TECHNISCHE UNIVERSITÄT DARMSTADT

MODEL-BASED BAYESIAN INFERENCE, LEARNING, AND DECISION-MAKING WITH APPLICATIONS IN COMMUNICATION SYSTEMS

Vom Fachbereich Elektrotechnik und Informationstechnik der
TECHNISCHEN UNIVERSITÄT DARMSTADT

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

BASTIAN ALT, M.SC.
geboren am 3. November 1989 in Erbach (Odenwald).

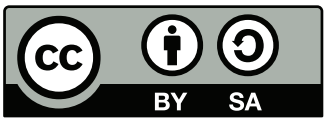
Referent: Prof. Dr. techn. Heinz Köppl
Korreferent: Prof. Dr.-Ing. Amr Rizk

Tag der Einreichung: 18. Oktober 2021
Tag der mündlichen Prüfung: 31. Januar 2022

D17
Darmstadt 2021

Die Arbeit von Bastian Alt wurde von der Deutschen Forschungsgemeinschaft (DFG) innerhalb des Sonderforschungsbereiches (SFB) 1053 „MAKI — Multi-Mechanismen-Adaption für das künftige Internet“ gefördert.

Alt, Bastian: *Model-Based Bayesian Inference, Learning, and Decision-Making with Applications in Communication Systems*
Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2022
URN: urn:nbn:de:tuda-tuprints-205151
Tag der mündlichen Prüfung: 31. Januar 2022



KURZFASSUNG

Diese Dissertation befasst sich mit der mathematischen Modellierung von dynamischen Systemen unter Unsicherheit, der Bayes'schen Inferenz und dem Lernen von unbekannten Größen, wie dem Zustand des Systems und dessen Parametern sowie der Berechnung optimaler Entscheidungen innerhalb dieser Modelle. Probabilistische dynamische Modelle erzielen erhebliche Leistungsgewinne bei dem Prozess der Entscheidungsfindung (engl.: *decision-making*). Ihre Fähigkeit den Systemzustand in Abhängigkeit der Entscheidungen zu prädizieren ermöglicht effizientes Lernen mit kleinen Datenmengen und somit werden geführte optimale Entscheidungen ermöglicht. Mehrere probabilistische Modelle für dynamische Systeme im Zustandsraum unter zeitdiskreten und zeitkontinuierlichen Annahmen werden vorgestellt. Sie bieten die Grundlage für die Berechnung der Bayes'schen Einschätzung (engl.: *belief*) und der Berechnung der optimalen Entscheidungen unter Unsicherheit. Numerische Algorithmen werden entwickelt, indem mit der exakten Systembeschreibung begonnen wird und prinzipielle Approximationen vorgenommen werden, um sowohl zu berechenbaren Algorithmen für Inferenz und Lernen, als auch für die Entscheidungsfindung zu gelangen. Die entwickelten Methoden werden anhand von Kommunikationssystemen und anderen Anwendungen demonstriert. Die spezifischen Beiträge zur Modellierung, Inferenz und Entscheidungsfindung gestalten sich inhaltlich wie folgt:

Der erste Beitrag ist eine Inferenzmethode für nicht-stationäre Punktprozessdaten, wie sie zum Beispiel bei Warteschlangen in Kommunikationssystemen üblich sind. Ein hierarchisches Bayes'sches nicht-parametrisches Modell mit einer Gamma-Verteilungsannahme für die Haltezeiten des Prozesses dient als Grundlage. Für die Inferenz wird eine berechenbare Methode auf der Grundlage eines Markov-Ketten-Monte-Carlo-Samplers hergeleitet und anschließend unter der Modellierungsannahme mittels synthetischer Daten und in einem Szenario mit Echtdaten validiert.

Der zweite Beitrag ist ein schneller Algorithmus zur Anpassung von Bitraten beim Videostreaming. Dies wird durch einen neuen Algorithmus für adaptives Bitraten-Videostreaming erreicht, der ein schwachbesetztes Bayes'sches lineares Modell für eine Quality-of-Experience-Metrik verwendet. Der Algorithmus nutzt dabei ein berechenbares Inferenzschema, um relevante Merkmale aus Netzwerkdaten zu extrahieren und baut auf einer Contextual-Bandit-Strategie für die Entscheidungsfindung auf. Neben der numerischen Validierung des Algorithmus erfolgt die Darlegung einer Implementierung und Evaluierung in einem Named-Data-Networking-Szenario.

Der dritte Beitrag ist eine neuartige Methode, die sich Korrelationen in Entscheidungsproblemen zu Nutzen macht. Die zugrundeliegenden Modellparameter können sehr dateneffizient inferiert werden, basierend auf einem Bayes'sches Modell für korrelierte Zähldaten in Markov-Entscheidungsprozessen. Um Intraktabilitäten, die bei der exakten Bayes'schen Inferenz auftreten, zu überwinden, wird ein berechenbarer Variationsinferenz-Algorithmus

vorgestellt, der ein Augmentierungsschema nutzt. Die Methode wird ausgiebig in verschiedenen Entscheidungsszenarien evaluiert, wie beispielsweise bei dem Reinforcement-Learning in einem Warteschlangensystem.

Der letzte Beitrag befasst sich mit gleichzeitiger Zustandsinferenz und Entscheidungsfindung in zeitkontinuierlichen, teilobservierbaren Umfeldern. Dabei wird ein neues Modell für diskrete Zustands- und Aktionsraumsysteme vorgestellt, wobei eine Besprechung der entsprechenden Gleichungen für exakte Bayes'sche Inferenz erfolgt. Die Optimalitätsbedingungen für die Entscheidungsfindung werden hergeleitet. Zusätlich werden zwei numerische Verfahren vorgestellt, die Funktionsapproximatoren nutzen, um die Lösung im *Belief*-Raum zu lernen. Schließlich erflogt die Vorstellung der Anwendbarkeit der Methode anhand mehrerer Beispiele, einschließlich eines Scheduling-Algorithmus unter Teilobservierbarkeit.

ABSTRACT

This dissertation discusses the mathematical modeling of dynamical systems under uncertainty, Bayesian inference and learning of the unknown quantities, such as the system's state and its parameters, and computing optimal decisions within these models. Probabilistic dynamical models achieve substantial performance gains for decision-making. Their ability to predict the system state depending on the decisions enables efficient learning with small amounts of data, and therefore make guided optimal decisions possible. Multiple probabilistic models for dynamical state-space systems under discrete-time and continuous-time assumptions are presented. They provide the basis to compute Bayesian beliefs and optimal decisions under uncertainty. Numerical algorithms are developed, by starting with the exact system description and making principled approximations to arrive at tractable algorithms for both inference and learning, as well as decision-making. The developed methods are showcased on communication systems and other commonplace applications. The specific contributions to modeling, inference and decision-making are outlined in the following.

The first contribution is an inference method for non-stationary point process data, which is common, for example, in queues within communication systems. A hierarchical Bayesian non-parametric model with a gamma-distributional assumption on the holding times of the process serves as a basis. For inference, a computationally tractable method based on a Markov chain Monte Carlo sampler is derived and subsequently validated under the modeling assumption using synthetic data and in a real-data scenario.

The second contribution is a fast algorithm for adapting bitrates in video streaming. This is achieved by a new algorithm for adaptive bitrate video streaming that uses a sparse Bayesian linear model for a quality-of-experience score. The algorithm uses a tractable inference scheme to extract relevant features from network data and builds on a contextual bandit strategy for decision making. The algorithm is validated numerically and an implementation and evaluation in a named data networking scenario is given.

