@few.vu.nl

22nd January 2006

1 Introduction

This course is concerned with **dynamic decision problems**. Dynamic refers to the notion of time, i.e., decisions taken at some point in time have consequences at later time instances. This is, historically speaking, the crucial difference with linear programming. For this reason the main solution technique is called **dynamic programming** (Bellman [2]). Often the evolution of the problem is subject to randomness, hence the name stochastic dynamic programming (cf. Ross [14]). Nowadays the usual term is (semi-)Markov decision theory, emphasizing the connection with (semi-)Markov chains. Note also that dynamic programming is usually identified with a solution technique, and not with a class of problems.

The focus is on practical aspects (especially on the control of queueing systems), what we can do with it.

Some typical examples of Markov decision chains are:
- admission control to a queue. Objective: minimize queue length and rejections. Dynamics: Decisions influence future queue length.
- routing in a call center. Objective: minimize waiting of calls. Dynamics: routing decisions influence future availabilities.
- investment decisions, when to buy a new car, etc.

During most of this course we impose the following restrictions on the problems considered:
- there is a single decision maker (no games or decentralized control);
- discrete state spaces and "thus" discrete events (continuous state space and thus also possibly continuously changing states: system theory)

Prior knowledge: basic probability theory (e.g., Poisson process) and some programming experience.

2 Refresher probability theory

(copied from lecture notes modeling of business processes)

The Poisson process and the exponential distribution play a crucial role in many parts of these lecture notes. Recall that for X exponentially distributed with parameter μ holds:

$$F_X(t) = \mathbb{P}(X \leq t) = 1 - e^{-\mu t}, \quad f_X(t) = F_X'(t) = \mu e^{-\mu t},$$

$$\mathbb{E}X = \frac{1}{\mu}, \text{ and } \sigma^2(X) = \mathbb{E}(X - \mathbb{E}X)^2 = \frac{1}{\mu^2}.$$

We start with some properties of $\min\{X,Y\}$ if both X and Y are exponentially distributed (with parameters λ and μ) and independent:

$$\mathbb{P}(\min\{X,Y\} \leq t) = 1 - \mathbb{P}(\min\{X,Y\} > t) = 1 - \mathbb{P}(X > t, Y > t) =$$

$$1 - \mathbb{P}(X > t)\mathbb{P}(Y > t) = 1 - e^{-\lambda t}e^{-\mu t} = 1 - e^{-(\lambda+\mu)t}.$$

Thus $\min\{X,Y\}$ is again exponentially distributed with as rate the sum of the rate. Repeating this argument shows that the minimum of any number of exponentially distributed random variables has again an exponential distribution. We also have:

$$\mathbb{P}(X \leq Y | \min\{X,Y\} \geq t) = \frac{\mathbb{P}(X \leq Y, \min\{X,Y\} \geq t)}{\mathbb{P}(\min\{X,Y\} \geq t)} =$$

$$\frac{\mathbb{P}(X \leq Y, X \geq t, Y \geq t)}{\mathbb{P}(X \geq t, Y \geq t)} = \frac{\mathbb{P}(X \leq Y, X \geq t)}{\mathbb{P}(X \geq t)\mathbb{P}(Y \geq t)} = \frac{\int_t^\infty \int_x^\infty \lambda e^{-\lambda x} \mu e^{-\mu y} dy dx}{e^{-\lambda t}e^{-\mu t}} =$$

$$\frac{\int_t^\infty \lambda e^{-\lambda x} e^{-\mu x} dx}{e^{-\lambda t}e^{-\mu t}} = \frac{\frac{\lambda}{\lambda+\mu} e^{-\lambda t}e^{-\mu t} dx}{e^{-\lambda t}e^{-\mu t}} = \frac{\lambda}{\lambda+\mu}.$$

This means that the probability that the minimum is attained by X in $\min\{X,Y\}$ is proportional to the rate of X, independent of the value of $\min\{X,Y\}$.

A final extremely important property of the exponential distribution is the fact that it is memoryless:

$$\mathbb{P}(X \leq t+s | X > t) = \frac{\mathbb{P}(X \leq t+s, X > t)}{\mathbb{P}(X > t)} = \frac{\mathbb{P}(X \leq t+s) - \mathbb{P}(X \leq t)}{e^{-\lambda t}} =$$

$$\frac{e^{-\lambda t} - e^{-\lambda(t+s)}}{e^{-\lambda t}} = 1 - e^{-\lambda s} = \mathbb{P}(X \leq s).$$

We continue with characterizing the Poisson process with rate λ. A Poisson process is a counting process on $\mathbb{R}_+$, meaning that it "counts" events. The Poisson process is commonly defined by taking $N(s,t)$, the number of events in $[s,t]$, equal to a Poisson distribution with parameter $\lambda(t-s)$:

$$\mathbb{P}(N(s,t) = k) = e^{-\lambda(t-s)} \frac{(\lambda(t-s))^k}{k!}.$$

Next to that we assume that the numbers of arrivals in disjunct intervals are stochastically independent.

One of its many equivalent characterizations is by the interevent times, which are independent and exponentially distributed with parameter λ. That this is equivalent can be seen by looking at the probability that there are no events in $[s,t]$:

$$\mathbb{P}(\text{next event after } s+t|\text{event at } s) = \mathbb{P}(N(s,s+t)=0) = e^{-\lambda t}.$$

Thus the time until the kth event has as distribution a sum of exponentially distributed random variables, which is commonly known as Gamma or Erlang distribution with shape parameter k and scale parameter λ.

Note that, thanks to the properties of the exponential distribution, the superposition of two Poisson processes is again a Poisson process with as rate the sum of the rates.

Finally a few words on conditioning. Let A and B be events in some probability space. Then $\mathbb{P}(A|B)$, the probability of A given B, is defined as

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(AB)}{\mathbb{P}(B)}.$$

Now $\mathbb{P}(A) = \mathbb{P}(AB) + \mathbb{P}(AB^c) = \mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|B^c)\mathbb{P}(B^c)$. This is called the *law of total probability*. It can be generalized as follows: let $B_1, B_2, \ldots$ be events such that $B_i \cap B_j = \emptyset$, and $\cup_{k=1}^{\infty} B_k \supset A$. Then

$$\mathbb{P}(A) = \sum_{k=1}^{\infty} \mathbb{P}(A|B_k)\mathbb{P}(B_k).$$

Exercise 2.1 (**thinning Poisson processes**) Consider a Poisson process with rate λ. Construct a new point process by selecting independently each point of the Poisson process with the same probability p.
a. Show that the interarrival times of the new process are exponentially distributed, and give the parameter (hint: use the law of total probability).
b. Prove that the new process is again a Poisson process (check all conditions!).

Exercise 2.2 Let X and Y be i.i.d. exponentially(λ) distributed.
a. Compute $\mathbb{P}(X \leq t, X+Y > t)$.
b. Explain why $\mathbb{P}(N(0,t)=1) = \mathbb{P}(X \leq t, X+Y > t)$ with N as above.
c. Verify the answer of a) using the Poisson distribution.

3 Markov chains: forward recursion

A Markov chain is a dynamic process taking values in its state space $\mathcal{X}$. From one time to the next it changes state occurding to its transition probabilities $p(x,y) \geq 0$, $x,y \in \mathcal{X}$, $\sum_{y \in \mathcal{X}} p(x,y) = 1$. Let the r.v. X_t be the state at t, with distribution π_t. Then for $t > 0$ $\pi_t(x) = \sum_y \pi_{t-1}(y)p(y,x)$,

π_0 is given. Thus $p(x,y)$ should be interpreted as the probability that the chain moves to state y given it is in state x.

We will make the following three assumptions. Relaxing any of these is possible, but usually leads to additional constraints or complications. Moreover, in most practical situations all three constraints are satisfied. Before formulating the assumptions it is convenient to define the notion of a path in a Markov chain.

Definition 3.1 *(path) A sequence of states $z_0, z_1, \ldots, z_{k-1}, z_k \in \mathcal{X}$ with the property that $p(z_0,z_1), \ldots, p(z_{k-1},z_k) > 0$ is called a* path *from z_0 to z_k of length k.*

Assumption 3.2 $|\mathcal{X}| < \infty$.

Assumption 3.3 *There is at least one state $x \in \mathcal{X}$, such that there is a path from any state to x. If this is the case we call the chain unichain, state x is called recurrent.*

Assumption 3.4 *The gcd of all paths from x to x is 1, for some recurrent state x. If this is the case we call the chain aperiodic.*

Define the matrix P as follows: $P_{xy} = p(x,y)$. Then $\pi_t = \pi_{t-1}P$, and it follows immediately that $\pi_t = \pi_0 P^t$. For this reason we call $p^t(x,y) = P^t_{xy}$ the t-step transition probabilities.

Theorem 3.5 *Under Assumptions 3.2–3.4, and for π_0 some arbitrary distribution, $\lim_{t\to\infty} \pi_t = \pi_*$, with the distribution π_* the unique solution of*

$$\pi_* = \pi_* P, \tag{1}$$

independent of π_0.

Theorem 3.5 gives an iterative procedure to compute π_*, by multiplying some initial distribution repetitively with P. We will call this method *forward recurrence*.

Note that π_* is the limiting distribution, but also the "stationary" distribution: if $\pi_0 = \pi_*$, then $\pi_t = \pi_*$ for all t.

Writing out the matrix equation $\pi_* = \pi_* P$ gives $\pi_*(x) = \sum_y \pi_*(y) p(y,x)$. The right-hand side is the probability that, starting from stationarity, the chain is in x the next time instant. Note that there $|\mathcal{X}|$ equations, but as this system is dependent, there is no unique solution. The solution is unique if the equation $\sum_x \pi_*(x) = 1$ is added.

If we skip one of the assumptions, then Theorem 3.5 does not hold anymore. We give three examples, each one violating one of the assumptions.

Example 3.6 (necessity Assumption 3.2) Let $\mathcal{X} = \mathbb{N}_0$, $\pi_0(0) = 1$, $p(0,1) = 1$ and $p(x,x+1) = p(x,x-1) = 1/2$ for all $x > 0$. Any solution to Equation (1) has $\pi_0(x+1) = \pi_0(x)$. This, combined with the countable state space, leads to the fact that the normalizing assumption cannot be satisfied.

Example 3.7 (necessity Assumption 3.3) Let $\mathcal{X} = \{0,1\}$ and $p(x,x) = 1$. There is no path from 0 to 1 or vice versa. It holds that $\pi_* = \pi_t = \pi_0$, and thus the limiting distribution depends on π_0.

Example 3.8 (necessity Assumption 3.4) Let $\mathcal{X} = \{0,1\}$ and $p(0,1) = p(1,0) = 1$. Then all paths from 0 to 0 and from 1 to 1 have even length, the minimum being 2: the chain is periodic with period 2. We find $\pi_{2t} = \pi_0$ and $\pi_{2t+1}(x) = 1 - \pi_0(x)$. There is no limiting distribution, unless $\pi_0(0) = 1/2$.

Forward recurrence can be interpreted as the simultaneous 'simulation' of all possible paths of the Markov chain. Compare this with regular simulation where only one path of the Markov chain is generated. From this single path the stationary probabilities can also be obtained: $T^{-1}\sum_{t=0}^{T-1}\mathbb{I}\{X_t = x\} \to \pi_*(x)$ a.s., because of the law of large numbers. Thus with probability 1 every path of the Markov chain visits a state x a fraction of times that converges to $\pi_*(x)$ in the long run. The stationary or limiting distribution π_* can thus also be interpreted as the long-run average distribution.

Exercise 3.1 Consider a Markov chain with $\mathcal{X} = \{1,2,3,4\}$,

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix},$$

and $\pi_0 = (1,0,0,0)$.
a. Compute by hand π_t for $t \leq 6$.
b. Compute using a suitable tool (for example Maple or Excel) π_t for $t = 10$, 20 and 30.
c. Compute by hand π_*.

4 Markov reward chains: the Poisson equation

Quite often we have direct rewards attached to states. Then we are interested in the limiting expected rewards.

Now we have, next to $\mathcal{X}$ and $p(x,y)$, $r(x) \in \mathbb{R}$, the direct reward that is obtained each time state x is visited. Thus, instead of the distribution of X_t, we are interested in $\mathbb{E}r(X_t)$, and especially in its limit for $t \to \infty$. This number g is given by $g = \sum_x \pi_*(x)r(x)$. Thus g can be obtained by computing π_* first and then taking the expectation of r.

An alternative is simulating X_t, and then computing $\sum_{t=0}^{T-1} r(X_t)/T \to g$ a.s.

It is not possible to generalize this method to include actions: actions depend on future behavior, and under simulation/forward recursion only the history is known.

Thus a 'backward recursion' method is needed for optimization, one that already takes the (possible) future(s) into account. We need some notation to develop this crucial idea.

Let $V_T(x)$ be the total expected reward in $0,\ldots,T-1$ when starting at 0 in x: $V_T(x) = \sum_{t=0}^{T-1}\sum_{y\in\mathcal{X}} p^t(x,y)r(y) = \mathbb{E}\sum_{t=0}^{T-1} r(X_t)$ with $X_0 = x$. Note that $\sum_x \pi_*(x)V_T(x) = \sum_x \pi_*(x)\sum_{t=0}^{T-1}\sum_y p^t(x,y)r(y) = \sum_{t=0}^{T-1}\sum_y\sum_x \pi_*(x)p^t(x,y)r(y) = \sum_{t=0}^{T-1}\sum_y \pi_*(y)r(y) = gT$. Let $V(x) = \lim_{T\to\infty}[V_T(x) - gT]$. Then $V(x)$ is the total expected difference in reward between starting in x and starting in stationarity.

