1

Introduction

This chapter provides a short guided tour through the five parts of the book. We begin with some terminology.

- A Markov decision process (MDP) is obtained by controlling the transition probabilities of a Markov chain as it evolves.
- A hidden Markov model (HMM) is a Markov chain observed with noise.
- A partially observed Markov decision process (POMDP) is obtained by controlling the transition probabilities and/or observation probabilities of an HMM.

These relationships are illustrated in Figure 1.1.

A POMDP specializes to an MDP if the observations are noiseless. A POMDP specializes to an HMM if the control is removed. Finally, an HMM specializes to a Markov chain if the observations are noiseless.

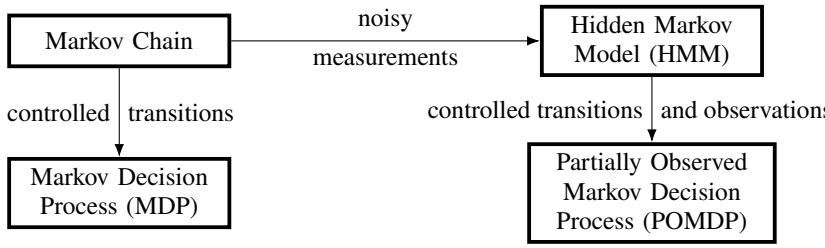


Figure 1.1 Terminology of HMMs, MDPs and POMDPs.

This introductory chapter is organized as follows:

- §1.1 to §1.5 contain a brief outline of the five parts of the book.
- §1.6 outlines some applications of controlled sensing and POMDPs.

1.1 Part I. Stochastic Models and Bayesian Inference

Part I of the book discusses Bayesian inference. Figure 1.2 illustrates the setup.

A sensor provides noisy observations $\{y_k\}$ of the evolving state $\{x_k\}$ of a Markov stochastic system where k denotes discrete time. The sample paths of the observations and underlying Markov state are shown in Figure 1.3. The Markov system, together with the noisy sensor, constitutes a partially observed Markov model, also called an HMM[1]. The aim is to estimate the state x_k at each time instant k given the observations $y_1, \ldots, y_k$.

[1] In this book, the term HMM is used for the special case when $\{x_k\}$ is a finite-state Markov chain that is observed via noisy observations $\{y_k\}$.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

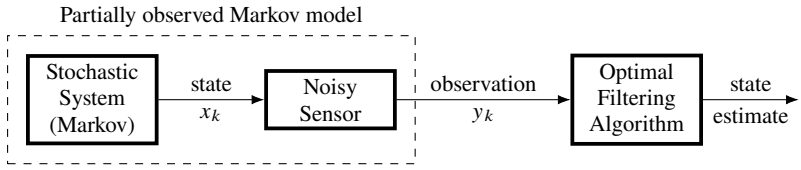


Figure 1.2 Part I discusses hidden Markov models (HMMs) and optimal filtering for state estimation. The framework is classical statistical signal processing.

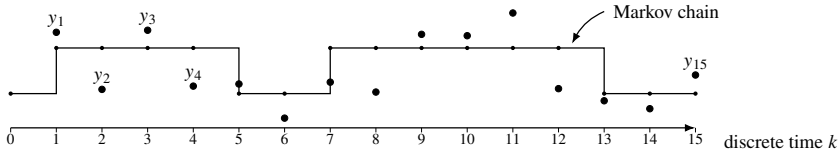


Figure 1.3 The underlying signal is a two-state Markov chain $\{x_k, k = 0, \ldots, 15\}$ shown as a piecewise constant binary waveform. A noisy sensor observes the Markov chain plus zero mean unit variance Gaussian noise. These noisy observations $\{y_k, k = 1, \ldots, 15\}$ are the HMM observations and denoted by the dots $\bullet$.

An important topic discussed in Part I is *optimal (Bayesian) filtering*. The optimal filter computes the posterior distribution π_k of the state x_k at each time k, given noisy observations $y_1, \ldots, y_k$ via the recursive algorithm

$$\boxed{\pi_k = T(\pi_{k-1}, y_k), \quad k = 1, 2, \ldots} \tag{1.1}$$

where the operator T denotes Bayes formula with a slight modification to account for the state evolution. Once the posterior π_k is evaluated, the optimal estimate (in the minimum mean square sense) of the state x_k given the noisy observations $y_1, \ldots, y_k$ can be computed by integration.