The third contribution is a novel method that exploits correlations in decision-making problems. Underlying model parameters can be inferred very data-efficiently, by building a Bayesian model for correlated count data from Markov decision processes. To overcome intractabilities arising in exact Bayesian inference, a tractable variational inference algorithm is presented exploiting an augmentation scheme. The method is extensively evaluated in various decision-making scenarios, such as, reinforcement learning in a queueing system.

The final contribution is concerned with simultaneous state inference and decision-making in continuous-time partially observed environments. A new model for discrete state and action space systems is presented and the corresponding equations for exact Bayesian inference are discussed. The optimality conditions for decision-making are derived. Two tractable numerical schemes are presented, which exploit function approximators to learn the solution in the belief space. Applicability of the method is shown on several examples, including a scheduling algorithm under partial observability.

PUBLICATIONS

The following works were published during the period of the doctoral candidacy:

PEER-REVIEWED CONFERENCE PROCEEDINGS

[1] B. Alt, M. Schultheis, and H. Koeppl, "POMDPs in continuous time and discrete spaces", in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 13 151–13 162

[2] B. Alt, A. Šošić, and H. Koeppl, "Correlation priors for reinforcement learning", in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 14 155–14 165

[3] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual quality adaptation for adaptive bitrate video streaming", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1000–1008

[4] B. Alt, M. Messer, J. Roeper, G. Schneider, and H. Koeppl, "Non-parametric Bayesian inference for change point detection in neural spike trains", in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, 2018, pp. 258–262

[5] L. Köhs, B. Alt, and H. Koeppl, "Variational inference for continuous-time switching dynamical systems", in *Advances in Neural Information Processing Systems*, vol. 34, 2021, to appear, arXiv:2109.14492 [cs.LG]

[6] W. R. KhudaBukhsh, B. Alt, S. Kar, A. Rizk, and H. Koeppl, "Collaborative uploading in heterogeneous networks: Optimal and adaptive strategies", in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 1–9

PEER-REVIEWED JOURNAL ARTICLES

[7] B. Alt, M. Weckesser, C. Becker, M. Hollick, S. Kar, A. Klein, R. Klose, R. Kluge, H. Koeppl, B. Koldehofe, *et al.*, "Transitions: A protocol-independent view of the future internet", *Proceedings of the IEEE*, vol. 107, no. 4, pp. 835–846, 2019

[8] S. Kar, R. Rehrmann, A. Mukhopadhyay, B. Alt, F. Ciucu, H. Koeppl, C. Binnig, and A. Rizk, "On the throughput optimization in large-scale batch-processing systems", *Performance Evaluation*, vol. 144, p. 102 142, 2020

[9] W. R. KhudaBukhsh, S. Kar, B. Alt, A. Rizk, and H. Koeppl, "Generalized cost-based job scheduling in very large heterogeneous cluster systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2594–2604, 2020

ADDITIONAL PEER-REVIEWED PUBLICATIONS

The following works were produced prior to the doctoral candidacy:

[10] J. Machkour, M. Muma, B. Alt, and A. M. Zoubir, "A robust adaptive Lasso estimator for the independent contamination model", *Signal Processing*, vol. 174, p. 107 608, 2020

[11] J. Machkour, B. Alt, M. Muma, and A. M. Zoubir, "The outlier-corrected-data-adaptive Lasso: A new robust estimator for the independent contamination model", in *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, 2017, pp. 1649–1653

ACKNOWLEDGEMENTS

This thesis is the product of years of work, in which I learned so much, met so many exceptional people, and made great friendships. I want to say thank you!

First, I would like to express my sincere gratitude to my supervisor Prof. Heinz Koeppl, for giving me the opportunity to work together with so many great people in and out of his research lab, letting me pursue my research interests, and supporting and guiding me during my doctoral studies.

A special thanks goes to Prof. Amr Rizk for his support and guidance, not only but especially during the first years of my doctoral candidacy.

I would like to thank the German research foundation (DFG) and the collaborative research center (SFB) 1053 “MAKI — Multi-Mechanisms Adaptation for the Future Internet” for funding my work. I always felt very comfortable in MAKI, and I would like to say thank you to the whole team, with Prof. Ralf Steinmetz and all its other people involved.

I want to acknowledge all the people who were involved in my early years at TU Darmstadt - Prof. Abdelhak M. Zoubir, Andreas Ring, Christian Sledz, Hauke Radtki, Jasin Machkour, Maximilian Hüttenrauch, Michael Fauss, Michael Muma, Patrick Wenzel, Stefan Vlaski, Thomas Kübert and Tim Schäck.

Research is definitely not a field of lone wolves - I would like to thank all my collaborators - Adrian Šošić, Anam Tahir, Prof. Gaby Schneider, Lukas Köhs, Matthias Schultheis, Michael Messer, Sounak Kar, Trevor Ballard and Wasiur Rahman Khuda Bukhsh.

Now, to all the people from the BCS team not yet mentioned - Ahmed Elshamanhory, Alina Kuzembayeva, Christian Wildner, Christiane Hübner, Christine Cramer, Derya Altintan, Dominik Linzner, Felix Reinhardt, François Lehr, Gamze Dogali, Gizem Ekinci, Hongfei Liu, Jascha Diemer, Kai Cui, Kilian Heck, Klaus-Dieter Voss, Leo Bronstein, Maik Molderings, Maleen Hanst, Mark Sinzger, Markus Baier, Markus Röder, Megan Bailey, Melanie Mikosch-Wersching, Mengguang Li, Michael Schmidt, Miloš Lješković, Nicolai Engelmann, Nikita Kruk, Nurgazy Sulaimanov, Özdemir Cetin, Philipp Fröhlich, Rogier Schoeman, Sara Al-Sayed, Sikun Yang, Sofia Startceva, Stanislav Stepaniuk, Sunil Kumar, Tabea Treppmann, Tim Prangemeier and Yujie Zhong - Thank you!

I want to thank all my friends for helping me through all the good and bad times.

Julia - Thank you so much for all your support!

Finally, I want to thank my whole family and thank you, mom and dad, for always being there for me!

To everybody I forgot - Thank you!

Darmstadt, October 18, 2021

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Mathematical models have a long history in the natural sciences and are the basis of modern-day engineering systems. Models enable to generalize for unseen effects, and therefore to reason about what might happen in the future. They can guide agents (the decision-makers) in making decisions, to both achieve objectives and at the same time learn about a system at hand. The model-based decision process is depicted schematically in Fig. 1.1. To illustrate the decision-making process under uncertainty, consider the following simple example, of walking to the fridge at night to get a snack. A trained agent may happily accept the extra effort of switching on a light, to more readily find the fridge and mitigate possible risks. The agent does this to increase the certainty or its *belief* of its model of the kitchen, enabling a faster and safer path to the objective. Generally, by using a probabilistic model for sensor data and how decisions influence the system, one can find plans which achieve (i) the goal of learning or inferring the model and (ii) accomplish optimal decisions, which is also discussed under the names of Bayesian reinforcement learning (BRL) or dual control [12–14].