We calculate $V_{T+1}(x)$ in two different ways. $V_{T+1}(x) = V_T(x) + \sum_y \pi_T(y) r(y)$ for π_0 with $\pi_0(x) = 1$. As $\pi_T \to \pi_*$ and $\sum_x \pi_*(x) r(x) = g$, $V_{T+1}(x) = V_T(x) + g + o(1)$, where $o(1)$ means that this term disappears if $t \to \infty$. On the other hand, for V_{T+1} the following recursive formulation exists:

$$V_{T+1}(x) = r(x) + \sum_y p(x,y) V_T(y). \tag{2}$$

Thus

$$V_T(x) + g + o(1) = r(x) + \sum_y p(x,y) V_T(y).$$

Subtract gT from both sides, and take $T \to \infty$:

$$V(x) + g = r(x) + \sum_y p(x,y) V(y). \tag{3}$$

This equation is also known as the Poisson equation. Note that V represents the information on the future.

Note however that Equation (3) does not have a unique solution: If V is a solution, then so is $V'(x) = V(x) + C$. There are two possible solutions: either take $V(0) = 0$ for some "reference" state 0, or add the additional condition $\sum_x \pi_*(x) V(x) = 0$. Only under the latter condition V has the interpretation as the total expected difference in reward between starting in a state and starting in stationarity.

Exercise 4.1 Consider the Markov chain from Exercise 3.1, and take $r = (0,0,0,1)$.
a. Find g using the results obtained for Exercise 3.1.
b. Argue why $\lim_{T\to\infty} V_T(x)/T = g$.
c. Compute using a suitable computer tool V_T for $T = 10, 20, 50$, and 100. Compute also $V_T - gT$ and V_T/T.
d. Find g by solving the Poisson equation. Give also all solutions for V and also the one with $\sum_x \pi_*(x) V(x) = 0$.

5 Markov decision chains: policy iteration

Finally we introduce decisions. Next to the state space $\mathcal{X}$ we have an action set $\mathcal{A}$. The idea is that depending on the state X_t an action A_t is selected, according to some policy $R : \mathcal{X} \to \mathcal{A}$. Thus $A_t = R(X_t)$.

Evidently the transition probabilities also depend on the action: $p(x,a,y)$ is the probability of going from x to y when a is chosen. Also the rewards depend on the actions: $r(x,a)$.

Assumption 5.1 $|\mathcal{A}| < \infty$.

We also have to adapt the assumptions we made earlier. Assumption 3.2, which states that $|\mathcal{X}| < \infty$, remains unchanged. For the other two assumption we have to add that they should hold for any policy R.

Assumption 5.2 *For every policy R there is at least one state $x \in \mathcal{X}$ (that may depend on R), such that there is a path from any state to x. If this is the case we call the chain unichain, state x is called recurrent.*

Assumption 5.3 *For every policy R the gcd of all paths from x to x is 1, for some recurrent state x. If this is the case we call the chain aperiodic.*

Let $V_t^R(x)$ be the total expected reward in $0,\ldots,t-1$, when starting at 0 in x, under policy R. We are interested in finding $\arg\max_R \lim_{T\to\infty} V_T^R(x)/T$, the maximal average expected long-run reward. This maximum is well defined because the number of different policies is equal to $|\mathcal{X}||\mathcal{A}|$, and thus finite.

How to compute the optimal policy?
1. Take some R.
2. Compute g^R and $V^R(x)$ for all x.
3. Find a **better** R'. If none exists: stop.
4. $R := R'$ and go to step 2.

This algorithm is called *policy iteration.* Step 2 can be done using the Poisson equation: for a fixed policy R the direct rewards are $r(x,R(x))$ and the transition probabilities are $p(x,R(x),y)$. How to do step 3? Take

$$R'(x) = \arg\max_a \{r(x,a) + \sum_y p(x,a,y)V^R(y)\}$$

in each state. If the maximum is attained by R for each x then no improvement is possible.

Example 5.4 (Replacement decisions) Suppose we have a system that is subject to wear-out, for example a car. Every year we have to pay maintenance costs, that are increasing in the age of the system. Every year we have the option to replace the system at the end of the year by a new one. After N years we are obliged to replace the system. Thus $\mathcal{X} = \{1,\ldots,N\}$, $\mathcal{A} = \{1,2\}$, action 1 meaning no replacement, action 2 replacement. Thus $p(x,1,x+1) = 1$ for $x < N$ and $p(x,2,1) = p(N,1,1) = 1$ for all x. The rewards are given by $r(x,1) = -C(x)$ for $x < N$ and $r(x,2) = r(N,1) = -C(x) - P$ for all x, with P the price of a new system. Consider a policy R defined by $R(1) = 1$ and $R(x) = 2$ for $x > 1$, thus we replace the system if its age is two years or higher. The Poisson equations are as follows:

$$V^R(1) + g^R = -C(1) + V^R(2), \quad V^R(x) + g^R = -C(x) - P + V^R(1) \text{ for } x > 1.$$

We take $V(1) = 0$. Then the solution is $g^R = \frac{-C(1)-C(2)-P}{2}$ and $V^R(x) = -g^R - C(x) - P$, $x > 1$. Next we do the policy improvement step, giving

$$R'(x) = \arg\max\{-C(x) + V^R(x+1), -C(x) - P + V^R(1)\} = \arg\max\{-g^R - C(x+1), 0\} \text{ for } x < N.$$

If we assume that C is increasing, then R' is 1 up to some x and 2 above it.

Take, for example, $C(x) = x$ and $P = 10$. Then $g^R = -6.5$ and $V^R(x) = -3.5 - x$, $x > 1$. From this it follows that $R'(1) = \cdots = R'(5) = 1$, $R'(6) = \cdots = R'(N) = 2$, with average reward $g^{R'} = (-1-2-3-4-5-6-10)/6 \approx -5.17 > -6.5 = g^R$. Note that there are two optimal policies, that replace after 4 or 5 years, with average cost 5.

For the optimal policy R^* it holds that

$$r(x,R^*(x))) + \sum_y p(x,R^*(x),y)V^{R^*}(y) = \max_a \{r(x,a) + \sum_y p(x,a,y)V^{R^*}(y)\}.$$

At the same time, by the Poisson equation:

$$V^{R^*}(x) + g^{R^*} = r(x,R^*(x))) + \sum_y p(x,R^*(x),y)V^{R^*}(y).$$

Combining these two gives

$$V^{R^*}(x) + g^{R^*} = \max_a \{r(x,a) + \sum_y p(x,a,y)V^{R^*}(y)\}.$$

This equation is called the *optimality equation* or *Bellman equation*. Often the superscript is left out: g and V are simply the average reward and value function of the optimal policy.

Exercise 5.1 Consider Example 5.4, with $N = 4$, $P = 3$, $C(1) = 5$, $C(2) = 10$, $C(3) = 0$ and $C(4) = 10$. Start with $R(x) = 2$ for all x. Apply policy iteration to obtain the optimal policy.

6 Markov reward chains: backward recursion

In this section we go back to the Markov reward chains to obtain an alternative method for deriving V. Recall that $V_{T+1}(x) - V_T(x) \to g$, and note that $V_T(x) - V_T(y) \to V(x) - V(y)$. Thus simply by computing V_T for T big we can obtain all values we are interested in. To compute V_T we can use the recursion (2). Initially one usually takes $V_0 = 0$, although a good initial value can improve the performance significantly. One stops iterating if the following holds:

$$\text{span}(V_{t+1}(x) - V_t(x)) \leq \varepsilon \text{ for all } x \in \mathcal{X},$$

which is equivalent to

$$\text{there exists a } g \text{ such that } g - \frac{\varepsilon}{2} \leq V_{t+1}(x) - V_t(x) \leq g + \frac{\varepsilon}{2}$$

for all $x \in \mathcal{X}$.

Value iteration algorithm pseudo code Let $|\mathcal{X}| = N$, $E(x) \supset \{y | p(x,y) > 0\}$, and ε some small number (e.g., 10^{-6}).

Vector $V[1..N]$, $V'[1..N]$
Float *min*, *max*
$V \leftarrow 0$
do {
  $V' \leftarrow V$

for$(x = 1,..,N)$ { % iterate
$V(x) \leftarrow r(x)$
for$(y \in E(x))$ $V(x) \leftarrow V(x) + p(x,y)V'(y)$ }
$max \leftarrow -10^{10}$
$min \leftarrow 10^{10}$
for$(x = 1,..,N)$ { % compute span$(V - V')$
if$(V(x) - V'(x) < min)$ $min \leftarrow V(x) - V'(x)$
if$(V(x) - V'(x) > max)$ $max \leftarrow V(x) - V'(x)$ } }
while$(max - min > \varepsilon)$

Some errors to avoid:
- Avoid taking $E(x) = \mathcal{X}$, but use the sparseness of the transition matrix P;
- Compute P online, instead of calculating all entries of P first;
- Insert the code for calculating P at the spot, do not use a function or subroutine;
- Span$(V - V')$ need not be calculated at every iteration, but once every 10th iteration for example.

7 Markov decision chains: backward recursion

Value iteration works again in the situation of a Markov decision chain, by including actions in the recursion of Equation (2):

$$V_{t+1}(x) = \max_a \{r(x,a) + \sum_y p(x,a,y)V_t(y)\}. \tag{4}$$

We use the same stop criterion as for the Markov reward case.

Remark 7.1 (terminology) The resulting algorithm is known under several different names. The same holds for that part of mathematics that studies stochastic dynamic decision problems. The backward recursion method is best known under the name value iteration, which stresses the link with policy iteration. The field is known as Markov decision theory, although stochastic dynamic programming is also used. Note that in a deterministic setting dynamic programming can best be described by backward recursion. Thus the field is identified by its main solution method. We will call the field Markov decision theory, and we will mainly use value iteration for the backward recursion method.

Remark 7.2 $V_t(x)$ is interesting by itself, not just because it helps in finding the long-run average optimal policy: it is the maximal reward over an horizon t. For studying these finite-horizon rewards we do not need Assumptions 3.3–3.4: they were only necessary to obtain limit results.

Example 7.3 Consider a graph with nodes $\mathcal{V} = \{1,\ldots,N\}$, arc set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and distances $d(x,y) > 0$ for all $(x,y) \in \mathcal{E}$. What is the shortest path from 1 to N? This can be solved using backward recursion as follows. Take $\mathcal{X} = \mathcal{A} = \mathcal{V}$. For all $x < N$ we define $r(x,a) = -d(x,a)$ if $(x,a) \in \mathcal{E}$, $-\infty$ otherwise, and $p(x,a,a) = 1$. Define also $r(N,a) = 0$ and $p(N,a,N) = 1$. Start with $V_0 = 0$. Then V_{N-1} gives the minimal distances from all points to N.

Exercise 7.1 Prove the claim of Example 7.3.

Exercise 7.2 Consider again Example 7.3.
a. Formulate the Poisson equation for the general shortest path problem.
b. Give an intuitive explanation why $V(N) = 0$.
c. Give an interpretation of the values $V(x)$ for $x \neq N$.
d. Verify this using the Poisson equation.

Exercise 7.3 Consider a model with $\mathcal{X} = \{0,\ldots,N\}$, $\mathcal{A} = \{0,1\}$, $p(x,0,x) = p(N,1,N) = \lambda$ for all $x > 0$, $p(x,1,x+1) = \lambda$ for all $x < N$, $p(0,0,0) = 1$, $p(0,1,0) = p(x,a,x-1) = 1-\lambda$ for all $a \in \mathcal{A}$ and $x > 0$, $r(x,0) = x - c$ for some $c > 0$, $r(N,1) = N - c$, and $r(x,1) = x$ for $x < N$. Implement the value iteration algorithm in some suitable programming environment and report for different choices of the parameters (with N at least 10) on the optimal policy and values.

8 Continuous time: semi-Markov processes

Consider a Markov chain where the time that it takes to move from a state x to the next state is not equal to 1 anymore, but some random variable $T(x)$. This is called a semi-Markov process (Çinlar [6], Ch. 10). We assume that $0 < \tau(x) = \mathbb{E}T(x) < \infty$.

If we study the semi-Markov process only at the moments it changes state, the jump times, then we see what is called the embedded Markov chain. This Markov chain has stationary distribution π_*. Consider now the stationary distribution over time, i.e., the time-limiting distribution that the chain is in a certain state. This distribution ν_* is specified by:

$$\frac{\nu_*(x)}{\nu_*(y)} = \frac{\tau(x)\pi_*(x)}{\tau(y)\pi_*(y)},$$

from which it follows that

$$\nu_*(x) = \frac{\pi_*(x)\tau(x)}{\sum_y \pi_*(y)\tau(y)}. \tag{5}$$

Example 8.1 (Repair process) Take a model with $\mathcal{X} = \{0,1\}$, $p(0,1) = p(1,0) = 1$, and some arbitrary $T(0)$ and $T(1)$. This could well model the repair of a system, with $T(1)$ the time until failure, and $T(0)$ the repair time. Note that the embedded Markov chain is periodic, but the stationary distribution exists: $\pi_*(0) = \pi_*(1) = \frac{1}{2}$. Using Equation (5) we find

$$\mathbb{P}(\text{system up in long run}) = \nu_*(1) = \frac{\pi_*(1)\tau(1)}{\pi_*(0)\tau(0)+\pi_*(1)\tau(1)} = \frac{\tau(1)}{\tau(0)+\tau(1)}.$$

The same result can be obtained from renewal theory.

Exercise 8.1 Calculate π_0, π_1, π_2, π_*, and ν_* for the following semi-Markov process: A machine can be in three states: in perfect condition, deteriorated, or failed. Repair takes 5 hours, it stays on average 4 days in perfect condition, and 2 hours in a deteriorated condition. After repair the condition is perfect, from the perfect state it breaks down completely in 60% of the cases, in 40% of the cases it continues working in the deteriorated state. It starts in perfect condition.

9 Markov processes

A special type of semi-Markov process is the one where all $T(x)$ are exponentially distributed. Then it is called a Markov process. However, usually a Markov process is defined by its transition rates $\lambda(x,y)$. Thus the time until the process moves from x to y is exponentially distributed, unless another transition to another state occurs first. Thus, by standard properties of the exponential distribution, $T(x)$ is exponentially distributed with rate $\sum_y \lambda(x,y)$, and $p(x,y) = \lambda(x,y)/\sum_z \lambda(x,z)$. Thus, Markov processes, defined through their rates $\lambda(x,y)$, are indeed special cases of semi-Markov processes.