Chapters 2 and 3 cover state space models, the Kalman filter, HMM filter and suboptimal filtering algorithms such as the particle filter.

Chapter 4 discusses how the optimal filters can be used to devise numerical algorithms (general-purpose optimization algorithms and also Expectation Maximization algorithms) for maximum likelihood parameter estimation.

Chapter 5 discusses multiagent filtering over a social network. Social learning models for Bayesian decision making and data incest models are formulated. These models arise in applications such as online reputation systems and polling systems.

Finally, Chapter 6 discusses nonparametric Bayesian inference involving Dirichlet processes and Gaussian processes. In nonparametric Bayesian inference, the prior is a random process rather than a random variable.

The material in Part I is classical (to a statistical signal processing audience). However, some nontraditional topics are discussed including filtering of reciprocal processes; geometric ergodicity of the HMM filter; forward-only filters for the Expectation Maximization algorithm; multiagent filtering for social learning and data incest; slow learning with interacting Kalman filters; nonparametric Bayes, variational Bayes, and conformal prediction. Appendix B (at the end of the book) discusses continuous-time HMM filters and Markov modulated Poisson filters.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

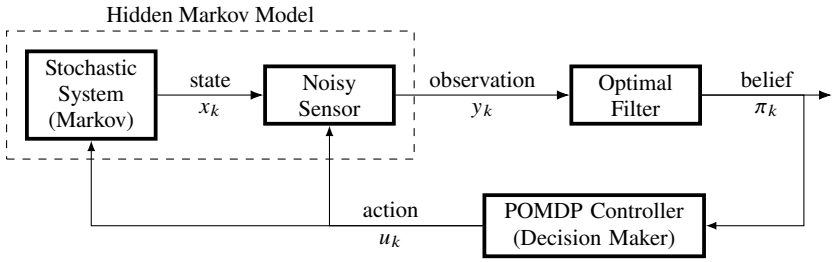


Figure 1.4 Schematic of partially observed Markov decision process (POMDP). Part II of the book discusses algorithms and applications of POMDPs where the stochastic system (Markov chain) and the sensor are controlled. Part III studies structural results: What conditions on the POMDP model ensure that the optimal policy is a monotone function of the belief? The monotone structure is then exploited to design algorithms that compute the optimal policy. Part IV discusses reinforcement learning (RL), namely, solving POMDPs when the stochastic system and sensor model are not known. Finally, Part V studies inverse RL: If an analyst observes the response of the controller, how can it estimate the cost function?

1.2 Part II. POMDPs. Models, Algorithms and Applications

Statistical signal processing (Part I) focuses on extracting signals from noisy measurements. In Parts II, III and IV of the book, motivated by physical, communication and social constraints, we address the deeper issue of how to dynamically schedule and optimize signal processing resources to extract signals from noisy measurements. These problems are formulated as POMDPs. Figure 1.4 displays the schematic setup.

Part II of the book studies the formulation, algorithms and applications of POMDPs. As in the filtering problem, at each time k, a decision maker has access to the noisy observations y_k of the state x_k of a Markov process. Given these noisy observations, the aim is to control the trajectory of the state and observation process by choosing actions u_k at each time k. The decision maker knows ahead of time that if it chooses action u_k when the system is in state x_k, then a cost $c(x_k, u_k)$ will be incurred at time k. (Of course the decision maker does not know state x_k at time k but can estimate the cost based on the observations y_k.) The goal of the decision maker is to choose the sequence of actions $u_0, \ldots, u_{N-1}$ to minimize the expected *cumulative* cost $\mathbb{E}\big\{\sum_{k=0}^{N-1} c(x_k, u_k)\big\}$ where $\mathbb{E}$ denotes mathematical expectation.