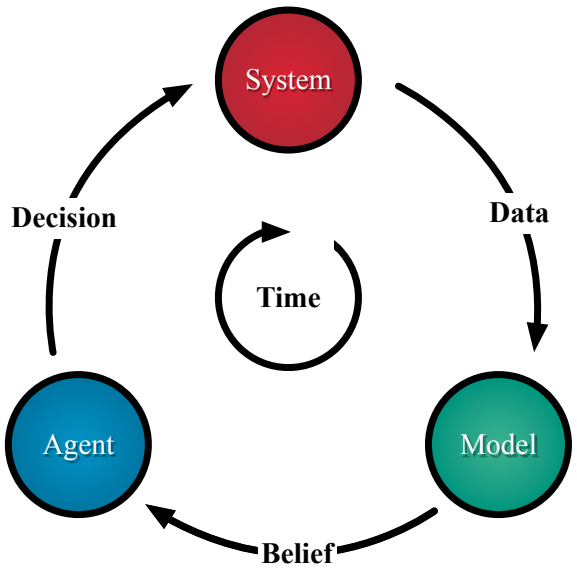


FIGURE 1.1: Schematic of model-based decision-making under uncertainty. The *agent* influences the *system* sequentially over time by its *decisions*. By obtaining *data*, the *model* can be learned, and a *belief* over specific properties of the system can guide the agent in its decision-making.

This thesis focuses on calculating beliefs with Bayesian inference and how to make (near) optimal decisions for a large class of decision-making problems. The key questions which occur in this context are: (i) How can we specify probabilistic models for real systems under uncertainty? (ii) How can we computationally efficiently calculate beliefs or infer unknown quantities of the systems? (iii) How should we make and compute optimal decisions under uncertainty?

This thesis answers these questions by building on methods from machine learning, statistics, optimization, and control theory. Numerous application scenarios for the proposed methods are considered, with a particular focus on communication systems in computer networking. Example applications include decision-making in queueing networks, design of optimal scheduling algorithms, or how to decide sequentially for a quality level in a video stream. However, the ideas, models and solution methods presented in this thesis are by no means limited to these examples and can be applied to a vast range of topics.

1.1 MOTIVATION

Automated decision-making is one of the key challenges for intelligent systems. When building decision-making schemes, model-free techniques often need a massive amount of training data to obtain a good policy since they purely learn by trial and error. This limits their use in many applications, as we wish to make guided decisions, which obtain suitable results with as little training data as possible. A promising data-efficient direction, as opposed to a model-free method, is a model-based method. Building models can guide the agent in its decision-making and are therefore superior to model-free methods for systems, where the modeling assumptions hold. Though it is persistently hard to specify all parameters in a model completely, we still might want to incorporate as much knowledge into the model as possible, as it will lift a burden from the decision-making agent. Free unknown parameters can then be learned sequentially from the data obtained. Since a problematic aspect in training such models is that a naive greedy estimation of the system parameters will lead to overconfident agents who *exploit* their knowledge of the system too much, we will use a Bayesian approach for inference instead. Bayesian methods provide (i) a natural way to specify prior knowledge and (ii) they provide a degree of uncertainty for the decision-making strategy enabling a balance of *exploration* and *exploitation*.

Applications that benefit from such methods are numerous and can include many decision-making processes, such as communication systems, recommender systems, the control of biological systems, robotics, etc. In the context of this thesis the application of decision-making and inference in communication systems will be a reoccurring theme. These systems are often prime examples for decision-making under uncertainty as many environmental effects can influence the systems, such as for example, cross-traffic in computer networks. Queues are helpful and common models for systems in this regard. Here, the system state can be described by a discrete number of packets in the queue. Other systems require a different description using a continuous state, such as the current round trip time in a scheduling system. Hence, we will specifically discuss both of these modeling aspects in this thesis. Another aspect is how to model the decision-making process in time. We will specifically consider both the cases of episodic (discrete-time) decision-making

and continuous-time decision-making and will choose a suitable description problem dependently.

Given this context, the contributions of this thesis are listed next, which address and solve the previously discussed challenges.

1.2 CONTRIBUTION AND OVERVIEW

This thesis aims to develop probabilistic models, tractable inference and learning algorithms, and optimal decision-making schemes. The main contributions are the following:

1. A model and a tractable inference algorithm for point process data in continuous time is developed. Here, the inference of a discrete latent state process modulating the rate of the point process is considered.
2. A fast episodic decision-making algorithm in a video streaming setting is derived. For this, the discrete-time system state is modeled using continuous-valued features, which probabilistically determine the objective to be optimized.
3. A new probabilistic correlation model for decision-making is designed. In the presented discrete-time setting, the level of the underlying latent correlations is inferred from the observed discrete state data.
4. The general problem of partially observable decision-making with discrete states in continuous time is discussed. A new model is provided, the optimality conditions are derived, and numerical solution algorithms are presented for the latent state inference and the decision-making.

For an overview of these contributions see Table 1.1. An exhaustive list of our publications can be found in the front matter of this thesis on Pages vii to viii.

The thesis is structured in the following chapters.

CHAPTER 2. This background chapter gives some preliminaries on the underlying mathematical ideas, modeling techniques, and numerical solution methods. In the context of this thesis, some foundational related work is given, and the central machinery of Bayesian inference and optimal decision-making is discussed.

CHAPTER 3. This chapter discusses a model for non-stationary continuous-time point processes. These types of models can be used for describing discrete events at continuous-valued time stamps. An important example in communication systems is a queueing system, where we model the arrival time of a packet. Still, we will also look at a different instance from neuroscience, where we want to model the firing times of neurons in a brain. To model the non-stationarity, a discrete-state continuous-time latent process modulating the rate is assumed. The regimes of different rates are modeled using a Bayesian non-parametric prior and the underlying state and parameters of the process are inferred using a tractable

Contribution	Time	State	Inference	Decision-Making
1. Chapter 3: BAYESIAN NON-PARAMETRIC INFERENCE FOR A CONTINUOUS-TIME POINT PROCESS MODEL	C	D	✓	✗
2. Chapter 4: DECISION-MAKING WITH A SPARSE BAYESIAN MODEL IN VIDEO STREAMING	D	C	✓	✓
3. Chapter 5: EXPLOITING BAYESIAN CORRELATION MODELS IN DECISION-MAKING	D	D	✓	✓
4. Chapter 6: OPTIMAL DECISION-MAKING IN CONTINUOUS TIME AND DISCRETE SPACES WITH BAYESIAN STATE INFERENCE	C	D	✓	✓

TABLE 1.1: An overview of the contributions. The table shows different discussed model-based methods for decision-making and inference, and learning. The models use for this a description in continuous time (C) or in discrete time (D). Similar, the state of the system at hand is described by a value in a continuous state-space (C) or a discrete state-space (D).

sampling method. The decision-making aspect in similar models is then discussed later on in Chapter 6. This chapter is based on the published work

[4] B. Alt, M. Messer, J. Roeper, G. Schneider, and H. Koeppl, "Non-parametric Bayesian inference for change point detection in neural spike trains", in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, 2018, pp. 258–262.

CHAPTER 4. In this chapter, we consider a fast decision-making algorithm to adjust the bitrate in video streaming. The system features are modeled using a Bayesian model, which extracts essential features from the data. For inference, multiple scalable schemes are developed based on variational inference. We discuss approximately optimal decision-making using a bandit heuristic, which considers the uncertainty over parameters of the model. This fast algorithm is then evaluated extensively in the setting of adaptive bitrate video streaming. This chapter is based on the published work

[3] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual quality adaptation for adaptive bitrate video streaming", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1000–1008.