Sometimes it is convenient to have $T(x)$ equally distributed for all x. Let γ be such that $\sum_y \lambda(x,y) \leq \gamma$ for all x. We construct a new process with rates $\lambda'(x,y)$ as follows. First, take $\lambda'(x,y) = \lambda(x,y)$ for all $x \neq y$. In each state x with $\sum_y \lambda(x,y) < \gamma$, add a 'fictituous' or 'dummy' transition from x to x such that the rates sum up to γ: $\lambda'(x,x) = \gamma - \sum_{y \neq x} \lambda(x,y)$ for all $x \in \mathcal{X}$. This new process has expected transition times τ' as follows: $\tau'(x) = 1/\gamma$. Because $\tau'(x) = \tau'(y)$ it follows that $\pi'_* = \nu'_*$, from Equation (5). The idea of adding dummy transitions to make the rates out of states constant is called *uniformization*.

Using uniformization we can derive the standard balance equations for Markov processes from Equation (1). To do so, let us write out Equation (1), in terms of the rates. Note that for the transition probabilities we have $p'(x,y) = \lambda'(x,y)/\gamma$:

$$\sum_{y \in \mathcal{X}} \frac{\lambda'(x,y)}{\gamma} \nu'_*(x) = \nu'_*(x) = \sum_{y \in \mathcal{X}} \nu'_*(y) p'(y,x) = \sum_{y \neq x} \nu'_*(y) \frac{\lambda'(y,x)}{\gamma} + \nu'_*(x) \frac{\lambda'(x,x)}{\gamma}.$$

Multiplying by γ, and subtracting $\nu'_*(x)\lambda'(x,x)$ from both sides leads to

$$\sum_{y \neq x} \lambda'(x,y) \nu'_*(x) = \sum_{y \neq x} \nu'_*(y) \lambda'(y,x), \tag{6}$$

the standard balance equations.

Example 9.1 (M/M/1 queue) An M/M/1 queue has $\mathcal{X} = \mathbb{N}_0$, and is defined by its arrival rate λ and its service rate μ, thus $\lambda(x,x+1) = \lambda$ for all $x \geq 0$ and $\lambda(x,x-1) = \mu$ for all $x > 0$. All other transition rates are 0. (We assume that $\lambda < \mu$ for reasons to be explained later.) Filling in the balance equations (6) leads to

$$\lambda \nu_*(0) = \mu \nu_*(1), \quad (\lambda + \mu)\nu_*(x) = \lambda \nu_*(x-1) + \mu \nu_*(x+1),\ x > 0.$$

It is easily verified that the solution is $\nu_*(x) = (1-\rho)\rho^x$, with $\rho = \lambda/\mu$. Note that Assumption 3.2 is violated. For ν_* to be a distribution we had to assume that $\lambda < \mu$.

We could have used Equation (5) right away. Note that $\tau(0) = 1/\lambda$, $\tau(x) = 1/(\lambda+\mu)$ for $x > 0$, and that $p(0,1) = 1$ and $p(x,x-1) = \mu/(\lambda+\mu) = 1 - p(x,x+1)$ for $x > 0$, from standard properties of the exponential distribution. This embedded chain has solution

$$\pi_*(1) = \frac{\lambda+\mu}{\mu}\pi_*(0), \quad \pi_*(x) = \frac{\lambda+\mu}{\mu}\left(\frac{\lambda}{\mu}\right)^{x-1}\pi_*(0) \text{ for } x > 0.$$

This leads to

$$\pi_*(0) = \frac{\mu-\lambda}{2\mu}, \quad \pi_*(1) = \frac{\mu-\lambda}{2\mu}\frac{\lambda+\mu}{\mu}, \quad \pi_*(x) = \frac{\mu-\lambda}{2\mu}\frac{\lambda+\mu}{\mu}\left(\frac{\lambda}{\mu}\right)^{x-1} \text{ for } x > 0.$$

The denominator of Equation (5) is equal to $1/(2\lambda)$, leading to

$$\nu_*(0) = 2\lambda\pi_*(0)\tau(0) = 2\lambda\frac{\mu-\lambda}{2\mu}\frac{1}{\lambda} = 1-\rho,$$

equal to what we found above. In a similar way we can find $\nu_*(x)$ for $x > 0$.

Exercise 9.1 Calculate ν_* for the following Markov processes:
a. A shop can handle 2 customers at a time, and has 2 additional places for waiting. Customers that find all places occupied do not enter and go elsewhere. Arrivals occur according to a Poisson process with rate 3, service times are exponential with rate 2. Take as states the number of customers that are in the shop. Initially the system is empty.
b. Consider a pool of 4 machines, each of which is either up or down. Time until failure is exponentially distributed with rate 1 for each machine, repair takes an exponentially distributed amount of time with rate 2. There is a single repairman, thus only one machine can be repaired at the same time. Take as states the number of machines that are up. Initially all machines are functioning.
c. A shop can handle 1 customer at a time, and has 3 additional places for waiting. Customers that find all places occupied do not enter and go elsewhere. Arrivals occur according to a Poisson process with rate 4, service times are exponential with rate 3. Waiting customers leave the shop unserved, each at a rate 1. Take as states the number of customers that are in the shop. Initially the system is empty.

Exercise 9.2 Consider the M/M/*s*/*s* queue, which is an *s*-server system with arrival rate λ and service rate μ, but without waiting places. It is also known as the Erlang B or Erlang blocking model.
a. What is a suitable uniformization parameter?
Formulate Equation (6) for this model.
b. Find its solution.

10 Generalized semi-Markov processes

Sometimes a semi-Markov process is not general enough. Think of discrete-event simulation: multiple events are active at the same time, and the one that 'fires' first generates a change in state. The mathematical framework for this are generalized semi-Markov processes. They make a distinction between states and events, and multiple events can be active at any time. The main solution method of generalized semi-Markov processes is simulation ("We think of a gsMp as a model of discrete-event simulation", Whitt '80); for this reason we will not go into detail and present only the G/G/*s* queue as an example.

Example 10.1 (G/G/s queue) Next to the state space $\mathcal{X} = \mathbb{N}_0$ we have a set of events $\mathcal{E} = \{0, \ldots, s\}$ of which event 0 (the arrival process) is always active, and $\min\{s, x\}$ of the remaining s events are active (the departures). For each of the active events there is a time when they will expire. When the first expires the state changes and new events might become active. E.g., when an arrival occurs (event 0) when $x < s$ then event 0 is rescheduled, but also event $x+1$ becomes active.

Remark 10.2 A semi-Markov process is a generalized semi-Markov process where only one event can be active at the same time. Also Markov processes are special cases of generalized semi-Markov processes: due to the exponentiality of the event times however they can be reformulated as semi-Markov processes allowing a treatment using backward recursion.

11 Semi-Markov reward processes: Poisson equation

As for the Markov reward processes we now add rewards to the process. In the continuous-time setting we have to decide how rewards are obtained: all at once at jump times or continuously over time. We choose the latter, in the section on modeling issues we discuss how one form of rewards can be translated into the other. Thus every time unit that the process remains in state x a reward $r(x)$ is obtained.

We are again interested in the expected long-run stationary rewards, which is equivalent to the long-run average or long-run expected rewards:

$$g = \lim_{t\to\infty} \mathbb{E} r(X_t) = \lim_{t\to\infty} \frac{1}{t} \mathbb{E} \int_0^t r(X_s) ds = \sum_x \nu_*(x) r(x).$$

Example 11.1 Take the repair process studied earlier, suppose a reward R for each unit of time the system is up and labour costs C for each unit of time the system is in repair. Then the expected long-run stationary reward is given by

$$g = \frac{-\tau(0)C + \tau(1)R}{\tau(0) + \tau(1)}.$$

A simple algorithm to compute g would consists of computing the stationary distribution of the embedded chain π_* and from that ν_* and finally g can be computed. Note that g can also be computed directly from π_*:

$$g = \sum_x \nu_*(x) r(x) = \frac{\sum_x \pi_*(x)\tau(x) r(x)}{\sum_y \pi_*(y)\tau(y)}. \tag{7}$$

The denominator of g has the following interpretation: it is the expected time between two jumps of the process.

An alternative method is to simulate the embedded chain X_t, and then to compute $\frac{\sum_{t=0}^T r(X_t)\tau(X_t)}{\sum_{t=0}^T \tau(X_t)} \to g$ a.s.

As for the discrete-time case, we move next to backward recursion. Let $V_t(x)$ be the total expected reward in $[0,t]$ when starting at 0 in x. We are again interested in $\lim_{t\to\infty} \frac{V_t(x)}{t}$, the average expected long-run rewards.

Using similar arguments as for the discrete-time case we find

$$V_T(x)+\tau(x)g+o(1)=r(x)\tau(x)+\sum_y p(x,y)V_T(y).$$

Subtracting gT from both sides and taking $T\to\infty$ leads to:

$$V(x)+\tau(x)g=r(x)\tau(x)+\sum_y p(x,y)V(y) \tag{8}$$

Note that again this equation does not have a unique solution.

Example 11.2 (Repair process) Consider again the repair process. We get the following set of equations:

$$V(0)+\tau(0)g=-C\tau(0)+V(1),\quad V(1)+\tau(1)g=R\tau(1)+V(0)$$

All solutions are given by:

$$g=\frac{-C\tau(0)+R\tau(1)}{\tau(0)+\tau(1)},$$

$$V(1)=V(0)+\frac{\tau(0)\tau(1)(R+C)}{\tau(0)+\tau(1)}$$

Exercise 11.1 We add a reward component to the semi-Markov process that we studied for some of the models of Exercises 8.1 and 9.1. Compute the expected stationary reward using two different methods: by utilizing the stationary distribution ν_*, and by solving the optimality equation. Give also expressions for V. The direct rewards are as follows:
a. Consider Exercise 8.1: There is a reward R for each time unit the machine is up. The costs for repairing are equal to C per unit of time.
b. Consider Exercise 9.1b: There is a reward R for each time unit that a machine is up. Every repair costs C per unit of time for labor costs.

12 Semi-Markov decision processes: policy improvement

Finally we consider semi-Markov decision processes. Now the transition times $T(x,a)$ depend also on the action that is taken at the beginning of the period after which the transition takes place. We define $\tau(x,a)=\mathbb{E}T(x,a)$, and continuous rewards $r(x,a)$, when action $a\in\mathcal{A}$ was chosen when reaching $x\in\mathcal{X}$ and while we're still in x.

Let $V_t^R(x)$ be the total expected reward in $[0,t]$, when starting at 0 in x, under policy R. We are again interested in $\arg\max_R \lim_{t\to\infty}\frac{V_t^R(x)}{t}$, the maximal average expected long-run rewards. Policy iteration can again be used, where Equation (8) is used to evaluate a policy and the policy improvement step consists of taking

$$R'(x)=\arg\max_a\{(r(x,a)-g^R)\tau(x,a)+\sum_y p(x,a,y)V^R(y)\}.$$

Exercise 12.1 Consider again Exercise 8.1. As in Exercise 11.1, there is a reward R for each time unit the machine is up. The costs for repairing are equal to C per unit of time. Suppose there is the option to repair in 3 hours for costs C' per unit of time.
a. Formulate this as a semi-Markov decision process.
b. Use policy iteration to determine the minimal value of C' for which it would be attractive to choose the short repair times.

13 Semi-Markov reward processes: backward recursion

To derive the backward recursion algorithm for semi-Markov reward processes we note first that the optimal policy depends only on $T(x)$ through $\tau(x)$: the distribution of $T(x)$ does not play a role. This means that we can choose $T(x)$ the way we like: with $0 < \tau \leq \tau(x)$ for all x, we take $T(x) = \tau G(x)$, where $G(x)$ has a geometric distribution with parameter $q(x) = \tau/\tau(x)$. This means that $\mathbb{P}(G(x) = 1) = q_x$, $\mathbb{P}(G(x) = 2) = (1-q(x))q(x)$, etc. Then $\sum_k k\mathbb{P}(G(x) = k) = \sum_k kq(x)(1-q(x))^{k-1} = q(x)^{-1}$, and thus indeed $\mathbb{E}T(x) = \tau\mathbb{E}G(x) = \tau(x)$.

Note that $G(x)$ is memoryless: after each interval of length τ the sojourn time in state x finishes with probability $q(x)$. Thus the original system is equivalent to one with sojourn times equal to τ and dummy transitions with probability $1-q(x)$. This leads to the following backward recursion:

$$V_{(t+1)\tau}(x) = r(x)\tau + q(x)\sum_y p(x,y)V_{t\tau}(y) + (1-q(x))V_{t\tau}(x).$$

From $V_{(t+1)\tau}(x) = V_{t\tau}(x) + \tau g + o(1)$ it follows that $V_{(t+1)\tau}(x) - V_{t\tau}(x) \to \tau g$. Thus the value iteration algorithm consists of taking $V_0 = 0$, and then computing $V_\tau, V_{2\tau}, \ldots$. The stop criterion is equivalent to the one for the discrete-time case.

14 Semi-Markov decision processes: backward recursion

Value iteration can again be generalized to the case that includes decisions. This leads to the following value function:

$$V_{(t+1)\tau}(x) = \max_a\{r(x,a)\tau + q(x,a)\sum_y p(x,a,y)V_{t\tau}(y) + (1-q(x,a))V_{t\tau}(x)\}.$$

The last policy is optimal, and $[V_{(t+1)\tau}(x) - V_{t\tau}(x)]/\tau$ for t sufficiently large gives the maximal average rewards.

Remark 14.1 In the discrete-time setting $V_t(x)$ had an interpretation: it is the total expected maximal reward in t time units. In the continuous-time case $V_{t\tau}(x)$ does not have a similar interpretation, due to the randomness of $T(x,a)$.

Exercise 14.1 Consider the repair process of Example 11.2. Take $\tau(1) = 5$, $\tau(0) = 2$, $R = 2$ and $C = 0$. Here there is also the additional option to shorten repair times to 1, for costs 1.

a. Formulate the value function.
b. Solve it using a suitable computer program of package.
c. Find the optimal policy as a function of the parameter t. Can you explain what you found?

