1 Introduction

We start with some terminology.

- A Markov decision process (MDP) is obtained by controlling the transition probabilities of a Markov chain as it evolves over time.
- A hidden Markov model (HMM) is a noisily observed Markov chain.
- 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 and equal to the state of the Markov chain. A POMDP specializes to an HMM if the control is removed. Finally, an HMM specializes to a Markov chain if the observations are noiseless and equal to the state of the Markov chain.

The remainder of this introductory chapter is organized as follows:

- §1.1 to §1.4 contain a brief outline of the four parts of the book.
- §1.5 outlines some applications of controlled sensing and POMDPs.

1.1 Part I: Stochastic models and Bayesian filtering

Part I of this book contains an introductory treatment of *Bayesian filtering*, also called *optimal filtering*. 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 Markov system, together with the noisy sensor, constitutes a partially observed Markov model (also called a stochastic state space model or hidden Markov model[1]). The aim is to estimate the state x_k at each time instant k given the observations $y_1, \ldots, y_k$.

Part I of the book deals with *optimal filtering*. The optimal filter computes the posterior distribution π_k of the state at time k via the recursive algorithm

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

[1] In this book, the term "hidden Markov model" is used for the special case when x_k is a finite state Markov chain that is observed via noisy observations y_k.

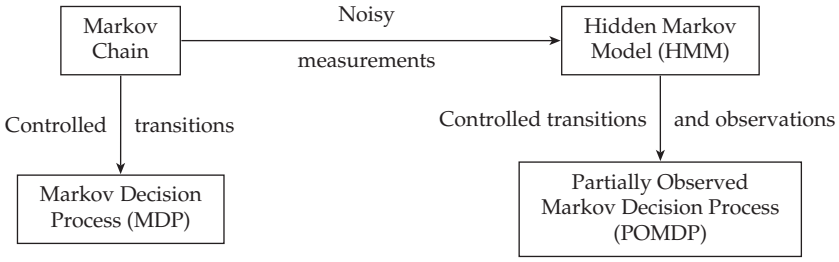

Figure 1.1 Terminology of HMMs, MDPs and POMDPs.

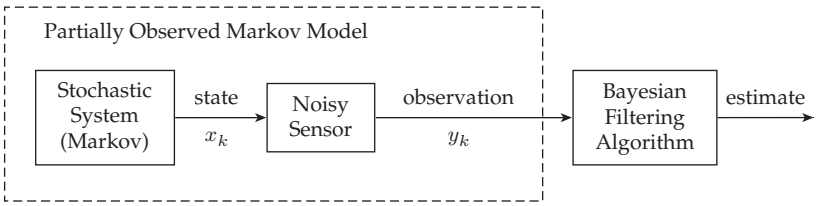

Figure 1.2 Part I deals with hidden Markov models and Bayesian filtering for state estimation. The framework is classical statistical signal processing.

where the operator T denotes Bayes' formula. 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.

Part I of the book deals with the properties of the filtering recursion (1.1). The aim is to provide in a concise manner, material essential for POMDPs.

Chapters 2 and 3 cover classical topics including state space models, the Kalman filter, hidden Markov model filter and suboptimal filtering algorithms such as the particle filter.

Chapter 4 discusses how the Bayesian 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 multi-agent filtering over a social network – social learning and data incest models are formulated; such models arise in applications such as online reputation systems and polling systems.

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; multi-agent filtering for social learning and data incest. Also, Appendix B discusses continuous-time HMM filters, Markov modulated Poisson filters, and their numerical implementation.

1.2 Part II: POMDPs: models and algorithms

Statistical signal processing (Part I) deals with extracting signals from noisy measurements. In Parts II, III and IV of the book, motivated by physical, communication and