CHAPTER 5. This chapter describes a method to model correlations in decision-making problems with discrete state and action spaces. A model for correlations in several aspects of decision-making is given. A new tractable variational inference scheme based on a Pólya-Gamma (PG) augmentation is discussed. Applications include, e.g., the control of a queueing network with unknown system parameters. This chapter is based on the published work

[2] B. Alt, A. Šošić, and H. Koeppl, "Correlation priors for reinforcement learning", in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 14 155–14 165.

CHAPTER 6. The final content chapter discusses the general problem of partially observable decision-making in discrete state spaces in continuous time. A new model, the equations for exact inference, and optimality conditions for decision-making are provided. Additionally, some numerical algorithms are developed which solve the decision-making problem using techniques from deep reinforcement learning. The method is evaluated on several examples, including a continuous-time adaptation of the slotted aloha scheduling algorithm. This chapter is based on the published work

[1] B. Alt, M. Schultheis, and H. Koeppl, "POMDPs in continuous time and discrete spaces", in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 13 151–13 162.

CHAPTER 7. This chapter gives a final summary of the discussed contributions, and an outlook on future research directions based on this work.

APPENDICES. In Appendices A and B some additional content of this thesis is provided. In Appendix A some probabilistic notation and definitions are given and in Appendix B some additional results are presented, which have been used in this thesis.

2

BACKGROUND

This chapter introduces mathematical notation and discusses established concepts from probability, statistical inference, and decision-making.

2.1 PROBABILISTIC MODELING

Throughout this thesis, we use probability theory extensively to build models for the desired systems. Therefore, we now give some probabilistic modeling concepts, which we will use in this thesis. We note that basic knowledge of probability theory suffices for the major part of this thesis, however, some definitions and notational conventions can be found in Appendix A.1. Additionally, probability distributions used in this thesis are given in Appendix A.2 and a list of special functions used in this thesis are provided in Appendix A.3.

PROBABILISTIC GRAPHICAL MODELS. An essential class of probabilistic models can be described by the use of probabilistic graphical models (PGMs) [15]. Within this framework, we express conditional independence statements in the form of a graph. Consider, for example, the distribution

$$p(x, y, z) = p(x)p(y \mid x)p(z \mid y),$$

the corresponding graphical model is depicted in Fig. 2.1. Throughout, this thesis we will

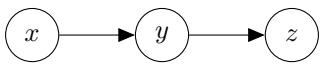


FIGURE 2.1: Example for a PGM.

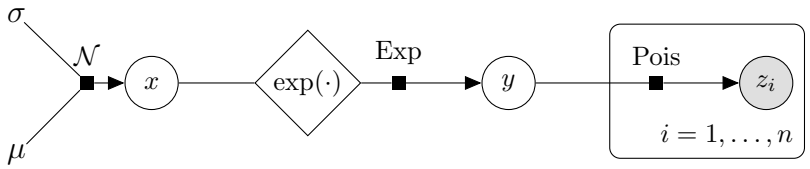


FIGURE 2.2: Example for a PGM in factor graph notation. Here, diamond-shaped nodes denote deterministic functions, and factors indicate distributions.

mainly use a representation in a factor graph notation [16]. For this, consider, for example, a distribution given as

$$p(x, y, \mathbf{z}) = \mathcal{N}(x \mid \mu, \sigma^2) \operatorname{Exp}(y \mid \exp(x)) \prod_{n=1}^{N} \operatorname{Pois}(z_n \mid y), \tag{2.1.1}$$

where the normally distributed x is the log-rate of the exponentially distributed y, which correspond to the rate of the i.i.d. Poisson variables $\mathbf{z} = [z_1, \ldots, z_n]^\top$. If we now consider that we observed the variables $\mathbf{z}$, we can express the distribution in Eq. (2.1.1) by the PGM in Fig. 2.2 using a factor graph notation.

2.1.1 *Continuous-Time Models*

Next, we discuss some fundamentals of continuous-time models. These are models which provide a probabilistic description of a stochastic process $\{x(t) \mid t \in \mathbb{R}_{\geq 0}\}$, which is a collection of random variables (RVs) indexed by a real-valued time stamp.

STOCHASTIC DIFFERENTIAL EQUATIONS. First, we discuss the class of stochastic differential equations (SDEs) [17]. An SDE is a Markov model on a continuous state space; i.e., $\mathbf{x}(t) \in \mathcal{X} \subseteq \mathbb{R}^n$. Its evolution can be described by an equation of the form

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)\,\mathrm{d}t + \mathbf{G}(\mathbf{x}(t), t)\,\mathrm{d}\mathbf{w}(t), \tag{2.1.2}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state at time t and $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{G}(\mathbf{x}, t)$ are the drift function and dispersion function evaluated at state $\mathbf{x}$ and time t, respectively. Here, $\mathbf{w}(t)$ is an n-dimensional standard Brownian motion, which has the defining properties: (i) It starts at the origin, $\mathbf{w}(0) = \mathbf{0}$, (ii) its increments $\Delta\mathbf{w} = \mathbf{w}(t) - \mathbf{w}(s)$, with $0 \leq s \leq t$ are Gaussian distributed; i.e., $\Delta\mathbf{w} \sim \mathcal{N}(\Delta\mathbf{w} \mid 0, (t-s)\mathbf{I})$ and (iii) non-overlapping increments are independent.

Generally, Eq. (2.1.2) is a short-form for the integral representation

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}(s), s)\,\mathrm{d}s + \int_0^t \mathbf{G}(\mathbf{x}(s), s)\,\mathrm{d}\mathbf{w}(s),$$

where the second integral w.r.t. Brownian motion is the Itô integral defined as

$$\int_0^t \mathbf{G}(\mathbf{x}(s), s)\,\mathrm{d}\mathbf{w}(s) \coloneqq \lim_{n\to\infty} \sum_k \mathbf{G}(\mathbf{x}(t_k), t_k)\left[\mathbf{w}(t_{k+1}) - \mathbf{w}(t_k)\right],$$

with $t_0 < t_1 \cdots < t_n = t$. This gives rise to the so-called Itô calculus, for more see [17, 18].

THE POISSON PROCESS. Another important continuous-time model is the Poisson process [19]. In this model, we consider the Poisson counting process $\{N(t) \mid t \in \mathbb{R}_{\geq 0}\}$, which has the properties: (i) It starts at the origin, $N(0) = 0$, (ii) for each t, $N(t)$ is Poisson distributed with rate parameter λt and (iii) non-overlapping increments are independent. Additionally, as a short-hand for the distribution of the process $\{N(t) \mid t \in \mathbb{R}_{\geq 0}\}$ we write $N(t) \sim \mathcal{PP}(N(t) \mid \lambda)$.

Similar to the case of an SDE it is helpful to define a differential equation with a counting process on the r.h.s., e.g.,

$$\mathrm{d}x(t) = h(x(t), t)\,\mathrm{d}N(t), \tag{2.1.3}$$

where $N(t) = \sum_{n=1}^{N} \mathbb{1}(t \leq t_n)$ is a counting process, with jump times $\{t_n \mid n = 1, \ldots, N\}$ and $h(x, t)$ is the jump amplitude for state x at time t. This yields the integral form

$$x(t) = x(0) + \int_0^t h(x(s), s)\,\mathrm{d}N(s).$$

A sensible definition for this integral w.r.t. the counting process is

$$\int_0^t h(x(s), s)\,\mathrm{d}N(s) \coloneqq \sum_{n=1}^{N(t)} h(x(t_n^-), t_n^-),$$

where t_n^- denotes the time just before the nth jump-time t_n of $N(t)$; i.e., $t_n^- = \lim_{h \to 0} t_n - h$. Generally, $N(t)$ can be any counting process, however, most often we consider $N(t)$ to be a Poisson process, which makes Eq. (2.1.3) an SDE driven by a Poisson process, for more see, e.g., [20, 21].