15 Other criterion: discounted rewards

Average rewards are often used, but sometimes there are good reasons to give a lower value to a reward obtained in the future than the same reward right now.

Example 15.1 A reward of 1 currently incurred will be $1+\beta$ after one year if put on a bank account with an interest rate of β. Thus a reward of 1 after 1 year values less than a reward of 1 right now.

In the example we considered a yearly payment of interest of rate β. To make the step to continuous-time models, we assume that each year is divided in m periods, and after each period we received an interest of β/m. Thus after t years our initial amount 1 has grown to $(1+\frac{\beta}{m})^{tm}$. This converges to $e^{\beta t}$ as $m \to \infty$. Thus, in a continuous-time model with interest β, an amount of 1 values $e^{\beta t}$ after t years. By dividing by $e^{\beta t}$ we also obtain: Reward 1 at 1 is evaluated at 0 as $e^{-\beta}$. This generalizes to reward 1 at t is evaluated at 0 as $e^{-\beta t}$.

Because $\int_0^\infty e^{-\beta t}dt < \infty$ the total expected discounted rewards are well defined. Let $V_\beta(x)$ be the total expected discounted rewards, starting in x. Note that the starting state is crucial here, unlike for the average reward model.

To determine $V_\beta(x)$, we first have to derive the total expected discounted rate rewards if the model is in x from 0 to $T(x)$. If $r(x)=1$, then this is equal to

$$\mathbb{E}\int_0^{T(x)} e^{-\beta s}ds.$$

We write $\mathbb{E}f(T(x)) = \int_0^\infty f(t)dT(x)(t)$, irrespective of the type of distribution of $T(x)$. (This notation comes from measure theory.) Then

$$\mathbb{E}\int_0^{T(x)} e^{-\beta s}ds = \int_0^\infty \int_0^t e^{-\beta s}dsdT(x)(t) = \beta^{-1}(1-\gamma(x)),$$

with $\gamma(x) = \mathbb{E}e^{-\beta T(x)}$, the so-called *Laplace-Stieltjes transform* of $T(x)$ in β. From $T(x)$ on the discounted rewards are equal to $V_\beta(y)$ with probability $p(x,y)$, but discounted with $\mathbb{E}e^{-\beta T(x)} = \gamma(x)$. Thus the Poisson equation (8) has the following discounted equivalent:

$$V_\beta(x) = \beta^{-1}(1-\gamma(x))r(x) + \gamma(x)\sum_y p(x,y)V_\beta(y).$$

This can of course be utilized as part of a policy improvement algorithm. The improvement step is then given by:

$$R'(x) = \arg\max_a\{\beta^{-1}(1-\gamma(x,a))r(x,a) + \gamma(x,a)\sum_y p(x,a,y)V_\beta^R(y)\}.$$

The optimality equation becomes

$$V_\beta(x) = \max_a \{\beta^{-1}(1-\gamma(x,a))r(x,a) + \gamma(x,a)\sum_y p(x,a,y)V_\beta(y)\},$$

and also value iteration works for the discounted model.

Remark 15.2 It is interesting to note that discounting is equivalent to taking total rewards up to T with T random and exponential. Indeed, let $r(t)$ be a function indicating the rate reward at t. Let $T \sim \exp(\beta)$. Then $c(t)e^{-\beta t}$ is the expected reward at t, equal to the discounted reward.

Exercise 15.1 Determine the Laplace-Stieltjes tranforms of the exponential and gamma distributions.

Exercise 15.2 Repeat exercise 12.1 for the discounted reward case. Consider the cases in which the transition times are constant and exponentially distributed, for some well-chosen β.

Exercise 15.3 Consider a Markov reward process with state space $\mathcal{X} = \{0,1\}$, $p(0,1) = p(1,0) = 1$, T_0 is exponentially distributed, T_1 is constant, and $r = (1,0)$. Assume that the reward at t is discounted with a factor $e^{-\beta t}$ for some $\beta > 0$. Let $V_\beta(x)$ be the long-run expected discounted reward for initial state x.
a. Give a set of equations for V_β and solve it.
b. Compute $\lim_{\beta\to 0} \beta V_\beta$.
c. Give an interpretation for the results you found for b.

16 (Semi-)Markov decision processes: literature

Some literature on the theory of (semi-)Markov decision chains/processes: Bertsekas [3], Kallenberg [10], Puterman [13], Ross [14], Tijms [16].

17 Modeling issues

Suppose you have some system or model that requires dynamic optimization. Can it be (re)formulated as a (semi-)Markov decision problem, and if so, how to do this in the best way? Different aspects of this question will be answered under the heading 'modeling issues'.

Modeling issues: dependence on next state

Let us start by introducing some simple generalizations to the models that can sometimes be quite helpful.

Sometimes it is more appropriate to work with costs instead or rewards. This is completely equivalent, by multiplying all rewards with -1 and replacing max by min everywhere.

It sometimes occurs that the direct rewards r' depend also on the next state: $r'(X_t, A_t, X_{t+1})$. We are interested in the sum of the expected rewards, $\mathbb{E}\sum_{t=0}^{T-1} r'(X_t, A_t, X_{t+1})$, which gives

$$\mathbb{E}\sum_{t=0}^{T-1} r'(X_t, A_t, X_{t+1}) = \sum_{t=0}^{T-1} \mathbb{E} r'(X_t, A_t, X_{t+1}) = \mathbb{E}\sum_{t=0}^{T-1}\sum_{y\in\mathcal{X}} p(X_t, A_t, y) r'(X_t, A_t, y).$$

Thus we can replace the direct rewards $r'(x,a,y)$ by $r(x,a) = \sum_y p(x,a,y) r'(x,a,y)$, which fits within our framework.

A similar reasoning can be applied to the case where the transitions times depend also on y, notation $T'(x,a,y)$ with $\tau'(x,a,y) = \mathbb{E}T'(x,a,y)$. In this case, take $\tau(x,a) = \sum_y p(x,a,y)\tau'(x,a,y)$.

Modeling issues: lump rewards

We presented the theory for continuous-time models assuming that rewards are obtained in a continuous fashion, so-called rate rewards. Sometimes rewards are obtained in a discrete fashion, once you enter or leave a state (and after having chosen an action): *lump rewards*. In the long run lump rewards $r_l(x,a)$ are equivalent to rewards $r_l(x,a)/\tau(x,a)$. If we write out Equation (7) for lump rewards instead of rate rewards, then we get the following somewhat simpler expression:

$$g = \frac{\sum_x \pi_*(x) r_l(x)}{\sum_y \pi_*(y)\tau(y)}.$$

Note that now also the numerator has a simple interpretation: it is the expected lump reward per jump of the process.

Modeling issues: aperiodicity

Forward and backward recursion do not need to converge in the case of Markov (decision) chains that are periodic. This we illustrate with an example.

Example 17.1 Consider a Markov reward chain with $\mathcal{X} = \{0,1\}$, $r = (1,0)$, $p(0,1) = p(1,0) = 1$. This chain is periodic with period 2. If we apply value iteration, then we get:

$$V_{n+1}(0) = 1 + V_n(1), \quad V_{n+1}(1) = 1 + V_n(0).$$

Take $V_0(0) = V_0(1) = 0$. Then

$$V_n(0) = \begin{cases} \frac{n}{2} & \text{if } n \text{ even;} \\ \frac{n+1}{2} & \text{if } n \text{ odd,} \end{cases} \quad V_n(1) = \begin{cases} \frac{n}{2} & \text{if } n \text{ even;} \\ \frac{n-1}{2} & \text{if } n \text{ odd.} \end{cases}$$

From this it follows that

$$V_{n+1}(0) - V_n(0) = \begin{cases} 1 & \text{if } n \text{ even;} \\ 0 & \text{if } n \text{ odd,} \end{cases} \quad V_{n+1}(1) - V_n(1) = \begin{cases} 0 & \text{if } n \text{ even;} \\ 1 & \text{if } n \text{ odd.} \end{cases}$$

But in this case the stop criterion is never met: $\text{span}(V_{n+1} - V_n) = 1$ for all n.

A simple trick to avoid this problem is to introduce a so-called *aperiodicity transformation*. It consists in fact of adding a dummy transition to each state, by replacing P by $\delta P + (1-\delta)I$, $0 < \delta < 1$. Thus with probability $1-\delta$ the process stays in the current state, irrespective of the state and action; with probability δ the transition occurs according to the original transition probabilities (which could also mean a transition to the current state). Because it is now possible to stay in every state, the current chain is aperiodic, and forward and backward recursion converge. This model has the following Poisson equation:

$$V + g = r + (\delta P + (1-\delta)I)V = r + P\delta V + (1-\delta)V \Leftrightarrow \delta V + g = r + P\delta V. \tag{9}$$

If (V,g) is a solution of $V + g = r + PV$, then $(V/\delta, g)$ is a solution of Equation (9). Thus the average rewards remain the same, and V is multiplied by a constant. This is intuitively clear, because introducing the aperiodicity transformation can be interpreted as slowing down the system, making it longer for the process to reach stationarity.

Modeling issues: states

The first and perhaps most important choice when modeling is how to choose the states of the model. When the states are chosen in the wrong way, then sometimes the model cannot be put into our framework. This fact is related to the Markov property, which we discuss next.

Definition 17.2 *(Markov property) A Markov decision chain has the Markov property if* $\mathbb{P}(X_{t+1} = x | X_t = x_t, A_t = a_t) = \mathbb{P}(X_{t+1} = x | X_s = a_s, A_s = a_s,\ s = 1,\ldots,t)$ *for all* t, x, *and* x_s, a_s *for* $s = 1,\ldots,t$.

Thus the Markov property implicates that the history does not matter for the evolution of the process, only the current state does. It also shows us how to take the transition probabilities p: $p(x,a,y) = \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a_t)$, where we made the additional assumption that $X_{t+1}|X_t, A_t$ does not depend on t, i.e., the system is time-homogeneous. If the Markov property does not hold for a certain sytem then there are no transition probabilities that describe the transition law, and therefore this system cannot be modeled as a Markov decision chain (with the given choice of states). For semi-Markov models we assume the Markov property for the embedded chain. Note that for Markov (decision) processes the Markov property holds at all times, because of the memoryless property of the exponentially distributed transition times.

It is important to note that whether the Markov property holds might depend on the choice of the state space. Thus the state space should be chosen such that the Markov property holds.

Example 17.3 Consider some single-server queueing system for which we are interested in the number of customers waiting. If we take as state the number of customers in the queue, then information on previous states gives information on the state of the server which influences the transitions of the queue. Therefore the Markov property does not hold. Instead, one should take as state the number of customers in the system. Under suitable conditions on service and interarrival times the Markov property holds. The queue length can be derived from the number of customers in the system.

Our conclusion is that the state space should be chosen such that the Markov property holds. Next to that, note that policies are functions of the states. Thus, for a policy to be implementable, the state should be observable by the decision maker.

Remark 17.4 A choice of state space satisfying the Markov property that always works is taking all previous observations as state, that is, the observed history is the state. Disadvantages are the growing size of the state space, and the complexity of the policies.

Modeling issues: decision epochs

Strongly related to the choice of states is the choice of decision epoch. Several choices are possible: does the embedded point represent the state before or after the transition or decision? This choice often has consequences for the states.

Example 17.5 Consider a service center to which multiple types of customers can arrive and where decisions are made on the basis of the arrival. If the state represents the state after the arrival but before the decision, then the type of arrival should be part of the state space. On the other hand, if the state represents the state before the potential arrival then the type of arrival will not be part of the state; the actions however will be more-dimensional, with for each possible arrival an entry.

The general rule when choosing the state space and the decision epochs is to do it such that the size of the state space is minimized. To make this clear, consider the following examples from queueing theory.

The M/M/1 queue can be modeled as a Markov process with as decision epochs all events (arrivals and departures) in the system, and as states the number of customers. In the case of the M/G/1 queue this is impossible: the remaining service time depends not only on the number of customers in the system, and thus the Markov property does not hold. There are two solutions: extend the state space to include the attained or remaining service time (the supplementary variable approach) or choose the decision epoch in such a way that the state space remains simple. We discuss methods to support both approaches, the latter first.

Example 17.6 Consider a model with s servers that each work with rate μ, two types of customers that arrive with rates λ_1 and λ_2, no queueing, and the option to admit an arriving customer. Blocking costs c_i for type i, if all servers are busy the only option is blocking. How to minimize blocking costs? When modeling this as a Markov decision process we have to choose the states and decision epochs. The problem here is that the optimal action depends on the type of event (e.g., an arrival of type 1) that is happening. There are two solutions to this: either we take state space $\{0,\ldots,s\}\times\{a1,a2,d\}$, the second dimension indicating the type of event. This can be seen as epoch the state after the transition but before a possible assignment. We can also take as state space $\{0,\ldots,s\}$, but then we have as action $\{0,1\}\times\{0,1\}$ with the interpretation that action $(a_1,a2)$ means that if an arrival of type i occurs then action a_i is selected. This has the advantage of a smaller state space and is preferable.

Exercise 17.1 Give the transition probabilities and direct rewards for both choices of decision epochs of Example 17.6.

Modeling issues: Poisson arrivals in interval of random length

Consider again the M/G/1 queue. To keep the state space restricted to the number of customers in the system the decision epochs should lie at the beginning and end of the service times, which have a duration S. This means that $\tau(x) = \mathbb{E}S$ for $x > 0$. We take $\tau(0) = 1/\lambda$, and thus $p(0,1) = 1$, $p(x, x-1+k) = \mathbb{P}(k \text{ arrivals in } S)$ for $x > 0$. Let us consider how to calculate these probabilities. First note that

$$q_k = \mathbb{P}(k \text{ arrivals in } S) = \int_0^\infty \frac{(\lambda t)^k}{k!} e^{-\lambda t} dS(t).$$

Let us calculate the generating function Q of the q_ks:

$$Q(\alpha) := \sum_{k=0}^{\infty} q_k \alpha^k = \int_0^\infty e^{-\lambda t(1-\alpha)} dS(t) = g(\lambda(1-\alpha)),$$

with g thus the Laplace-Stieltjes transform of S. The coefficients q_k can be obtained from Q in the following way: $Q^{(k)}(0) = (-\lambda)^k g^{(k)}(\lambda) = q_k k!$. In certain cases we can obtain closed-form expressions for this.