It will be shown in Part II that the optimal action u_k at each time $k = 0, 1, \ldots, N-1$ is determined by a *policy* (strategy) as $u_k = \mu_k^*(\pi_k)$ where the optimal policy μ_k^* satisfies *Bellman's backward stochastic dynamic programming equation*:

$$\boxed{\begin{aligned} \mu_k^*(\pi) &= \operatorname*{argmin}_u Q_k(\pi, u), \quad J_k(\pi) = \min_u Q_k(\pi, u), \\ Q_k(\pi, u) &= \sum_x c(x, u)\, \pi(x) + \sum_y J_{k+1}(T(\pi, y, u))\, \sigma(\pi, y, u). \end{aligned}} \tag{1.2}$$

Here T is the optimal filter (1.1) and σ is a normalization term for the filter. Also, π is the vector of posterior probabilities computed by the optimal filter (1.1) and is called the belief state.

Part II of the book studies the formulation of POMDPs and algorithms for solving Bellman's equation (1.2) along with several applications in controlled sensing.

Chapter 7 studies finite-state MDPs including constrained and entropy-regularized MDPs.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

Chapter 8 starts our formal discussion of POMDPs. The POMDP model and stochastic dynamic programming recursion are formulated in terms of the belief state computed by the optimal filter discussed in Part I. Several algorithms for solving POMDPs over a finite horizon are then presented. Optimal search theory for a moving target is used as an illustrative example.

Chapter 9 studies the formulation and applications of POMDPs in controlled sensing. Several examples are discussed: linear quadratic state and measurement control with applications in radar control, sensor scheduling using POMDPs with nonlinear costs, social learning POMDPs (where local decision makers interact with a global decision maker), and risk averse POMDPs.

1.3 Part III. POMDP Structural Results

In general, solving Bellman's dynamic programming equation (1.2) for a POMDP is computationally intractable. This is because the space of probability vectors π is a continuum. (The space of probability vectors is the unit simplex.) Part III of the book shows that by introducing assumptions on the POMDP model, important structural properties of the optimal policy can be determined without brute-force computations. These structural results can then be exploited to compute the optimal policy using stochastic gradient algorithms (reinforcement learning).

The main idea behind Part III is to give conditions on the POMDP model so that the optimal policy $\mu_k^*(\pi)$ is increasing in belief π (denoted as $\uparrow \pi$) in terms of a suitable stochastic order. This is achieved by showing that $Q_k(\pi, u)$ in Bellman's equation (1.2) is *submodular*. The main structural result is:

$$\boxed{\underbrace{Q_k(\pi, u+1) - Q_k(\pi, u) \downarrow \pi}_{\text{submodular}} \implies \underbrace{\mu_k^*(\pi) \uparrow \pi.}_{\text{increasing policy}}} \tag{1.3}$$

Obtaining conditions for $Q_k(\pi, u)$ in a POMDP to be submodular involves powerful ideas in stochastic dominance and lattice programming.

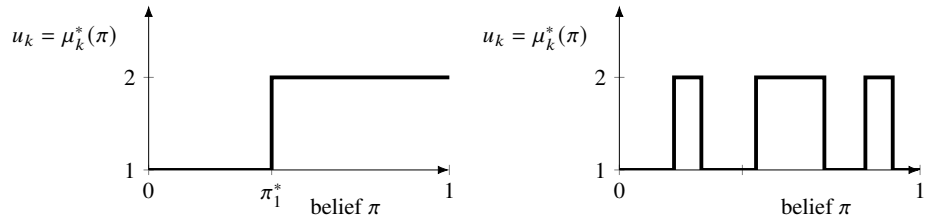
Once the optimal policy of a POMDP is shown to be monotone, this structure can be exploited to devise efficient algorithms. Figure 1.5(a) illustrates a monotone increasing optimal policy $\mu_k^*(\pi)$ in the belief π with two actions $u_k \in \{1, 2\}$. Any increasing function which takes on only two possible values must be a step function. So computing $\mu_k^*(\pi)$ in Figure 1.5(a) boils down to determining the single belief π_1^* at which the step function jumps. Estimating the threshold belief π_1^* is significantly easier than directly solving Bellman's equation (1.2) for $\mu_k^*(\pi)$ across all beliefs π, especially when $\mu_k^*(\pi)$ lacks any special structure, as shown in Figure 1.5(b).

Part III comprises six chapters (Chapters 10 to 15) devoted to structural results.