CONTINUOUS-TIME MARKOV CHAINS. The Poisson counting process can be seen as a simple example of a continuous-time Markov chain (CTMC). Generally, a CTMC $\{x(t) \mid t \in \mathbb{R}_{\geq 0}\}$, with $x(t) \in \mathcal{X}$, is a Markov model on a discrete state space, e.g., $\mathcal{X} \subseteq \mathbb{N}$ and in continuous time $t \in \mathbb{R}_{\geq 0}$. Its evolution for a short time step h is characterized by

$$\mathsf{P}(X(t + h) = x \mid X(t) = x') = \Lambda(x', x)h + o(h),$$

for all $x \neq x'$ and $\lim_{h \to 0} \frac{o(h)}{h} = 0$. Here, $\Lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the rate function, where $\Lambda(x', x)$ denotes the rate of switching from state x' to state x. We define the exit rate of a state $x \in \mathcal{X}$ as $\Lambda(x) \coloneqq \sum_{x' \in \mathcal{X} \setminus \{x\}} \Lambda(x, x')$ and the diagonal element as $\Lambda(x, x) \coloneqq -\Lambda(x)$. Note that, due to the memoryless property of the Markov process, the waiting times between two jumps of the chain is exponentially distributed. The definition can also be extended to time-dependent rate functions, which correspond to a time inhomogeneous CTMC, for further details see [19].

2.2 BAYESIAN INFERENCE

For the purpose of inference, we will largely follow the Bayesian viewpoint. Under this viewpoint, we express our estimates as beliefs over unknown parameters. Using Bayes'

rule, we can condition on data and incorporate observed quantities to update our belief; this is known as *Bayesian inference*, for more see, e.g., [22–24].

As illustration, consider for example a simple model, where the data $\mathcal{D} = \{x_n \mid n = 1, \dots, N\}$ is generated as

$$x_n \sim p(x_n \mid \theta) \quad n = 1, \dots, N$$

This yields the *likelihood*

$$p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} p(x_n \mid \theta).$$

If we want now to estimate the unknown parameters θ, we can express our initial belief using a *prior* distribution $p(\theta)$. Using Bayes' rule

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}, \tag{2.2.4}$$

we find the *posterior* distribution $p(\theta \mid \mathcal{D})$, which expresses our belief as a distribution after observing the data, by conditioning on it.

Frequently, a problem with Bayesian inference is to calculate the *evidence*

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta) \, \mathrm{d}\theta, \tag{2.2.5}$$

since we have to integrate over all possible parameters $\theta \in \Theta$. If the space Θ is large, Eq. (2.2.5) can not be solved numerically, which makes the exact posterior update in Eq. (2.2.4) intractable for all but analytically solvable models. Unfortunately, the class of models where we can compute the posterior distribution exactly is rather small. Therefore, we often have to find an approximation to the posterior distribution, which is tractable.

2.2.1 *Approximate Bayesian Inference*

MARKOV CHAIN MONTE CARLO. One of the most used approximation methods for Bayesian inference is Markov chain Monte Carlo (MCMC) [25]. We approximate the exact posterior distribution by sampling from the true posterior distribution (Monte Carlo). For a lot of sampling techniques, the intractable evidence is not necessary to compute.

One example for MCMC is the *Metropolis-Hastings algorithm*. In this algorithm, we sample from the true distribution by performing a random walk in the parameter space, which can be seen as a Markov chain. We reject and accept the resulting samples by calculating an acceptance probability. This yields samples from the true posterior distribution, which can be used to compute, e.g., an empirical estimate of the posterior distribution.

Another method, which we will exploit is *Gibbs sampling*, which is also a form of MCMC. Gibbs sampling is performed by iterating between sampling from the *full conditionals* of the joint distribution over model parameters and data. For further details on these and other sampling-based methods, see, e.g., [22, 24, 25].

VARIATIONAL INFERENCE. Another alternative scheme is to use a deterministic approximation to the true posterior distribution. Within this line of methods, one popular approach is variational inference (VI) [26]. In VI, we approximate the true posterior distribution $p(\theta \mid \mathcal{D})$ by an approximating distribution $q(\theta)$. As the objective function, we select in this thesis the Kullback–Leibler (KL) divergence between the variational distribution and the intractable posterior distribution; i.e., we aim to solve the optimization problem

$$\underset{q}{\text{minimize}} \quad \mathsf{KL}(q(\theta) \parallel p(\theta \mid \mathcal{D})). \tag{2.2.6}$$

This KL divergence computes to

$$\mathsf{KL}(q(\theta) \parallel p(\theta \mid \mathcal{D})) = \mathsf{E}[\log q(\theta)] - \mathsf{E}[\log p(\mathcal{D}, \theta)] + \log p(\mathcal{D}),$$

where the expectations are carried out w.r.t. the variational distribution $q(\theta)$. By Jensen's inequality, a lower bound on the marginal likelihood (evidence) can then be found as

$$\mathsf{L}[q] = \mathsf{E}[\log p(\mathcal{D}, \theta)] - \mathsf{E}[\log q(\theta)] \leq \log p(\mathcal{D}).$$

We can use the evidence lower bound (ELBO) $\mathsf{L}[q]$ in an equivalent optimization problem to Eq. (2.2.6); i.e.,

$$\underset{q}{\text{maximize}} \quad \mathsf{L}[q]. \tag{2.2.7}$$

The problem in Eq. (2.2.7) is tractable compared to computing the KL divergence in Eq. (2.2.6), since for calculating $\mathsf{L}[q]$ it is not required to compute the intractable log-evidence $\log p(\mathcal{D})$.

Note that, to arrive at a tractable solution for the optimization problem in Eq. (2.2.7), we have to constrain the class of variational distributions further. In this thesis, we will mainly use the approach of mean-field VI, where we assume independence between components of the parameters θ.

2.3 DECISION-MAKING: OPTIMAL CONTROL AND REINFORCEMENT LEARNING

For decision-making, we will focus on optimization-based approaches. In control theory, this is known as optimal control (OC) [13, 27]. In OC, we consider a model, which describes the dynamical evolution of a *state* $x(t) \in \mathcal{X}$ at time $t \in \mathcal{T}$ which can be altered by applying an *action* or *control* signal $\{u(t) \mid t \in \mathcal{T}\}$, with $u(t) \in \mathcal{U}$ to it. A decision-making agent aims to maximize a *reward* function $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ over a time horizon. In this framework, future effects on the state signal have to be incorporated by *planning* for the whole action signal $\{u(t) \mid t \in \mathcal{T}\}$. Mathematically, depending on whether we have a continuous-time model $t \in \mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ or a discrete-time model $t \in \mathcal{T} \subseteq \mathbb{N}_{\geq 0}$ we can write down the optimization problems, as

$$\begin{aligned} &\underset{u_{[0,\infty]}}{\text{maximize}} \quad J\left[u_{[0,\infty]}\right] = \mathsf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(x(t), u(t))\right] \\ &\text{subject to} \quad x(t+1) \sim p(x(t+1) \mid x(t), u(t)), \quad \forall t \in \{0, 1, \dots\} \end{aligned} \tag{2.3.8}$$

for discrete-time problems and for continuous-time problems we have

$$\begin{aligned} \underset{u_{[0,\infty]}}{\text{maximize}} \quad & J\left[u_{[0,\infty]}\right] = \mathsf{E}\left[\int_0^\infty e^{-\frac{t}{\tau}} R(x(t), u(t))\,\mathrm{d}t\right] \\ \text{subject to} \quad & x(t+h) \sim p(x(t+h) \mid x(t), u(t)), \quad \forall t \in \mathbb{R}_{\geq 0}, \end{aligned} \tag{2.3.9}$$

where h denotes a small time step.