Example 17.7 (S exponential) Suppose $S \sim \text{Exp}(\mu)$, then $g(x) = \frac{1}{1+x/\mu}$. From this it follows that $Q(\alpha) = (1+\lambda(1-\alpha)/\mu)^{-1}$, and thus

$$Q^{(n)}(0) = n!(\frac{\lambda}{\mu})^n (1+\lambda/\mu)^{-(n+1)} = n!(\frac{\lambda}{\lambda+\mu})^n \frac{\mu}{\lambda+\mu}.$$

From this it follows that $q_k = (\frac{\lambda}{\lambda+\mu})^k \frac{\mu}{\lambda+\mu}$, and thus the number of arrivals is geometrically distributed.

If there is no closed-form expression for the q_k then some numerical approximation has to be used.

Modeling issues: Phase-type distributions

As said when discussing the choice of decision epochs, another option to deal with for example the M/G/1 queue is adding an additional state variable indicating the attained service time. This not only adds an additional dimension to the state space, but this extra dimension describes a continuous-time variable. This can be of use for theoretical purposes, but from an algorithmic point of view this variable has to be discretized in some way. A method to do so with a clear interpretation is the use of *phase-type distributions*.

The class of phase-types distributions can be defined as follows.

Definition 17.8 *(Phase-type distributions) Consider a Markov process with a single absorbing state 0 and some initial distribution. The time until absorbtion into 0 is called to have a Phase-type (PH) distribution.*

Example 17.9 The exponential, gamma (also called Erlang or Laplace), and hyperexponential distributions (the latter is a mixture of 2 exponential distributions) are examples of PH distributions.

An interested property of PH distributions is the fact that the set of all PH distributions is closed in the set of all non-negative distributions, i.e., any distribution can be approximated arbitrarily close by a PH distribution. This holds already for a special class of PH distributions, namely for mixtures of gamma distributions (all with the same rate).

Consider some arbitrary distribution function F, and write $E(k,m)$ for the distribution function of the gamma distribution with k phases (the shape parameter) and rate m.

Theorem 17.10 *For $m \in \mathbb{N}$, take $\beta_m(k) = F(\frac{k}{m}) - F(\frac{k-1}{m})$ and $\beta_m(m^2) = 1 - F(\frac{m^2-1}{m})$. Then for F_m with $F_m(x) = \sum_{k=1}^{m^2} \beta_k(m) E(k,m)(x)$ it holds that $\lim_{m\to\infty} F_m(x) = F(x)$ for all $x \geq 0$.*

Proof The intuition behind the result is that $E(km,m)(x) \to \mathbb{I}\{k \leq x\}$ and that $\sum_{k=1}^{m^2} \beta_k(m)\mathbb{I}\{k/m \leq x\} \to F(x)$. An easy formal proof is showing that the Laplace-Stieltjes transforms of F_m converge to the transform of F (which is equivalent to convergence in distribution). □

Modeling issues: Little's law and PASTA

Sometimes we are interested in maximizing performance measures that cannot be formulated directly as long-run average direct rewards, e.g., minimizing the average waiting time in some queueing system. There are two results that are helpful in translating performance measures such that they can be obtained through immediate rewards: Little's law and PASTA.

Little's law is an example of a *cost equation*, in which performance at the customer level is related to performance at the system level. E.g., in a system with Poisson arrivals it generally holds that $\mathbb{E}L = \lambda \mathbb{E}W$ with L the stationary queue length, W the stationary waiting time of customers, and λ the arrival rate. See El-Taha & Stidham [8] for an extensive treatment of cost equations.

PASTA stands for 'Poisson arrivals see time averages'. It means that an arbitrary arrival sees the system in stationarity. For general arrival processes this is not the case.

Example 17.11 (Waiting times in the M/M/1 queue) Suppose we want to calculate the waiting time in the M/M/1 queue by backward recursion. Denote the waiting time by W_q, and the queue length by L_q. Then Little's law states that $\mathbb{E}W_q = \mathbb{E}L_q/\lambda$. If the state x denotes the number in the system, then taking immediate reward $r(x) = (x-1)^+/\lambda$ will give $g = \mathbb{E}W_q$.

We can calculate $\mathbb{E}W_q$ also using PASTA. PASTA tells us that $\mathbb{E}W_q = \mathbb{E}L/\mu$: we have to wait for all the customers to leave, including the one in service. Thus taking $r(x) = x/\mu$ also gives $g = \mathbb{E}W_q$.

The equivalence of both expressions for $\mathbb{E}W_q$ can be verified directly from results for the M/M/1 queue, using $\mathbb{E}L = \rho/(1-\rho)$ and $\mathbb{E}L = \mathbb{E}L_q + \rho$ with $\rho = \lambda/\mu$:

$$\frac{\mathbb{E}L_Q}{\lambda} = \frac{\mathbb{E}L - \rho}{\lambda} = \frac{\frac{\rho}{1-\rho}}{\lambda} - \frac{1}{\mu} = \frac{1}{\mu-\lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu-\lambda)} = \frac{\mathbb{E}L}{\mu}.$$

Modeling issues: Countable state spaces

In our physical world all systems are finite, but there are several reasons why we would be interested in systems with an infinite state space: infinite systems behave 'nicer' than finite systems, bounds cannot exactly be given, the state space does not represent something physical but some concept of an unbounded nature, and so forth.

Example 17.12 (M/M/1 vs. M/M/1/N queue) For the M/M/1 queue there are nice expressions for the most popular performance measures, for the M/M/1/N queue these expressions are less attractive.

Example 17.13 (Work in process in production systems) In certain production systems the amount of work of process that can be stocked is evidently bounded. In other production systems such as administrative processes this is less clear: how many files that are waiting for further processing can a computer system store? This question is hard to answer if not irrelevant, given the current price of computer disk space.

Example 17.14 (Unobserved repairable system) Suppose we have some system for which we have no immediate information whether it is up or not, but at random times we get a signal when it is up. A natural candidate for the states are the numbers of time units ago since we last got a signal. By its nature this is unbounded.

Although we have good reasons to prefer in certain cases infinite state spaces, we need a finite state space as soon as we want to compute performance measures or optimal policies. A possible approach is as follows.

Consider a model with countable state space $\mathcal{X}$. Approximate it by a series of models with finite state spaces $\mathcal{X}^n$, such that $\mathcal{X}^{n+1} \supset \mathcal{X}^n$ and $\lim_{n\to\infty} \mathcal{X}^n = \mathcal{X}$. Let the nth model have transition probabilities $p^{(n)}$. These should be changed with respect to p such that there are no transitions from $\mathcal{X}^n$ to $\mathcal{X}\backslash\mathcal{X}^n$. This can for example be done by taking for each $x \in \mathcal{X}^n$

$$p^{(n)}(x,y) = p(x,y) \text{ for } x \neq y \in \mathcal{X}^n,\ p^{(n)}(x,y) = 0 \text{ for } y \notin \mathcal{X}^n,\ p^{(n)}(x,x) = p(x,x) + \sum_{y \notin \mathcal{X}^n} p(x,y).$$

Having defined the approximating model it should be such that the performance measure(s) of interest converge to the one(s) of the original model. There is relatively little theory about these types of results, in practice one compares for example $g^{(n)}$ and $g^{(n+1)}$ for different values of n.

Example 17.15 (M/M/1 queue) As finite nth approximation for the M/M/1 queue we could take the M/M/1/n queue. If and only if the queue is stable (i.e., $\lambda < \mu$) then $\nu_*^{(n)}(x) \to \nu_*(x)$ for all x.

Exercise 17.2 Consider the $M|D|1$ queue with a controllable server: at the beginning of a service time we may decide whether the service time is d_1 or d_2 ($d_1 < d_2$). There are additional costs for making the server work hard. Next to that there are costs for every unit of time that a customer spends waiting. The objective is to minimize the long-run average costs.
a. Give the value iteration formula (i.e., give an expression for $V_{(n+1)\tau}$ in terms of $V_{n\tau}$), specifying the values for p, r, τ, etc.

b. Make the state space finite by choosing some suitable bound and formulate again the value iteration formula.
c. Implement the value iteration algorithm in some suitable programming environment and report for different choices of the parameters on the optimal policy and values. Pay special attention to the question whether the bound approaches the unbounded system well enough, and take the implementation remarks from the end of Section 6 into account.

Exercise 17.3 Consider the $D|M|1$ queue. Every finished customer gives a reward r, by giving a discount d when entering we can decrease the interarrival time from a_1 to a_2. There are costs for every unit of time that a customer spends waiting. The objective is to minimize the long-run average costs. Answer the same questions as in Exercise 17.2 for this model.

18 Curse of dimensionality

We see in practice that Markov decision chains are less used than we might expect. The reason for this is the *curse of dimensionality*, as already described in 1961 in Bellman [2]. A fact is that most practical problems have multi-dimensional state spaces. Now suppose we have a model with state space $\mathcal{X} = \{(x_1, \ldots, x_n) | x_i = 1, \ldots, B\}$. Then the number of different states is $|\mathcal{X}| = n^B$. Note from the backward recursion algorithm that we need in memory at least one double for every state, thus the memory use is at least $|\mathcal{X}|$ floating point numbers. For $n = B = 10$ this would require $\pm$100 Gb of memory, without even thinking about the processing requirements that would be necessary.

Example 18.1 (Production process) Consider a production process for N products, where product n has K_n production steps (on in total M machines), and production step k for product n has B_{nk} buffer spaces for waiting jobs. A typical decision is which job to schedule next on each machine. The number of dimensions is (at least) $\sum_{n=1}^{N} K_n$, with $\prod_{n=1}^{N} \prod_{k=1}^{K_n} (B_{nk}+1)$ the number of states. This would be sufficient if all machines had a production time of 1 and no costs for switching from one product to the next. If this is the case (as it is often in job shops) then additional state variables indicating the states of the machines have to be added.

Example 18.2 (Service center) Consider a service center (such as a call center) where customers of different types arrive, with multiple servers that can each process a different subset of all customer types. To model this we need at least a variable for each customer class and a variable for each server (class). Service centers with 5 or 10 customer and server classes are no exception, leading to 10 or 20 dimensions.

In the following sections we will discuss a number of approximation methods that can be used (in certain cases) to solve high-dimensional problems.

Exercise 18.1 Consider a factory with m machines and n different types of products. Each product has to be processed on each machine once, and each type of product has different processing requirements. There is a central place for work-in-process inventory that can hold a total of k items, including the products that are currently in service. Management wants to find optimal

dynamic decisions concerning which item to process when on which machine. Suppose one considers to use backward recursion.
a. What is the dimension of the state space?
b. Give a description of the state space.
c. Give a formula for the number of states.
d. Give a rough estimate of this number for $m = n = k = 10$.

19 One-step improvement

The one-step improvement works only for models for which the value function of a certain fixed policy R can be obtained, without being hindered by the curse of dimensionality. Crucial is the fact that practice shows that policy improvement gives the biggest improvement during the first steps. One-step improvement consist of doing the policy improvement step on the basis of the value of R. It is guaranteed to give a better policy R', but how good R' is cannot be obtained in general, because of the curse of dimensionality.

Note that it is almost equally demanding to store a policy in memory than it is to store a value function in memory. For this reason it is often better to compute the action online. That is, if the current state is x, then $V^R(y)$ is calculated for each y for which $p(x,a,y) > 0$ for some a, and on the basis of these numbers $R'(x)$ is computed. Note that the sparseness of the matrix P is crucial in keeping the computation time low.

The most challenging step in one-step improvement is computing V^R for some policy R. The practically most important case is where R is such that the Markov process splits up in several independent processes. We show how to find V^R on the basis of the value functions of the components.

Suppose we have n Markov reward processes with states $x^i \in \mathcal{X}^i$, rewards r^i, transition rates $\lambda^i(x^i,y^i)$, uniformization parameters γ^i, average rewards g^i, and value functions V^i. Consider now a model with states $x = (x^1,\ldots,x^n) \in \mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^n$, rewards $r(x) = \sum_{i=1}^n r^i(x^i)$, transition rates $\lambda(x,y) = \lambda^i(x^i,y^i)$ if $x_j = y_j$ for all $j \neq i$, average reward g, and value function V.

Theorem 19.1 *$g = \sum_i g^i$ and $V(x) = \sum_i V^i(x^i)$ for all $x \in \mathcal{X}$.*

Proof Consider the Poisson equation of component i:

$$V^i(x^i)\sum_{y^i}\lambda^i(x^i,y^i) + g^i = r^i(x^i) + \sum_{y^i}\lambda^i(x^i,y^i)V^i(y^i).$$

Sum over i and add $\sum_{y^i}\lambda^i(x^i,y^i)\sum_{j\neq i}V^j(x^j)$ to both sides. Then we get the optimality equation of the system as specified, with $g = \sum_i g^i$ and $V = \sum_i V^i$. □

Example 19.2 Consider a fully connected communication network, where nodes are regional switches and links consists of several parallel communication lines. The question to be answered in this network is how to route if all direct links are occupied? To this model we apply one-step optimization with initial policy R that rejects calls if all direct links are occupied. This implies that all link groups are independent,

and Theorem 19.1 can be used to calculate the value function for each link group. If a call arrives for a certain connection and all direct links are occupied, then online it is calculated if and how this call should be redirected. Note that in this case it will at least occupy two links, thus it might be optimal to reject the call, even if routing over multiple links is possible.

Calculating the value for each link group can be done very rapidly, because these are one-dimensional problems. But they even allow for an exact analysis, if we assume that they behave like Erlang loss models.

We derive the value function of an Erlang loss model with parameters λ, $\mu = 1$, s, and "reward" 1 per lost call. The Poisson equation is as follows:

$$sV(s) + g = \lambda + sV(s-1);$$

$$x < s: \quad (x+\lambda)V(x) + g = \lambda V(x+1) + xV(x-1).$$

Note that we are only interested in differences of value functions, not in the actual values. These differences and g are given by $g = \lambda B(s,a)$ and $V(k+1) - V(k) = B(s,a)/B(k,a)$ with $a = \lambda/\mu$ and

$$B(k,a) = \frac{a^k/k!}{\sum_{j=0}^{k} a^j/j!},$$

the Erlang blocking formula. This example comes from Ott & Krishnan [12].