Chapter 10 gives sufficient conditions for an MDP to have a monotone (increasing) optimal policy. We also study multimodular conditions under which the value function is integer-convex. Furthermore, we show how differential sparsity of the monotone optimal policy can be exploited to compute the optimal policy. (The monotone optimal policy in Figure 1.5(a) is differentially sparse, since the derivative of the policy μ^* w.r.t. π is zero except at the single point π_1^*. In comparison, the unstructured optimal policy in Figure 1.5(b) is not differentially sparse.)

In order to provide conditions for the optimal policy of a POMDP to be monotone, it is first necessary to show monotonicity of the HMM (optimal) filter w.r.t. the prior and observations. To this end, Chapter 11 addresses the monotonicity of the HMM filter. This monotonicity result is exploited to construct reduced-complexity filtering algorithms that provide provable lower and upper bounds to the sample path estimates of the optimal filter.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press



(a) Structured Optimal Policy (b) Unstructured Optimal Policy

Figure 1.5 (a) Example of POMDP optimal policy $\mu^*(\pi)$ that is monotone (increasing) in the belief π. The optimal policy is a step function and is completely specified by the threshold state π_1^*. (b) Example of an arbitrary (unstructured) optimal policy. Part III of the book discusses sufficient conditions (involving submodularity) so that the optimal policy has a monotone structure. Then we develop algorithms that exploit this monotone structure to estimate the optimal policy. When the optimal policy is unstructured, computing the optimal policy is intractable.

Chapters 12 to 15 give conditions under which the dynamic programming recursion of a POMDP model yields a monotone solution. Chapter 12 discusses conditions for the value function in dynamic programming to be monotone. These are used to characterize the structure of two-state POMDPs and POMDP multi-armed bandits.

Chapter 13 provides conditions under which stopping-time POMDPs have monotone optimal policies. As examples, Chapter 14 covers quickest change detection, controlled social learning and a variety of other applications. The structural results provide a unifying theme and insight to what might otherwise simply be a collection of examples.

Finally, Chapter 15 gives conditions under which the optimal policy of a general POMDP can be bounded from below and above by judiciously chosen myopic policies. The chapter also discusses bounds on the sensitivity of the optimal cumulative cost of POMDPs to the parameters, and myopic policy bounds are constructed using the important concept of Blackwell dominance.

1.4 Part IV. Stochastic Gradient Algorithms and Reinforcement Learning

A key assumption in Parts I, II and III of the book is that the model of the stochastic system and noisy sensor is completely specified and known ahead of time. When this assumption does not hold, we need alternative methods. Part IV discusses *stochastic gradient algorithms* for estimating reasonable (locally optimal) strategies for POMDPs.

Suppose a decision maker can observe the noisy response y_k of a controlled stochastic system to any action u_k that it chooses. Let $\mathcal{I}_k = \{u_0, y_1, \ldots, u_{k-1}, y_k\}$ denote the history of actions and observed responses up to time k. The decision maker chooses its action as $u_k = \mu_\theta(\mathcal{I}_k)$ where μ_θ denote a parametrized policy (parametrized by a vector θ; for example the weights of a neural network). Then to optimize its policy, the decision maker needs to compute the optimal parameter θ^* which minimizes the expected cost criterion $\mathbb{E}\{C(\theta, \mathcal{I}_k)\}$. The decision maker uses the following stochastic gradient algorithm to estimate θ^*:

$$\boxed{\theta_{k+1} = \theta_k - \epsilon \nabla_\theta C(\theta_k, \mathcal{I}_k), \quad k = 0, 1, \ldots.} \tag{1.4}$$

Here $\nabla_\theta C(\theta_k, \mathcal{I}_k)$ denotes the gradient (or estimate of gradient) of the instantaneous cost with

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

respect to the parameter θ and ϵ denotes a small positive step size. Algorithms such as (1.4) lie within the class of reinforcement learning methods since the past experience $\mathcal{I}_k$ is used to adapt the parameter θ_k which in turn determines the actions; a good estimate of θ_k would result in good performance which in turn reinforces this choice at future times $k+1, k+2, \ldots$. Part IV studies such stochastic gradient algorithms, including how to compute the gradient estimate and analyze convergence of the resulting algorithm.