The OC problems in Eqs. (2.3.8) and (2.3.9), which are given for an infinite time horizon, are optimized over the action trajectory $u_{[0,\infty]} \coloneqq \{u(t) \mid t \in \mathcal{T}\}$. This yields for the discrete-time action trajectory $u_{[0,\infty]} = \{u(0), u(1), \dots\}$ and for the continuous-time action trajectory $u_{[0,\infty]} = \{u(t) \mid t \in \mathbb{R}_{\geq 0}\}$. In Eqs. (2.3.8) and (2.3.9) we introduce a discount factor γ for the discrete-time problem and an inverse exponential discount rate $1/\tau$ for the continuous-time problem to ensure convergence of the sum or integral, respectively. Additionally, this models the intention that earlier rewards are more important than future ones. As it can be harder to control the future uncertain system state, we value the earlier rewards higher than the latter ones. Note that we choose both discounts as further calculations will be simplified for respective problems. Generally, these discount factors are equivalent as we can quickly transform one into another, as $1/\tau = \rho = -\log\gamma$. For the evolution of the state $x(t)$, we assume a Markovian dynamic for both OC problems. This corresponds for the discrete-time problem in Eq. (2.3.8) to a discrete-time Markov chain with a transition kernel, with either countable state space or uncountable state space, in which case $p(x(t+1) \mid x(t), u(t))$ is either a probability mass function (PMF) or a probability density function (PDF), respectively. For the continuous-time case the model $p(x(t+h) \mid x(t), u(t))$ corresponds to a dynamic evolution either given by a CTMC or by an SDE for countable state spaces and uncountable state spaces, respectively. Throughout, we will assume a stationary dynamical system for the state. Note that, in the case of a finite horizon objective this assumption can be relaxed [27].

Traditionally, the case of countable state spaces is most prominently discussed in operations research, queueing theory, and machine learning [28–30]. Within this setting, the OC problem is known as a Markov decision process (MDP). An MDP is defined as a tuple $(\mathcal{X}, \mathcal{U}, \mathsf{P}, R)$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, P is the transition model described by a probability measure and R is a reward function. Vice versa, decision-making in uncountable state spaces has a long tradition in control theory [13, 31]. In this context, the problem is known as optimal control (OC) or stochastic optimal control (SOC), depending on the state's deterministic or stochastic evolution, respectively.

There exist two main lines of reasoning on how to solve OC problems. One is Pontryagin's maximum principle [31, 32], which we will not discuss in this thesis, and the other one is dynamic programming [27, 33]. We will use dynamic programming as it opposed to Pontryagin's maximum principle (i) yields the necessary *and* sufficient condition for optimality and (ii) it can be more easily be applied to problems with a stochastic state evolution.

2.3.1 *Dynamic Programming*

Next, we find equations for the so-called *value function*. Given the value function, it is easy to (i) find the optimal action trajectory $u^*_{[0,\infty]}$, which maximizes Eqs. (2.3.8) and (2.3.9), respectively, and (ii) find the optimal objective value $J^* \coloneqq J\left[u^*_{[0,\infty]}\right]$.

2.3.1.1 *Discrete-Time Bellman Equations*

First, we consider the case where the time t is in a countable space; i.e., $t \in \mathcal{T} \subseteq \mathbb{N}_{\geq 0}$. We aim to solve the OC problem

$$\begin{aligned} \underset{u_{[0,\infty]}}{\text{maximize}} \quad & J\left[u_{[0,\infty]}\right] = \mathsf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(x(t), u(t))\right] \\ \text{subject to} \quad & x(t+1) \sim p(x(t+1) \mid x(t), u(t)), \quad \forall t \in \{0, 1, \dots\}. \end{aligned}$$

We define the value function (or the negative cost to go) as

$$V(x) \coloneqq \max_{u_{[t,\infty]}} \mathsf{E}\left[\sum_{s=t}^{\infty} \gamma^{s-t} R(x(s), u(s)) \;\middle|\; x(t) = x\right]. \tag{2.3.10}$$

Next, we compute for the value function

$$\begin{aligned} V(x) &= \max_{u_{[t,\infty]}} \mathsf{E}\left[\sum_{s=t}^{\infty} \gamma^{s-t} R(x(s), u(s)) \;\middle|\; x(t) = x\right] \\ &= \max_{u_{[t,\infty]}} \mathsf{E}\left[R(x(t), u(t)) + \sum_{s=t+1}^{\infty} \gamma^{s-t} R(x(s), u(s)) \;\middle|\; x(t) = x\right] \\ &= \max_{u(t) \in \mathcal{U}} \left\{R(x, u(t)) + \gamma\, \mathsf{E}[V(x(t+1)) \mid x(t) = x]\right\}, \end{aligned} \tag{2.3.11}$$

where Eq. (2.3.11) is known as the *principle of optimality*. Solving this principle of optimality is known as *dynamic programming*.

DISCRETE STATE SPACES. Let $x(t+1) = x' \in \mathcal{X}$ and $x(t) = x \in \mathcal{X}$ be the states at time $t+1$ and t, respectively. Additionally, we use $u(t) = u \in \mathcal{U}$ as the action applied at time t. If we assume a discrete state space, e.g., $\mathcal{X} \subseteq \mathbb{N}$, we find the *Bellman equation*

$$V(x) = \max_{u \in \mathcal{U}} \left\{R(x, u) + \gamma \sum_{x' \in \mathcal{X}} p(x' \mid x, u) V(x')\right\}, \tag{2.3.12}$$

where $p(x' \mid x, u)$ is a PMF.