Example 19.3 (value function M/M/1 queue) The M/M/1 queue with the queue length as immediate reward is another example for which we can compute the value function explicitly. Instead of solving the Poisson equation we give a heuristic argument. It is known that $g = \lambda/(\mu - \lambda)$. By using a coupling argument it is readily seen that $V(x+1) - V(x) = (x+1)\mathbb{E}B$, with B the length of a busy period. For $\mathbb{E}B$ we can obtain the following relation by conditioning on the first event in a busy period:

$$\mathbb{E}B = \frac{1}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} 2\mathbb{E}B \Rightarrow \mathbb{E}B = (\mu-\lambda)^{-1},$$

and thus

$$V(x) - V(0) = \sum_{i=1}^{x} (V(i) - V(i-1)) = \frac{x(x+1)}{2(\mu-\lambda)}.$$

Exercise 19.1 Consider an $M/M/s/s$ queue with two types of customers, arrival rates λ_1, λ_2, and $\mu_1 = \mu_2$. Blocking costs are different, we are interested in the average weighted long-run blocking costs. Give g and V for this queue. Now we have the possibility to block customers, even if there are servers free. What will be the form of the policy after one step of policy iteration? Compute it for some non-trivial parameter values.

Exercise 19.2 We consider a single arrival stream with rate 2 that can be routed to two parallel single-server queues, with service rates 1 and 2. Consider first a routing policy that assigns all customers according to i.i.d. Bernoulli experiments, so-called Bernoulli routing. We are interested in the average total long-run number of customers in the system. Compute the optimal routing probability and the g and V belonging to this policy. Show how to compute the one-step improved policy. How can you characterize this policy?

20 Approximate dynamic programming

Approximate dynamic programming is another, more ambitious method to solve high-dimensional problems. It was introduced in the dynamic programming community and further formalized in Bertsekas & Tsitsiklis [4].

The central idea is that $V^R(x)$ can be written as or estimated by a function with a known structure (e.g., quadratic in the components). Let us call this approximation $W^R(r,x)$, with r the vector of parameters of this function. Thus the problem of computing $V^R(x)$ for all x is replaced by computing the vector r, which has in general much less entries.

Example 20.1 Let $W^R(r,x)$ represent the value function for the long-run average weighted queue length in a single-server 2-class preemptive priority queue with Poisson arrivals and exponential service times. Then $W^R(r,x) = r_0 + r_1x_1 + r_2x_2 + r_{11}x_1^2 + r_{22}x_2^2 + r_{12}x_1x_2$, with x_i the number of customers of type i in the system. Instead of computing $V^R(x)$ for all possible x we only have to determine the six coefficients. (Groenevelt et al. [9])

A summary of the method is as follows (compare with policy iteration):
0. Choose some policy R.
1. Simulate/observe model to estimate V^R by V^R in states $x \in \mathcal{X}' \subset \mathcal{X}$;
2. Approximate $\tilde{V}^R(x)$ by $W^R(r,x)$ (with r such that it minimizes $\sum_{x\in\mathcal{X}'} |\tilde{V}^R(x) - W^R(r,x)|$);
3. Compute new policy R' minimizing $W^R(r,x)$, go to step 1 with $R = R'$.

21 LP approach

For Markov reward chains we saw in Section 4 that the average reward g is equal to

$$\sum_x \pi_*(x) r(x)$$

with π_* the unique solution of

$$\sum_y \pi_*(y) p(y,x) - \pi_*(x) = 0$$

and

$$\sum_x \pi_*(x) = 1, \quad \pi_*(x) \geq 0.$$

Of course, π_* can be seen as the stationary distribution.

The same approach applies to Markov decision chains: the maximal average reward is given by

$$\max \sum_x \sum_a \pi_*(x,a) r(x,a)$$

subject to

$$\sum_y \sum_a \pi_*(y,a) p(y,a,x) - \sum_a \pi_*(x,a) = 0$$

and

$$\sum_x \sum_a \pi_*(x,a) = 1, \quad \pi_*(x,a) \geq 0.$$

Now the $\pi_*(x,a)$ are called the (stationary) state-action frequencies, they indicate which fraction of the time the state is x and action a is chosen at the same time, $\sum_a \pi_*(x,a)$ can be interpreted as the stationary probability of state x under the optimal policy R^*. For recurrent states $\pi_*(x,a) > 0$ for at least one $a \in \mathcal{A}$, but there might be more. Any action a for which $\pi_*(x,a) > 0$ is optimal in x. However, the standard simplex method will find only solutions with $\pi_*(x,a) > 0$ for one a per x. To see this, consider the number of equations. This is $\mathcal{X}+1$. However, one inequation is redundant, because the first $\mathcal{X}$ rows are dependent:

$$\sum_x [\sum_y \sum_a \pi_*(y,a) p(y,a,x) - \sum_a \pi_*(x,a)] = 0.$$

Thus we can leave out one of the inequalities, giving a total of $\mathcal{X}$. Now the simplex method rewrites the constraint matrix such that each solution it evaluates consists of a number of non-negative *basic* variables (equal to the number of constraints) and a number of *non-basic* variables equal to 0. In our case there are thus $\mathcal{X}$ basic variables, one corresponding to each state.

Exercise 21.1 Give the LP formulation for semi-Markov decison processes.

22 Multiple objectives

Up to now we looked at models with a single objective function. In practice however we often encounter problems with more than one objective function, which are often formulated as maximizing one objective under constraints on the other objectives.

There are two solution methods for this type of problems: linear programming and Lagrange multipliers. A general reference to constrained Markov decision chains is Altman [1].

Let us first consider the LP method. Constraints of the form $\sum_x \pi_*(x,a)c(x,a) \leq \alpha$ can be added directly to the LP formulation. Note that by adding a constraint the number of basic variables increases by one too: the corresponding policy randomizes. Suppose that in state x we find that $\pi_*(x,a) > 0$ for more than one a, then the optimal policy chooses action a' with probability $\pi_*(x,a')/\sum_a \pi_*(x,a)$. The next example shows that this type of randomization now plays a crucial role in obtaining optimal policies.

Example 22.1 Take $|\mathcal{X}| = 1$, $\mathcal{A} = \{1,2,3\}$, $r = (0,1,2)$, $c = (0,1,4)$, $\alpha = 2$. The LP formulation is:

$$\max\{\pi(1,2) + 2\pi(1,3)\}$$

s.t.

$$\pi(1,1) + \pi(1,2) + \pi(1,3) = 1,$$
$$\pi(1,2) + 4\pi(1,3) \leq 2.$$

It is readily seen that the optimal solution is given by $(0, 2/3, 1/3)$, thus a randomization between action 2 and 3.

Let us now discuss the Lagrange multiplier approach. The main disadvantage of the LP approach is that for a problem with K constraints one has to solve an LP with $|\mathcal{X}| \times |\mathcal{A}|$ decision variables and $\mathcal{X} + K$ constraints.

Instead, we use backward recursion together with Lagrange multipliers. Assume we have single constraint, and introduce the Lagrange multiplier γ.

The crucial idea is to replace the direct reward r by $r - \gamma c$. We will use the following notation: the average reward for a policy R is written as usual with g^R, the average constraint value is written as f^R.

Theorem 22.2 *Suppose there is a $\gamma \geq 0$ such that*

$$R(\gamma) = \arg\max_R \{g^R - \gamma f^R\}$$

with $f^{R(\gamma)} = \alpha$, then $R(\gamma)$ is constrained optimal.

Proof Take some R such that $f^R \leq \alpha$. Then $g^R - \gamma f^R \leq g^{R(\gamma)} - \gamma f^{R(\gamma)}$ and thus $g^R \leq g^{R(\gamma)} - \gamma(\alpha - f^R) \leq g^{R(\gamma)}$. □

The function $f^{R(\gamma)}$ is decreasing, but not continuous in γ. The function $g^{R(\gamma)} - \gamma f^{R(\gamma)}$ is piecewise linear, and non-differentiable in those values of γ for which multiple policies are optimal: there the derivative changes. How this can be used to construct an optimal policy is first illustrated the next example, and then formalized in the algorithm that follows the example.

Example 22.3 Consider again $|\mathcal{X}| = 1$, $\mathcal{A} = \{1,2,3\}$, $r = (0,1,2)$, $c = (0,1,4)$. Now $g - \gamma f = (0, 1-\gamma, 2-4\gamma)$. Policy 3 is optimal up to $\gamma = 1/3$, then policy 2 is optimal until $\gamma = 1$.

Suppose that $\alpha = 2$. $\gamma < 1/3$ gives $R(\gamma) = 2$ and $f = 4$. $1 > \gamma > 1/3$ gives $R(\gamma) = 1$ and $f = 1$. For $\gamma = 1/3$ both policy 2 and 3 are optimal. By randomizing between the policies we find an R with $f = \alpha$, which is therefore optimal.

We introduce the following notation: Let $\mathcal{R}(\gamma)$ be the set of optimal policies for Lagrange parameter γ, and with $R = pR_1 + (1-p)R_2$ we mean that in every state with probability p the action according to R_1 is chosen and with $1-p$ the action according to R_2.

Then we have the following algorithm to construct optimal policies.

Algorithm:

1. if $\exists R \in \mathcal{R}(0)$ with $f^R \leq \alpha \Rightarrow R$ is optimal
2. else: vary γ until one of 2 situations occurs:

A. $\exists R \in \mathcal{R}(\gamma)$ with $f^R = \alpha \Rightarrow R$ is optimal

B. $\exists R_1, R_2 \in \mathcal{R}(\gamma)$ with $f^{R_1} < \alpha$, $f^{R_2} > \alpha$
take $R = pR_1 + (1-p)R_2$ such that $f^R = \alpha \Rightarrow R$ is optimal

Exercise 22.1 Consider the $M|M|1|3$ queue with admission control, i.e., every customer can be rejected on arrival. The objective is to maximize the productivity of the server under a constraint on the number of customers that are waiting.

a. Formulate this problem as a LP with general parameter values.
b. Solve the LP for $\lambda = \mu = 1$ and $\alpha = 0.5$. Interpret the results: what is the optimal policy?
c. Reformulate the problem with general parameter values using a Lagrange multiplier approach.
d. Give for each γ the value of f and g for the same parameter values as used for the LP method.
e. What is the optimal value of γ?

23 Dynamic games

In this section we restrict to non-cooperative games (i.e., no coalitions are allowed) and 2 players. We distinguish between two situations: at each decision epoch the players reveal their decision without knowing the others decision, or the players play one by one after having witnessed the action of the previous player and its consequences (*perfect information*).

We start with the first situation. Now the action a is two-dimensional: $a = (a_1, a_2) \in \mathcal{A}_1 \times \mathcal{A}_2$ with a_i the action of player i. This looks like a trivial extension of the 1-player framework, but also the reward is two-dimensional: $r(x,a) = (r_1(x,a), r_2(x,a))$, and every player has as objective maximizing its own long-run average expected reward. A special case are *zero-sum games* for which $r_2(x,a) = -r_1(x,a)$; these can be seen as problems with a single objective, and one player maximizing the objective, and the other minimizing it. For each state the value is a 2-dimensional vector. In the one-dimensional situation choosing an action is looking for given x at $r(x,a) + \sum_y p(x,a,y)V(y)$ for various a; now we have to consider for given x the vector

$$\left(r_1(x,a) + \sum_y p(x,a,y)V_1(y), r_2(x,a) + \sum_y p(x,a,y)V_2(y)\right) =: \left(Q_1(x,a), Q_2(x,a)\right)$$

for vectors $a = (a_1, a_2)$. This is called a *bi-matrix game*, and it is already interesting to solve this by itself. It is not immediately clear how to solve these bi-matrix games. An important concept is that of the *Nash-equilibrium*. An action vector a' is called a Nash-equilibrium if no player has an incentive to deviate, which is in the two-player setting equivalent to

$$Q_1(x,(a_1,a_2')) \leq Q_1(x,a') \text{ and } Q_2(x,(a_1',a_2)) \leq Q_2(x,a').$$

Example 23.1 (prisoners dilemma) The prisoners dilemma is a bi-matrix game with the following value or *pay-off* matrix:

$$\begin{pmatrix} (-5,-5) & (0,-10) \\ (-10,0) & (-1,-1) \end{pmatrix}.$$

Its interpretation is as follows: If both players choose action 2 (keeping silent), then they get each 1 year prison. If they talk both (action 1), then they get each 5 years. If one of them talks, then the one who remains silent gets 10 years and the other one is released. The Nash-equilibrium is given by $a' = (1,1)$, while $(2,2)$ gives a higher pay-off for both players.

Two-player zero-sum games, also called *matrix games*, have optimal policies if we allow for randomization in the actions. This result is due to Van Neumann (1928). Let player 1 (2) be maximizing (minimizing) the pay-off, and let p_i be the policy of player i (thus a distribution on $\mathcal{A}_i$), and Q the pay-off matrix. Then the expected pay-off is given by $p_1 Q p_2$. Van Neumann showed that

$$\max_{p_1} \min_{p_2} p_1 Q p_2 = \min_{p_2} \max_{p_1} p_1 Q p_2 :$$

the equilibrium always exists, is unique, and knowing the opponent's distribution does improve the pay-off.

We continue with games where the players play one by one, and we assume a zero-sum setting. Good examples of these types of games are board games, see Smith [15] for an overview.

We assume that $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$. An epoch consists of two moves, one of each player. The value iteration equation (4) is now replaced by

$$V_{t+1}(x) = \max_{a_1 \in \mathcal{A}_1} \left\{ r(x,a_1) + \sum_y p(x,a_1,y) \min_{a_2 \in \mathcal{A}_2} \left\{ r(y,a_2) + \sum_z p(y,a_2,z) V_t(z) \right\} \right\}.$$

Quite often the chains are not unichain, but $r(x,a) = 0$ for all but a number of different absorbing states that corresponds to winning or losing for player 1. The goal for player 1 is to reach a winning end state by solving the above equation. Only for certain simple games this equation can be solved completely, for games such as chess other methods have to be used.