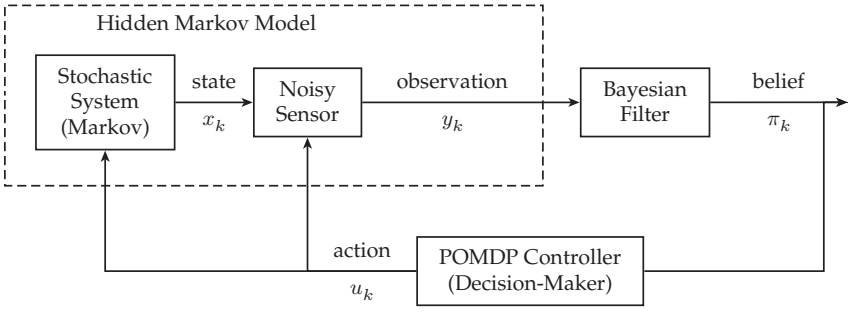

Figure 1.3 Schematic of partially observed Markov decision process (POMDP). Part II deals with algorithms and applications of POMDPs where the stochastic system (Markov Chain) and the sensor are controlled. Part III deals with determining the structure of a POMDP to ensure that the optimal action taken is a monotone function of the belief; this can result in numerically efficient algorithms to compute the optimal action (policy). Part IV deals with POMDPs when the stochastic system and sensor model are not known.

social constraints, we address the deeper issue of how to dynamically schedule and optimize signal processing resources to extract signals from noisy measurements. Such problems are formulated as POMDPs. Figure 1.3 displays the schematic setup.

Part II of the book deals with 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 $\mathbb{E}\{\sum_{k=0}^{N} c(x_k, u_k)\}$, where $\mathbb{E}$ denotes mathematical expectation.

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

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

Here T is the optimal filter (1.1) used in Part I, and σ is a normalization term for the filter. Also π is the posterior computed via the optimal filter (1.1) and is called the *belief state*.

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

Chapter 6 is a concise presentation of stochastic dynamic programming for fully observed finite-state MDPs.

Chapter 7 starts our formal presentation of POMDPs. The POMDP model and stochastic dynamic programming recursion (1.2) are formulated in terms of the belief state computed by the Bayesian 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 of a POMDP.

Chapter 8 deals with the formulation and applications of POMDPs in controlled sensing. Several examples are discussed, including linear quadratic state and measurement control with applications in radar control, sensor scheduling for POMDPs with nonlinear costs and social learning.

1.3 Part III: POMDPs: structural results

In general, solving Bellman's dynamic programming equation (1.2) for a POMDP is computationally intractable. 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 properties can then be exploited to compute the optimal policy.

The main idea behind Part III is to give conditions on the POMDP model so that the optimal policy $\mu_k^*(\pi)$ is monotone[2] in belief π. In simple terms, $\mu_k^*(\pi)$ is shown to be increasing in belief π (in terms of a suitable stocahastic ordering) by showing that $Q_k(\pi, u)$ in Bellman's equation (1.2) is *submodular*. The main 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 $Q_k(\pi, u)$ 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 1.4 illustrates an increasing optimal policy $\mu_k^*(\pi)$ in the belief π with two actions $u_k \in \{1, 2\}$. Note that any increasing function which takes on two possible values has to be a step function. So computing $\mu_k^*(\pi)$ boils down to determining the single belief π_1^* at which the step function jumps. Computing (estimating) π_1^* can be substantially easier than directly solving Bellman's equation (1.2) for $\mu_k^*(\pi)$ for all beliefs π, especially when $\mu_k^*(\pi)$ has no special structure.

Part III consists of six chapters (Chapters 9 to 14).

Chapter 9 gives sufficient conditions for a MDP to have a monotone (increasing) optimal policy. The explicit dependence of the MDPs optimal cumulative cost on transition probability is also discussed.

In order to give conditions for the optimal policy of a POMDP to be monotone, one first needs to show monotonicity of the underlying hidden Markov model filter. To this end, Chapter 10 discusses the monotonicity of Bayesian (hidden Markov model) filters.

[2] By monotone, we mean either increasing for all π or decreasing for all π. "Increasing" is used here in the weak sense; it means "non-decreasing". Similarly for decreasing.

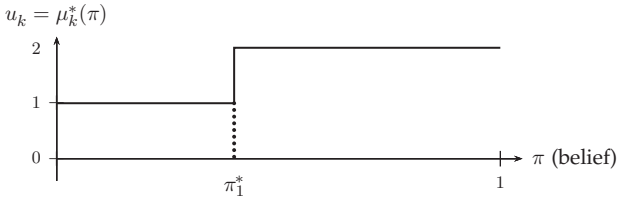

Figure 1.4 Example of optimal policy $\mu^*(\pi)$ that is monotone (increasing) in the belief π. The policy is a step function and completely characterized by the threshold state π_1^*.

This monotonicity of the optimal filter is used to construct reduced complexity filtering algorithms that provably lower- and upper-bound the optimal filter.