CONTINUOUS STATE SPACES. Similarly, for a continuous state space, e.g., $\mathcal{X} \subseteq \mathbb{R}$ and a PDF $p(x' \mid x, u)$, the Bellman equation reads

$$V(x) = \max_{u \in \mathcal{U}} \left\{ R(x,u) + \gamma \int p(x' \mid x, u) V(x') \, \mathrm{d}x' \right\}. \tag{2.3.13}$$

2.3.1.2 *Continuous-Time Bellman Equations*

Next, we discuss how, similar to Eqs. (2.3.12) and (2.3.13), we can find analog Bellman equations for the continuous-time case. We start with the OC problem

$$\begin{aligned} \underset{u_{[0,\infty]}}{\text{maximize}} \quad & J\left[u_{[0,\infty]}\right] = \mathsf{E}\left[\int_0^\infty e^{-\frac{t}{\tau}} R(x(t), u(t)) \, \mathrm{d}t \right] \\ \text{subject to} \quad & x(t+h) \sim p(x(t+h) \mid x(t), u(t)), \quad \forall t \in \mathbb{R}_{\geq 0}. \end{aligned}$$

Analog to Eq. (2.3.10) we define the continuous-time value function as

$$V(x) \coloneqq \max_{u_{[t,\infty]}} \mathsf{E}\left[\int_t^\infty e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right]. \tag{2.3.14}$$

The principle of optimality can then be found by computing

$$\begin{aligned} V(x) &= \max_{u_{[t,\infty]}} \mathsf{E}\left[\int_t^\infty e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right] \\ &= \max_{u_{[t,\infty]}} \mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s + \int_{t+h}^\infty e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right] \\ &= \max_{u_{[t,t+h]}} \left\{ \mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right] \right. \\ &\qquad \left. + e^{-\frac{h}{\tau}} \, \mathsf{E}[V(x(t+h)) \mid x(t) = x] \right\}. \end{aligned} \tag{2.3.15}$$

DISCRETE STATE SPACES. For the discrete state space case, e.g., $\mathcal{X} \subseteq \mathbb{N}$, we consider that the dynamic evolution is given by a CTMC. Therefore, we assume the infinitesimal definition

$$\mathsf{P}(X(t+h) = x' \mid X(t) = x, u(t) = u) = \mathbb{1}(x = x') + \Lambda(x, x' \mid u) h + o(h).$$

Let $x(t+h) = x'$ and $x(t) = x$, then by starting with the principle of optimality we find

$$\begin{aligned} V(x) &= \max_{u_{[t,t+h]}} \left\{ \mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right] \right. \\ &\qquad \left. + e^{-\frac{h}{\tau}} \, \mathsf{E}[V(x(t+h)) \mid x(t) = x] \right\} \\ &= \max_{u_{[t,t+h]}} \left\{ \mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(x(s), u(s)) \, \mathrm{d}s \;\middle|\; x(t) = x \right] \right. \\ &\qquad \left. + e^{-\frac{h}{\tau}} \sum_{x' \in \mathcal{X}} \left[\mathbb{1}(x = x') + \Lambda(x, x' \mid u) h + o(h) \right] V(x') \right\}. \end{aligned}$$

Next, we rearrange the terms and divide both sides by h, which yields

$$V(x)\left(\frac{1-e^{-\frac{h}{\tau}}}{h}\right) = \max_{u_{[t,t+h]}}\left\{\frac{1}{h}\mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(x(s),u(s))\,\mathrm{d}s \;\middle|\; x(t)=x\right] + e^{-\frac{h}{\tau}}\sum_{x'\in\mathcal{X}}\Lambda(x,x'\mid u)V(x') + \frac{o(h)}{h}\right\}.$$

Taking the limit $\lim_{h\to 0}$ and rearranging the equation, we find the Bellman equation for the continuous-time and discrete state space case as

$$V(x) = \max_{u\in\mathcal{U}}\left\{\tau R(x,u) + \tau\sum_{x'\in\mathcal{X}}\Lambda(x,x'\mid u)V(x')\right\}. \tag{2.3.16}$$

CONTINUOUS STATE SPACES. For the continuous state space case, we consider a dynamical system for the state $x(t)\equiv\mathbf{x}(t)\in\mathcal{X}\subseteq\mathbb{R}^n$ given by the SDE

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t),u(t))\,\mathrm{d}t + \mathbf{G}(\mathbf{x}(t),u(t))\,\mathrm{d}\mathbf{w}(t).$$

In order to compute the r.h.s. of Eq. (2.3.15), we compute $V(\mathbf{x}(t+h))$ using Itô calculus as

$$\begin{aligned} V(\mathbf{x}(t+h)) = V(\mathbf{x}(t)) &+ \int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial\mathbf{x}}\mathbf{f}(\mathbf{x}(s),u(s))\,\mathrm{d}s \\ &+ \int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial\mathbf{x}}\mathbf{G}(\mathbf{x}(s),u(s))\,\mathrm{d}\mathbf{w}(s) \\ &+ \int_t^{t+h}\frac{1}{2}\operatorname{tr}\left(\frac{\partial^2 V(\mathbf{x}(s))}{\partial\mathbf{x}^2}\mathbf{G}(\mathbf{x}(s),u(s))\mathbf{G}^\top(\mathbf{x}(s),u(s))\right)\mathrm{d}s. \end{aligned} \tag{2.3.17}$$

Using the principle of optimality in Eq. (2.3.15) yields with Eq. (2.3.17)

$$\begin{aligned} V(\mathbf{x}) &= \max_{u_{[t,t+h]}}\left\{\mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(\mathbf{x}(s),u(s))\,\mathrm{d}s \;\middle|\; \mathbf{x}(t)=\mathbf{x}\right] + e^{-\frac{h}{\tau}}\mathsf{E}[V(\mathbf{x}(t+h))\mid\mathbf{x}(t)=\mathbf{x}]\right\} \\ &= \max_{u_{[t,t+h]}}\left\{\mathsf{E}\left[\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(\mathbf{x}(s),u(s))\,\mathrm{d}s \;\middle|\; \mathbf{x}(t)=\mathbf{x}\right]\right. \\ &\quad + e^{-\frac{h}{\tau}}\mathsf{E}\left[V(\mathbf{x}(t)) + \int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial\mathbf{x}}\mathbf{f}(\mathbf{x}(s),u(s))\,\mathrm{d}s\right. \\ &\quad + \int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial\mathbf{x}}\mathbf{G}(\mathbf{x}(s),u(s))\,\mathrm{d}\mathbf{w}(s) \\ &\quad \left.\left. + \int_t^{t+h}\frac{1}{2}\operatorname{tr}\left(\frac{\partial^2 V(\mathbf{x}(s))}{\partial\mathbf{x}^2}\mathbf{G}(\mathbf{x}(s),u(s))\mathbf{G}^\top(\mathbf{x}(s),u(s))\right)\mathrm{d}s \;\middle|\; x(t)=x\right]\right\}. \end{aligned}$$

By rearranging and dividing both sides by h, we have

$$V(\mathbf{x})\left(\frac{1-e^{-\frac{h}{\tau}}}{h}\right) = \max_{u_{[t,t+h]}}\left\{\mathsf{E}\left[\frac{1}{h}\int_t^{t+h} e^{-\frac{s-t}{\tau}} R(\mathbf{x}(s),u(s))\,\mathrm{d}s \;\middle|\; \mathbf{x}(t)=\mathbf{x}\right]\right.$$
$$+\frac{e^{-\frac{h}{\tau}}}{h}\mathsf{E}\left[\int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}(s),u(s))\,\mathrm{d}s + \int_t^{t+h}\frac{\partial V(\mathbf{x}(s))}{\partial \mathbf{x}}\mathbf{G}(\mathbf{x}(s),u(s))\,\mathrm{d}\mathbf{w}(s)\right.$$
$$\left.\left.+\int_t^{t+h}\frac{1}{2}\operatorname{tr}\left(\frac{\partial^2 V(\mathbf{x}(s))}{\partial \mathbf{x}^2}\mathbf{G}(\mathbf{x}(s),u(s))\mathbf{G}^\top(\mathbf{x}(s),u(s))\right)\,\mathrm{d}s \;\middle|\; x(t)=x\right]\right\}.$$

Taking the limit $\lim_{h\to 0}$ and let $u(t) = u \in \mathcal{U}$, we have the partial differential equation (PDE)

$$V(\mathbf{x}) = \max_{u\in\mathcal{U}}\left\{\tau R(\mathbf{x},u) + \tau\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x},u) + \frac{\tau}{2}\operatorname{tr}\left(\frac{\partial^2 V(\mathbf{x})}{\partial \mathbf{x}^2}\mathbf{G}(\mathbf{x},u)\mathbf{G}^\top(\mathbf{x},u)\right)\right\}. \tag{2.3.18}$$

Equation (2.3.18) is known as the stochastic Hamilton-Jacobi-Bellman (HJB) equation, which has a long history within control theory; for further discussion, see, e.g., [13, 27, 31, 34].