Chapter 16 studies gradient estimation methods, namely, how to estimate the gradient $\nabla_\theta C(\theta_k, \mathcal{I}_k)$ by stochastic simulation. This forms the basis for policy gradient reinforcement learning. Chapter 17 discusses Q-learning and policy gradient algorithms for reinforcement learning of MDPs and POMDPs. Chapter 18 presents stochastic gradient algorithms for estimating the parameters and states of an HMM (which can be used for adaptive control of a POMDP) and also mean-field dynamics of large-scale Markov chains that arise in social networks. A detailed analysis using perturbed Lyapunov functions is given for correlated noise models. Chapter 19 discusses discrete stochastic optimization algorithms, including Boltzmann exploration and the upper confidence bound multi-armed bandit algorithm.

1.5 Part V. Inverse Reinforcement Learning

Part V of the book studies inverse reinforcement learning (IRL): How can an analyst reconstruct the cost function of an MDP (or POMDP) by observing its decisions?

IRL can be framed as an abstract inverse optimization problem. Suppose we are given a time series dataset $\mathbb{D} = \{(\alpha_k, \beta_k), k = 1, 2, \ldots, N\}$ of inputs and responses of a sensor. Here, the input α_k at time k is a strictly nonnegative vector of dimension m (representing prices), and the response β_k is a nonnegative vector of dimension m (representing consumption). The question is: Under what conditions is the $\mathbb{D}$ generated by the linear constrained utility maximizer

$$\beta_k \in \underset{\alpha_k' \beta \le 1}{\operatorname{argmax}} U(\beta) \tag{1.5}$$

where $U(\beta)$ is an increasing and continuous utility function?

Afriat's theorem from microeconomics provides a necessary and sufficient condition and also constructs the set of utility functions that rationalizes $\mathbb{D}$.

Chapter 20 discusses Afriat's theorem and generalizations in inverse controlled sensing problems. The topics studied include: How to identify if a radar is cognitive? How to identify if multiple agents are coordinating their behavior (IRL for Pareto optimality)? How can a cognitive radar hide its utility from an IRL (how can a smart sensor pretend to be dumb)?

Chapter 21 discusses Bayesian IRL for inverse stopping-time problems, such as inverse sequential hypothesis testing and inverse search problems. It addresses questions like: How can we estimate a detector's cost function by observing its decisions? We also discuss inverse Bayesian filtering, namely, how one can estimate an adversary's estimate of oneself.

Summary. The three main equations described above, namely the filtering recursion (1.1), Bellman's dynamic programming equation (1.2), and the stochastic gradient algorithm (1.4), are ubiquitous. Many algorithms in statistical signal processing and control are based on these. The submodularity equation (1.3) forms the foundation for analyzing the optimal policy structure. Finally, (1.5) is a utility maximizing controlled sensor, where the utility function can be estimated and used to predict the sensor's future decisions.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

1.6 Examples of Controlled Sensing

This section outlines five applications of controlled sensing POMDPs also known as "sensor adaptive signal processing" or "active sensing".

In controlled sensing, the decision maker controls the observation noise distribution by switching between various sensors or sensing modes. Accurate sensors provide less noisy measurements but are expensive to use. Inaccurate sensors yield more noisy measurements but are cheaper. How should the decision maker decide which sensor or sensing mode to use at each time? Equivalently, how can a sensor adapt its behavior to its environment in real time? Such an active sensor uses *feedback* control. As shown in Figure 1.4, the estimates of the signal are fed to a controller/scheduler that decides how the sensor should adapt to obtain improved measurements, or alternatively, minimize a measurement cost. The design and analysis of such closed loop systems which deploy stochastic control requires decision making under uncertainty.

Example 1. Adaptive/Cognitive Radars

Cognitive radars are capable of switching between various measurement modes, e.g., radar transmit waveforms, beam pointing directions, etc., so that the tracking system is able to tell the radar which mode to use at the next measurement epoch. Instead of the operator continually changing the radar from mode to mode depending on the environment, the aim is to construct feedback control algorithms that dynamically adapt where the radar radiates its pulses to achieve the command operator objectives. This results in radars that autonomously switch beams, transmitted waveforms, target dwell and revisit times. Several sections in this book discuss controlled sensing for adaptive and cognitive radars. Chapter 20 discusses how to use inverse reinforcement learning to detect the presence of cognitive radars, and conversely, how a radar can hide its cognition from an adversary.