Exercise 23.1 Determine the optimal policies for the matrix game

$$\begin{pmatrix} 1 & -2 \\ 0 & 3 \end{pmatrix}.$$

Does this game have an equilibrium if we do not allow for randomization?

Exercise 23.2 Determine the optimal starting move for the game of Tic-tac-toe.

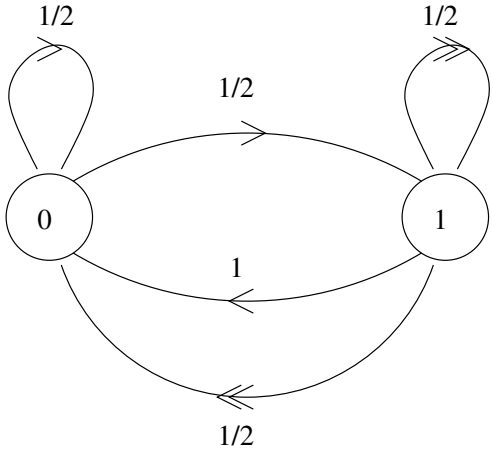
24 Disadvantages of average optimality

If there are multiple average optimal policies, then it might be interesting to consider also the 'transient' reward. Take the following example.

Example 24.1 $\mathcal{X} = \mathcal{A} = \{1,2\}$, $p(i,a,2) = 1$ for $i \in \mathcal{X}$ and $a \in \mathcal{A}$, $r(1,1) = 10^6$, other rewards are 0. Then all policies are average optimal, but action 1 in state 1 is evidently better.

We formalize the concept of 'better' within the class of average optimal policies. Let $\mathcal{R}$ be the set of average optimal policies. Then the policy $R \in \mathcal{R}$ that has the highest bias (= value function normalized w.r.t. stationary reward) is called *bias optimal*, a refinement of average optimality.

Example 24.2 Consider the model in the figure below. "$>$" denotes action 1, "$\gg$" denotes action 2; the numbers next to the arrows denote the transition probabilities. The direct rewards are given by $r(0,1) = 0$, $r(1,1) = 1$, and $r(1,2) = 2/3$. This model is communicating with 2 average optimal policies and 1 bias optimal policy, as we shall see next.



The optimality equation is as follows:

$$V(0)+g=\frac{1}{2}V(0)+\frac{1}{2}V(1);$$

$$V(1)+g=\max\{1+V(0),\frac{2}{3}+\frac{1}{2}V(0)+\frac{1}{2}V(1)\}.$$

The solutions are given by:

$$g=1/3,\ V(0)=c,\ V(1)=2/3+c,\ c\in\mathbb{R}.$$

The maximum is attained by actions 1 and 2, thus both possible policies ($R_1=(1,1)$ and $R_2=(1,2)$) are average optimal. Do both policies have the same bias?

The bias B^{R_i} is the solution of the optimality equation with, additionally, the condition $\sum_x \pi_*^{R_i}(x)B^{R_i}(x)=0$. This assures that the bias of the stationary state is equal to 0. Reformulating the last condition gives

$$B^{R_i}=V-<\pi_*^{R_i},V>e.$$

Let us calculate the bias. Take $V=(0,2/3)$, the stationary distributions under both policies are $\pi_*^{R_1}=(2/3,1/3)$, $\pi_*^{R_2}=(1/2,1/2)$. From that it follows that $B^{R_1}=(-2/9,4/9)$ and $B^{R_2}=(-1/3,1/3)$. Thus only R_1 is bias optimal. The reason for this difference is that $\pi_*^{R_1}\neq\pi_*^{R_2}$.

Next we formulate a method to determine the bias optimal policy. Let (g,V) be a solution of

$$V+g=\max_{R:\mathcal{X}\to\mathcal{A}}\{r(R)+P(R)V\},$$

and $\mathcal{R}=\arg\max\{r+PV\}$, the *set* of average optimal policies. We have to look for a policy in $\mathcal{R}$ that minimizes $<\pi_*^R,V>$. This is again an average reward problem, with optimality equation

$$V'+g'=\max_{R\in\mathcal{R}}\{-V+P(R)V'\}.$$

The solution is the set of bias optimal policies, with bias $V+g'$.

Remark 24.3 This process can be repeated, to get reward as early as possible, etc. The remaining policy is equal to the limiting discount optimal policy.

The lack of focus on transient reward is not the only objection that can be made against the long-run average reward as criterion: The focus on expectations is another one. Let us consider again an example.

Example 24.4 Consider a model with rewards $1,1,1,1\ldots$; under the long-run average reward criterion this cannot be distinguished from

$$\text{with prob. } \tfrac{1}{2}\text{: } 0,0,0,\ldots \text{ or } 2,2,2,\ldots,$$

or at each point in time 0 or 2 with prob. 1/2.

The first and last example can occur in a single-class model, the middle one can only occur in a multi-class model. It has to do with the distribution of the average reward, of which g is the expectation. This distribution has hardly been studied in the literature. A possible way to consider this type of problem is by decision trees.

Consider a single-class aperiodic model, the limiting average variance of policy R is given by:

$$\sigma^2(R) = \lim_{t\to\infty} \mathbb{E}(r(X_t(R),A_t(R)) - g^R)^2.$$

In terms of state-action frequencies:

$$\sigma^2(R) = \sum_{x,a} (r(x,a) - g^R)^2 \pi_*^R(x,a).$$

Because $g^R = \sum_{x,a} r(x,a)\pi_*^R(x,a)$, we find

$$\sigma^2(R) = \sum_{x,a} \left(r^2(x,a) - 2r(x,a)g^R + (g^R)^2 \right) \pi_*^R(x,a) =$$

$$\sum_{x,a} r^2(x,a)\pi_*^R(x,a) - \left(\sum_{x,a} r(x,a)\pi_*^R(x,a) \right)^2.$$

Now we have a multi-criteria decision problem. We have the following possibilities for choosing the objective:
- $\max_R\{g^R|\sigma^2(R) \le \alpha\}$;
- $\min_R\{\sigma^2(R)|g^R \ge \alpha\}$;
- $\max_R\{g^R - \gamma\sigma^2(R)\}$.

Note that all are risk-averse, i.e., the utility is concave. All problems are constrained Markov decision problems with either a quadratic objective or a quadratic constraint, thus we have to rely on mathematical programming to solve these problems.

Exercise 24.1 Consider a discrete-time Markov decision process with states $\{0,1,2\}$, 2 actions in 0 and 1 action in 1 and 2, and transition probabilities p: $p(0,1,1) = p(0,2,2) = 2/3$, $p(0,1,2) = p(0,2,1) = 1/3$, $p(1,1,0) = p(2,1,0) = 1$, and rewards r equal to: $r(1,1) = 1$, $r(0,2) = 1/3$, $r(0,1) = r(2,1) = 0$.
Compute the average reward, the bias and the average variance for both possible policies.
Which policy would you prefer?

25 Monotonicity

For several reasons it can be useful to show certain structural properties of value functions. This can be used to characterize (partially) optimal policies, or it can be used as a first step in comparing the performance of different systems.

Admission control

Consider the value function of the M/M/1 queue with admission control (assuming that $\lambda+\mu \leq 1$), for rejection costs r and direct costs $C(x)$ if there are x customers in the system:

$$V_{n+1}(x) = C(x) + \lambda \min\{r + V_n(x), V_n(x+1)\} + \mu V_n((x-1)^+) + (1-\lambda-\mu)V_n(x).$$

A *threshold policy* is a policy that admits customers up to a certain threshold value, above that value customers are rejected. Whether or not the optimal policy for a certain $n+1$ is a threshold policy depends on the form of V_n. The next theorem gives a sufficient condition.

Theorem 25.1 *If V_n is convex, then a threshold policy is optimal for V_{n+1}.*

Proof V_n convex means:

$$2V_n(x+1) \leq V_n(x) + V_n(x+2) \text{ for all } x \geq 0,$$

and thus

$$V_n(x+1) - r - V_n(x) \leq V_n(x+2) - r - V_n(x+1)$$

for all $x \geq 0$.

If rejection is optimal in x, thus $0 \leq V_n(x+1) - r - V_n(x)$, then also $0 \leq V_n(x+2) - r - V_n(x+1)$, and thus rejection is also optimal in $x+1$. □

Theorem 25.2 *If C and V_0 are convex and increasing (CI), then V_n is CI for all n.*

Proof By induction to n. Suppose V_n is CI. This means

$$V_n(x) \leq V_n(x+1) \text{ for all } x \geq 0$$

$$2V_n(x+1) \leq V_n(x) + V_n(x+2) \text{ for all } x \geq 0$$

They can be used to show that the same inequalities hold for V_{n+1}. □

Corollary 25.3 *If C is CI, then a threshold policy is optimal.*

Proof $V_n(x) - V_n(0)$ converges as $n \to \infty$ to a solution V of the optimality equation and thus V is CI, and therefore the average optimal policy is of threshold type. □

A standard choice is $C(x) = cx$, which amounts to c holding costs for each unit of time that a customer is in the system. If we take $C(x) = c(x-1)^+$, then only the customers in queue count. Note that $c(x-1)^+$ is convex (if $c \geq 0$).

Remark 25.4 This result holds also for the M/M/s queue.

A server-assignment model

Consider next a model with m classes of customers, arrival rates λ_i and service rates μ_i ($\max_i \mu_i = \mu$, $\sum_i \lambda_i + \mu \leq 1$), a single server, and dynamic preemptive server assignment.

The value function is as follows:

$$V_{n+1}(x) = C(x) + \sum_i \lambda_i V_n(x+e_i) +$$

$$\min_i \{\mu_i V_n((x-e_i)^+) + (\mu - \mu_i) V_n(x)\} + (1 - \sum_i \lambda_i - \mu) V_n(x).$$

Theorem 25.5 *If c and V_0 satisfy*

$$\mu_i f(x-e_i) + (\mu - \mu_i) f(x) \leq \mu_j f(x-e_j) + (\mu - \mu_j) f(x)$$

for all x and $i < j$, $x_i > 0$ and $x_j > 0$ and

$$f(x) \leq f(x+e_i)$$

for all x and i, then so do V_n for all $n > 0$.

Corollary 25.6 *Under the above conditions a preemptive priority policy is optimal. In the special case that $C(x) = \sum_i c_i x_i$ then the conditions are equivalent to $c_i \geq 0$ and $\mu_1 c_1 \geq \cdots \geq \mu_m c_m$, resulting in the so-called μc-rule.*

With this method many one and two-dimensional systems can be analyzed, but few multi-dimensional.

Remark 25.7 Note that other cost functions might lead to the same optimal policy. Whether or not this is the case is checked by verifying if the conditions hold for the choice of cost function.

Exercise 25.1 Consider the $M/M/s/s$ queue with admission control with two classes of customers. Both classes have the same average service time, but differ in the reward per admitted customer.
a. Formulate the backward recursion value function.
b. Is the value function concave or convex? Show it.
c. Consider a similar system but with rewards for each finished customer. How could you analyze that?

Exercise 25.2 Consider 2 parallel single-server queues with exponential service times, with equal rates. Customers arrive according to a Poisson process and are assigned in a dynamic way to the 2 queues. The objective is to minimize the average number of customers in the system.
a. Formulate the backward recursion equation.
b. Which relation on V_n must hold for shortest queue routing to be optimal?
c. Show that the value function is symmetric (i.e., $V_n(x,y) = V_n(y,x)$).
d. Prove by induction to n that shortest queue routing is optimal.

26 Incomplete information

The curse of dimensionality is not the only reason why Markov decision theory (or mathematical modeling in general) is little used in practice. There are two more reasons, that both are related to the same general concept of *incomplete* or *partial information*. The first is that our methods require all parameters to be given. This is in many cases an unrealistic assumption: parameters vary, depending on many other known and unknown underlying parameters. Therefore it is often not possible to give reliable estimates of parameters.

Example 26.1 A crucial parameter when optimizing a call center is the rate at which customers arrive. This parameter depends on many variables, some of which are known at the moment the planning is done (time of day, week of the year, internal events influencing the rate such as advertisement campaigns, etc.) and some of which are unknown (mainly external events such as weather conditions). This estimation issue is of major importance to call centers.

Next to unknown parameters it might occur that the state is not (completely) observed. In principle the observation at t can be any random function of the whole history of the system (thus including models with delay in observations), but usually it is a random function of the state at t.

Example 26.2 In a telecommunication network decisions are made at the nodes of the system. In each node we often have delayed incomplete info on the other nodes. How to make for example decisions concerning the routing of calls?

Example 26.3 The state of a machine deteriorates when it is functioning, but we only known whether it is up or down. Timely preventive maintenance prevents expensive repairs after failure. When to schedule preventive maintenance? What are the advantages of *condition monitoring*?

Example 26.4 In most card games we have partial information on the other hands.

The standard method in the case of unknown parameters is to observe the system first, estimate the parameters (the learning phase), and then control the system on the basis of the estimates (the control phase). The disadvantages are that we do not improve the system during the learning phase, and that we do not improve the parameter estimates during the control phase. This method gets even worse if the parameters change over time, which is the rule in practice. Thus we need more sophisticated methods.

There are several methods with each having their own advantages and disadvantages. Crucial to all these methods is that they do not first estimate and then control on the basis of these estimates, but that while the system is being controlled the parameter estimations are improved and with that the decisions. The simplest method, useful for example in the case of unknown arrival parameters consists of a standard statistical estimation procedure giving the most likely value, followed by the execution of for example backward recursion at each time epoch using the most recent estimates. There are other methods in which the estimation and optimization steps cannot so easily be separated. One is *Q-learning*, in which the value function is updated using ideas from stochastic approximation. It is useful in the case that nothing is known about the transition probabilities, it is a method that makes no initial assumptions. A mathematically more

sophisticated and numerically more demanding method is Bayesian dynamic programming, for which an initial ("prior") distribution of the unknown parameters is needed.