Chapters 11 to 14 give conditions on the POMDP model for the dynamic programming recursion to have a monotone solution. Chapter 11 discusses conditions for the value function in dynamic programming to be monotone. This is used to characterize the structure of two-state POMDPs and POMDP multi-armed bandits.

Chapter 12 gives conditions under which stopping time POMDPs have monotone optimal policies. As examples, Chapter 13 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 14 gives conditions under which the optimal policy of a general POMDP can be lower- and upper-bounded by judiciously chosen myopic policies. Bounds on the sensitivity of the optimal cumulative cost of POMDPs to the parameters are also discussed.

1.4 Part IV: Stochastic approximation and reinforcement learning

A major 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, one needs to devise alternative methods. Part IV deals with *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). Then, to optimize its choice of actions, 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 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; intuitively a good choice of θ would result in good performance which in turn reinforces this choice. Part IV deals with such stochastic gradient algorithms, including how to compute the gradient estimate and analyze the resulting algorithm.

Chapter 15 describes simulation-based gradient estimation methods that form the basis for gradient-based reinforcement learning. Chapter 16 deals with Q-learning and policy gradient algorithms for reinforcement learning. Chapter 17 presents stochastic approximation algorithms for estimating the parameters and states of a hidden Markov model (which can be used for adaptive control of a POMDP) and discrete policy search algorithms and mean field dynamics of large scale Markov chains that arise in social networks.

Remark

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 in electrical engineering. Most algorithms in statistical signal processing and control boil down to these. The submodularity equation (1.3) is the basis for analysis of the optimal policy structure.

1.5 Examples of controlled (active) sensing

This section outlines some applications of controlled sensing formulated as a POMDP. Controlled sensing also known as "sensor adaptive signal processing" or "active sensing" is a special case of a POMDP where the decision-maker (controller) controls the observation noise distribution but not the dynamics of the stochastic system. The setup is as in Figure 1.3 with the link between the controller and stochastic system omitted.

In controlled sensing, the decision-maker controls the observation noise distribution by switching between various sensors or sensing modes. An accurate sensor yields less noisy measurements but is expensive to use. An inaccurate sensor yields more noisy measurements but is cheap to use. How should the decision-maker decide at each time which sensor or sensing mode to use? Equivalently, how can a sensor be made "smart" to adapt its behavior to its environment in real time? Such an active sensor uses *feedback* control. As shown in Figure 1.3, the estimates of the signal are fed to a controller/scheduler that decides the sensor should adapt so as to obtain improved measurements; or alternatively minimize a measurement cost. Design and analysis of such closed loop systems which deploy stochastic control is nontrivial. The estimates from the signal processing algorithm are uncertain (they are posterior probability distribution functions). So controlled sensing requires decision making under uncertainty.

We now highlight some examples in controlled sensing covered in this book.

Example 1: Adaptive radars

Adaptive multifunction 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 re-visit times. §8.4 and §12.7 deal with simplified examples of radar control.

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 Twitter posts, Facebook status updates, and ratings on online reputation systems like Yelp and TripAdvisor. 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* [29, 56, 84] 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?

Example 3: Quickest detection and optimal sampling

Suppose a decision-maker records measurements of a finite-state Markov chain corrupted by noise. The goal is to decide when the Markov chain hits a specific target 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 aim is to optimize an objective comprising false alarm, delay cost and cumulative measurement sampling cost. Making more frequent measurements yields accurate estimates but incurs a higher measurement cost. Making an erroneous decision too soon incurs a false alarm penalty. Waiting too long to declare the target state incurs a delay penalty. What is the optimal sequential strategy for the decision-maker? It is shown in §13.5 that the optimal sampling problem results in a POMDP that has a monotone optimal strategy in the belief state.

Example 4: Interaction of local and global decision-makers

In a multi-agent network, how can agents use their noisy observations and 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 §13.4, these questions will be formulated as a multi-agent 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 (based on these local decisions) 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 also discuss how a monopolist should optimally price their product when agents perform social learning.

Other applications of POMDPs

POMDPs are used in numerous other domains. Some applications include:

- Optimal search: see §7.7.
- Quickest detection and other sequential detection problems: see Chapter 12.
- Dialog systems: see [350] and references therein.
- Robot navigation and planning: see [194] and references therein.
- Cognitive radio dynamic spectrum sensing: see [353] and references therein.

Website repositories

Code for POMDP solvers is freely downloadable from:

- `www.pomdp.org`
- `bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`