Example 2. Social Learning and Data Incest

A *social sensor* (human-based sensor) denotes an agent that provides information about its environment (state of nature) to a social network. Examples of such social sensors include Facebook status updates, ratings on online reputation systems like Yelp, and also decision makers that use large language models (such as GPT, LLaMA and Mixtral). Social sensors present unique challenges from a statistical estimation point of view, since they interact with and influence other social sensors. Also, due to privacy concerns, they reveal their decisions (ratings, recommendations, votes) which can be viewed as a low-resolution (quantized) function of their raw measurements.

In Chapter 5, the formalism of *social learning* will be used for modeling the interaction and dynamics of social sensors. The setup is fundamentally different from classical signal processing in which sensors use noisy observations to compute estimates – in social learning agents use noisy observations together with decisions made by previous agents, to estimate the underlying state of nature. Also, in online reputation systems such as Yelp or Tripadvisor which maintain logs of votes (actions) by agents, social learning takes place with information exchange over a graph. Data incest (misinformation propagation) occurs due to unintentional reuse of identical actions in the formation of public belief in social learning; the information gathered by each agent is mistakenly considered to be independent. This results in overconfidence and bias in estimates of the state. How can automated protocols be designed to prevent data incest and thereby maintain a fair online reputation system?

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press

Example 3. Quickest Detection with Optimal Sampling

Suppose a decision maker records measurements of a finite-state Markov chain corrupted by noise. The aim is to decide when the Markov chain jumps to a specific absorbing state. The decision maker can choose from a finite set of sampling intervals to pick the next time to look at the Markov chain. The decision maker optimizes an objective comprising false alarm, delay cost and measurement sampling cost. Making more frequent measurements yields accurate estimates but incurs a higher measurement cost. Declaring the target state too soon incurs a false alarm penalty. Waiting too long to declare the target state incurs a delay penalty. What is the optimal strategy for the decision maker?

Example 4. Interaction of Local and Global Decision Makers

How can interacting agents in a social network use their noisy observations together with decisions made by previous agents to estimate an underlying randomly evolving state? How do decisions made by previous agents affect decisions made by subsequent agents? In §14.4, these questions will be formulated as a multiagent sequential detection problem involving social learning. Individual agents record noisy observations of an underlying state process, and perform social learning to estimate the underlying state. They make local decisions about whether a change has occurred that optimizes their individual utilities. Agents then broadcast their local decisions to subsequent agents. As these local decisions accumulate over time, a global decision maker needs to decide whether or not to declare a change has occurred. How can the global decision maker achieve such change detection to minimize a cost function comprised of false alarm rate and delay penalty? The local and global decision makers interact, since the local decisions determine the posterior distribution of subsequent agents which determines the global decision (stop or continue) which determines subsequent local decisions. We discuss how a monopolist should optimally price its product when agents perform social learning. This framework also applies to controlled information fusion.

Example 5. How to Identify a Cognitive Radar?

A cognitive radar optimally adapts its waveform, aperture and dwell time subject to sensing constraints. In Part V of the book, we study inverse reinforcement learning (IRL) for cognitive radar. IRL addresses two questions: First, by observing the emissions from a radar, how to identify if it is cognitive? That is, are the radar's actions consistent with optimizing a utility function. Second, if the radar is cognitive, how to estimate its utility function and therefore predict its future actions? The IRL methods we will study transcend statistical signal processing to address the issue of how to infer strategy from sensing. The data-driven IRL approach that we will discuss is also widely used in economics (revealed preferences) and machine learning, e.g., how a robot learns by observing a human decision maker.

Conversely, we will also study, *How to design a cognitive sensor that hides its utility function from an adversary's IRL?* Obfuscating the utility is important because, if an adversary can estimate the utility function of a cognitive sensor, it can predict the sensor's sensing strategy and mitigate the sensor performance via electronic countermeasures.

https://doi.org/10.1017/9781009449441.002 Published online by Cambridge University Press