Note that also approximate dynamic programming can be used for problems with uncomplete information. We made no assumptions on the transition structure, and therefore it can also be used for the partial-information case. As for Q-learning no initial distribution is needed.

Remark 26.5 If we consider just the average long-run reward then learning over a (very) long period and then controlling is (almost) optimal. The disadvantage is the loss of reward during the learning phase, see the discussion of disadvantages of the average reward criterion in Section 24. For this reason we mainly look at discounted rewards. Another problem is the issue of parameters varying slowly over time. Then we can never stop learning, and learning and control have to be done simultaneously.

Exercise 26.1 Consider an $M/M/1$ queue with admission control, thus arriving customers may be rejected. Every customer in queue costs 1 unit per time holding costs, but every admitted customer gives a reward r. It is the objective to maximize the discounted revenue minus holding costs. The arrival rate is 1, but the service rate is unknown.
a. Give a procedure to estimate the parameter of the service time distribution on the basis of the realizations up to now.
b. How would you use this to design a control algorithm for this model?
c. Implement a computer program that repeats the following experiment a number of times: draw the service rate from a homogeneous $[0,2]$ distribution, and simulate the queue with admission with the algorithm of part b implemented. Report on it for a few choices of parameters.
d. Compare the results of c to the model where the service rate is known from the beginning.

Q-learning

Q-learning is a method that is suitable for systems of which the state is observed, but no information is known about the transition probabilities, not even the structure.

We will work with a model with discounting (with parameter, $\beta \in [0,1)$), in discrete time. We allow the reward to depend on the new state, thus r is of the form $r(x,a,y)$ with x the current state, a the action, and y the new state.

The value function is then given by:

$$V(x) = \max_a \{\sum_y p(x,a,y)[r(x,a,y) + \beta V(y)]\}.$$

Define

$$Q(x,a) = \sum_y p(x,a,y)[r(x,a,y) + \beta V(y)].$$

Then

$$Q(x,a) = \sum_y p(x,a,y)[r(x,a,y) + \beta \max_a Q(y,a)].$$

This gives an alternative way for calculating optimal policies.

Now we move to systems with unknown parameters, and we assume that a simulator exists: we have a system for which it is possible to draw transitions according to the correct probability laws, but it is too time consuming to calculate transition probabilities (otherwise standard value iteration would be preferable).

Let $\xi_n(x,a) \in \mathcal{X}$ be the outcome of the simulation at stage n, for state x and action a. Let b_n be a series with

$$\sum_n b_n = \infty, \quad \sum_n b_n^2 < \infty.$$

The Q-learning algorithm works as follows:

$$Q_{n+1}(x,a) = (1-b_n)Q_n(x,a) + b_n \max_a [r(x,a,\xi_n(x,a)) + \beta Q_n(\xi_n(x,a),a)].$$

Then: $Q_n \to Q$, and thus in the limit the algorithm gives the optimal actions.

Remark 26.6 If the system is controlled in real time, then the algorithm is executed in an asynchronous way: only $Q(x,a)$ is updated for the current x and a. This has the risk that certain (x,a) combinations never get updated. Therefore sub-optimal actions should also be chosen to give the algorithm the possibility to learn on all (x,a) combinations.

Remark 26.7 Why did we take $\sum_n b_n = \infty$ and $\sum_n b_n^2 < \infty$? If $\sum_n b_n < \infty$ then there is no guaranteed convengence to the right value (e.g., $b_n \equiv 0$). On the other hand, if $\sum_n b_n^2 = \infty$, then there need not be convengence (e.g., $b_n \equiv 1$). The usual choice of b_n is $b_n = \frac{1}{n}$.

See also the Robbins-Monro stochastic approximation algorithm.

Bertsekas & Tsitsiklis [4] gives more information on Q-learning.

Exercise 26.2 Consider the model of Exercise 26.1. Is it useful to apply Q-learning to this model? Motivate your answer.

Bayesian dynamic programming

In Bayesian dynamic programming we make a difference between the state of the model (with state space $\mathcal{X}$) and the state of the algorithm (the *information state*) that represents the information we have. The information state is actually a distribution on the set of model states. This information state holds all information on all observations up to the current time. Thus it is observable, it only depends on the observations, and the Markov property holds: all information about the past is used. Thus (see Section 17) the information state can be used to solve the problem.

Starting with some well-chosen prior, the information state is updated every time unit using Bayes' rule. This gives the optimal policy, given the initial distribution. Note however that for state space $\mathcal{X}$ the information state space is given by $[0,1]^{|\mathcal{X}|}$, thus an $|\mathcal{X}|$-dimensional state space, which each variable a probability. Thus $\mathcal{X}$ is not even countable, we will certainly have to *discretize* the state space to make computations possible. Even for small state spaces and a coarse discretization of the interval $[0,1]$ this quickly leads to unfeasible state space sizes.

Let us formalize the framework. Next to the state space $\mathcal{X}$ we have an *observation space* $\mathcal{Z}$. With $q(x,z)$ we denote the probability of observing $z \in \mathcal{Z}$ in $x \in \mathcal{X}$. Next we define the information states, which are thus distributions on $\mathcal{X}$: $\mathcal{P} = [0,1]^{\mathcal{X}}$ (in case $|\mathcal{X}|$ countable).

Consider $u \in \mathcal{P}$. For each $z \in \mathcal{Z}$ there is a transition to a new state $v \in \mathcal{P}$; for observation z the distribution v is defined by

$$v(y) = \mathbb{P}(\text{now at } y | \text{before at } u, a \text{ chosen, } z \text{ observed}) = \frac{\sum_x u(x)p(x,a,y)q(y,z)}{\sum_x \sum_y u(x)p(x,a,y)q(y,z)},$$

and thus the transition probabilities are given by

$$p'(u,a,v) = \sum_x \sum_y u(x)p(x,a,y)q(y,z).$$

Here we assumed that for u different observations lead to different v; if this is not the case then the transition probabilities should be added.

Example 26.8 A service center is modeled as a single-server queue. We have the option to send customers to another center when the queue gets too long (admission control). We assume that the capacity of the queue is N: above N customers balk automatically. The partial-information aspect is that we do not observe the queue, we only observe whether or not the server is busy. The state is the number of customers in the system, $\mathcal{X} = \{0,1,2,\ldots,N\}$, action set $\mathcal{A} = \{0,1\}$ representing rejection and admission. The observation is 1 (0) is the server is busy (idle), $\mathcal{Z} = \{0,1\}$, and $q(x,1) = 1$ for $x > 0$, $q(0,0) = 1$. Every state is a vector with $N+1$ probabilities. Let $u \in \mathcal{P}$, and let λ and μ be the uniformized transition parameters, $\lambda + \mu = 1$. Then $p'(u,a,e_0) = \lambda \mathbb{I}\{a=0\}u(0) + \mu(u(0)+u(1))$. The second possible transition is $p'(u,a,v) = 1 - p'(u,a,e_0)$ with

$$v(y) = \frac{\lambda \mathbb{I}\{a=0\}u(y) + \lambda \mathbb{I}\{a=1\}u(y-1) + \mu u(y+1)}{1 - \lambda \mathbb{I}\{a=0\}u(0) - \mu u(1)},$$

for $y > 0$ (we assume that $a = 0$ is chosen in state N). It can probably be shown that v is stochastically increasing in the number of times that 1 is observed since the last observation of 0. From this we conclude that an optimal policy rejects from a certain number of consecutive observations of 1 on. See [11] for a similar result.

Example 26.9 Suppose there are several different medical treatment for a certain illness, each with unknown success probability. How to decide which treatment to use for each patient?

At first sight this can be translated in a one-state model. However, the reward depends on the success probability of the chosen treatment. Thus the success probabilities should be part of the state space.

Suppose there are two possible treatments, then the state is represented by the tuple (θ_1, θ_2), giving the success probabilities of the two treatments. However, this state is unobserved: instead of this we have as information state a tuple of distributions on $[0,1]$. Each time unit a new patient arrives, the question is how to treat this patient. This question can be answered by solving a Markov decision problem with as state space all possible tuples of independent 0-1 distributions. Thus $\mathcal{X} = [0,1]^2$, $\mathcal{A} = \{1,2\}$, the treatment to be used, $\mathcal{Z} = \{0,1\}$, the result of the treatment (1 means success). Because $\mathcal{X}$ is a continuous set, information states are densities: $\mathbb{P} = (\mathbb{R}_+^{[0,1]}, \mathbb{R}_+^{[0,1]})$, tuples of densities, both on the probabilities $[0,1]$.

Take $u=(u_1,u_2)$. For action a only u_a is updated. Consider $a=1$ (the case $a=2$ is equivalent). Then a success occurs with probability $\int_0^1 xu_1(x)dx$, and the resulting information state is $v=(v_1,v_2)$ with $v_2=u_2$ and v_1 defined by

$$v_1(y)=\frac{yu_1(y)}{\int_0^1 xu_1(x)dx}$$

in case of a success (which occurs thus with probability $\int_0^1 xu_1(x)dx$), and v_1 such that

$$v_1(y)=\frac{(1-y)u_1(y)}{\int_0^1 (1-x)u_1(x)dx}$$

in case of a failure (with probability $\int_0^1 (1-x)u_1(x)dx$).

Using this in an optimization algorithm is computationally infeasable, therefore we have to look for a method to reduce the size of the state space. That we will discuss next. See [5] for more information on this example.

In the examples we saw that in partial-information problems the state space very quickly becomes so big that direct computation of optimal policies is infeasable. In the first example we saw that instead of working with all distributions we can work with the time until the last time the system was empty. This is a much simpler representation of the state space. For systems with unknown parameters such a simpler representation of the information states exists sometimes as well. In such cases the densities that occur as information states fall into a certain class of parametrized families, such as Beta distributions. The crucial property is that the distribution should be closed under the Bayesian update. We will show that this is the case in the setting of Example 26.9.

Let us introduce the class of Beta distributions. A Beta (k,l) distribution has density $f(x)\propto x^k(1-x)^l$. Note that for $k=l=0$ we find the uniform distribution on $[0,1]$.

Theorem 26.10 *Consider a Bernoulli random variable X with parameter θ that has a Beta(k,l) distribution. Then $\theta|X=1$ (the a posteriori distribution) has a Beta $(k+1,l)$ distribution; $\theta|X=0$ has a Beta$(k,l+1)$ distribution.*

Proof We have:

$$\mathbb{P}(x\le\theta\le x+h|X=1)=\frac{\mathbb{P}(x\le\theta\le x+h,X=1)}{\mathbb{P}(X=1)}=$$

$$\frac{\mathbb{P}(x\le\theta\le x+h)\mathbb{P}(X=1|x\le\theta\le x+h)}{\mathbb{P}(X=1)}.$$

Dividing by h and taking the limit as $h\to 0$ gives (with f_X denoting the density of X):

$$f_{\theta|X=1}(x)=\frac{f_\theta(x)\mathbb{P}(X=1|\theta=x)}{\mathbb{P}(X=1)}=\frac{f_\theta(x)x}{\mathbb{P}(X=1)}.$$

because $f_\theta(x)=C(k,l)x^k(1-x)^l$ we find indeed that $\theta|X=1$ has density $Cx^{k+1}(1-x)^l$, with $C=C(k,l)/\mathbb{P}(X=1)$. A similar argument applies to the case $X=0$. □

This theorem is very useful for Example 26.9: we can replace the densities by two-dimensional discrete variables, resulting for the problem with two possible treatment in four variables. It is crucial to start with an initial Beta distributed belief (the prior) for each treatment, for example each having a uniform distribution. Crucial here is that if we start with a uniform prior, then all distributions are Beta distributions which can be characterized by two discrete parameters. Thus under a Beta-distributed prior the problem is 4-dimensional and can be solved using backward recursion.

Two types of distributions, where one is the parameter of the other and one remains within the same class of distributions after a Bayesain update, are called *conjugate distributions* (see DeGroot [7]).

Exercise 26.3 Consider a machine that detects certain rare particles. At discrete points in time it can give a signal or not. When it gives a signal, it is functioning and it has detected a particle. When it does not give a signal, it is either broken down or there was no particle. Every time unit there is a particle with probability p, and when it is functioning the machine breaks down with probability q. To replace a machine that is broken down a new one has to be bought. The objective is to have a working machine as often as possible for as little money as possible.
a. Describe the states and the transition probabilities of this system if it is observed when the machine breaks down. What is a reasonable policy?
b. Describe the information states and the transition probabilities of this system if it cannot be observed when the machine is down. What is a reasonable policy?

Exercise 26.4 We observe a realization y of an exponentially distributed random variable that has a parameter that is gamma(n,α) distributed. What is the a posteriori distribution of the parameter?

Acknowledgments

I thank Sandjai Bhulai for his many remarks and suggestions.

References

[1] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[3] D.P. Bertsekas. *Dynamic Programming*. Prentice-Hall, 1987.

[4] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] S. Bhulai and G.M. Koole. On the value of learning for Bernoulli bandits with unknown parameters. *IEEE Transactions on Automatic Control*, 45:2135–2140, 2000.

[6] E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.

[7] M.H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[8] M. El-Taha and S. Stidham, Jr. *Sample-Path Analysis of Queueing Systems*. Kluwer, 1998.

[9] R. Groenevelt, G.M. Koole, and P. Nain. On the bias vector of a two-class preemptive priority queue. *Mathematical Methods of Operations Research*, 55:107–120, 2002.

[10] L.C.M. Kallenberg. Finite state and action MDPs. In E.A. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes*, pages 21–86. Kluwer, 2002.

[11] G.M. Koole. On the static assignment to parallel servers. *IEEE Transactions on Automatic Control*, 44:1588–1592, 1999.

[12] T.J. Ott and K.R. Krishnan. Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of Operations Research*, 35:43–68, 1992.

[13] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[14] S.M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, 1983.

[15] D.R. Smith. Dynamic programming and board games. Unpublished, www.maths.ex.ac.uk/∼DKSmith/if16pape.pdf, 1999.

[16] H.C. Tijms. *Stochastic Modelling and Analysis: A Computational Approach*. Wiley, 1986.