2.3.2 *The Optimal Policy*

For the presented OC problems in Eqs. (2.3.8) and (2.3.9) we have assumed that the transition kernels of the Markov processes are independent of the time variable t. In this case the value functions (Eqs. (2.3.10) and (2.3.14)) are equal to the corresponding optimal objective function, since the Bellman equations hold for every start point t including $t = 0$. Therefore, the optimal objective is given as $J^* = V(x)$, when starting at time $t = 0$ with state $x(0) = x$. The optimal action signal $u^*(t)$ at time t for a state $x(t) = x$ is therefore given as the maximizer of the r.h.s. of the Bellman equation, i.e., the maximizer of Eqs. (2.3.12), (2.3.13), (2.3.16) and (2.3.18). The optimal action $u^*(t)$ can also be written using a *feedback policy* $\mu : \mathcal{X} \to \mathcal{U}$, as $u^*(t) = \mu(x(t))$.

DISCRETE-TIME OPTIMAL POLICY For the discrete-time case, this policy is therefore

$$\mu(x) = \arg\max_{u\in\mathcal{U}}\left\{R(x,u) + \gamma\sum_{x'\in\mathcal{X}} p(x' \mid x,u)V(x')\right\} \quad \forall x \in \mathcal{X}$$

for the discrete state space case, and for the continuous state space case, we have

$$\mu(x) = \arg\max_{u\in\mathcal{U}}\left\{R(x,u) + \gamma\int p(x' \mid x,u)V(x')\,\mathrm{d}x'\right\} \quad \forall x \in \mathcal{X}.$$

CONTINUOUS-TIME OPTIMAL POLICY In continuous time we have for the discrete state space case

$$\mu(x) = \arg\max_{u\in\mathcal{U}}\left\{\tau R(x,u) + \tau\sum_{x'\in\mathcal{X}} \Lambda(x,x' \mid u)V(x')\right\} \quad \forall x \in \mathcal{X}$$

and for the continuous state space case, we have

$$\mu(\mathbf{x}) = \arg\max_{u \in \mathcal{U}} \left\{ \tau R(\mathbf{x}, u) + \tau \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, u) + \frac{\tau}{2} \operatorname{tr} \left(\frac{\partial^2 V(\mathbf{x})}{\partial \mathbf{x}^2} \mathbf{G}(\mathbf{x}, u) \mathbf{G}^\top(\mathbf{x}, u) \right) \right\},$$

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

2.3.3 *Numerical methods*

Often, closed-form solutions for the Bellman equations are rare and hard to find. In the case where all spaces are discrete, we can easily apply methods such as *value iteration*. For value iteration, we parameterize the value function for every $x \in \mathcal{X}$ as a table. Next, we iterate the value function by applying a one-step Bellman update. More precisely, consider the discrete-time discrete state space case from Eq. (2.3.12), this gives for the value function $V_{(l)}(x)$ at iteration step l the update formula

$$V_{(l)}(x) = \max_{u \in \mathcal{U}} \left\{ R(x, u) + \gamma \sum_{x' \in \mathcal{X}} p(x' \mid x, u) V_{(l-1)}(x') \right\},$$

where the update has to be performed for every $x \in \mathcal{X}$, with an arbitrary initialization $V_{(0)}(x)$, $\forall x \in \mathcal{X}$. It can be shown, that under certain technical conditions this algorithm converges to the solution of the Bellman equation [27]. Alternatively, one can use the state-action value function, or Q-function $Q(x, u)$, which can be defined as the r.h.s. of the Bellman equation, without the $\max$-operator; i.e., in the discrete-time discrete state space case

$$Q(x, u) \coloneqq R(x, u) + \gamma \sum_{x' \in \mathcal{X}} p(x' \mid x, u) V(x') \quad \forall (x, u) \in \mathcal{X} \times \mathcal{U}.$$

This definition implies by the Bellman equation that $V(x) = \max_{u \in \mathcal{U}} Q(x, u)$. The state-action value function can then be easily be used to obtain the optimal policy, as

$$\mu(x) = \arg\max_{u \in \mathcal{U}} Q(x, u) \quad \forall x \in \mathcal{X}.$$

For the value iteration scheme this yields for the state-action value function the algorithm

$$Q_{(l)}(x, u) = R(x, u) + \gamma \sum_{x' \in \mathcal{X}} p(x' \mid x, u) \max_{u' \in \mathcal{U}} Q_{(l-1)}(x', u'),$$

where the update has to be performed for every state and action; i.e., $\forall (x, u) \in \mathcal{X} \times \mathcal{U}$. For further discussion on value iteration and other numerical schemes see, e.g., [27, 30].

For the continuous-state space case, OC problems tend to be even more challenging than the discrete-state space case, as we can not anymore iterate over all possible states, as the state space is continuous. However, a classic method is to map a problem to a closed-form solution. A celebrated closed-form solution of the Bellman equation (HJB equation) is given for problems with linear state dynamics and quadratic cost functions. This is known as linear quadratic (LQ) control for deterministic problems and linear quadratic Gaussian (LQG) control for stochastic problems, for more see, e.g., [13, 27, 31].

Another alternative method for large state spaces, which we will exploit in this thesis, is to parameterize the unknown value function and use learning to find the optimal parameters, which fulfill the Bellman equation. In cases where the learning uses data points obtained by simulating the system, this is known as reinforcement learning (RL) [30], which we discuss next.

2.3.4 *Reinforcement Learning*

In reinforcement learning (RL) we solve the Bellman equations approximately, see, e.g., [30, 35]. Usually, the motivation is that we do not know the model of the system at hand; therefore, we use samples obtained by simulating or running the system to learn an approximately optimal policy. However, we can also use RL methods to solve *planning* problems, where the model is given. RL methods are appealing because they use stochastic approximation to solve the Bellman equation. This avoids the so-called *curse of dimensionality* [33], which makes the Bellman equation hard to solve.

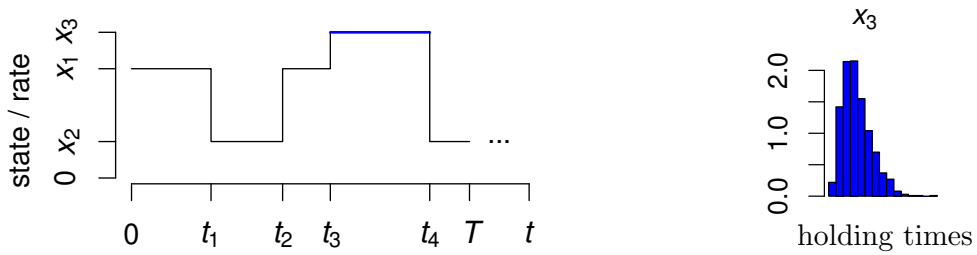
RL methods can be subdivided roughly into two subcategories: (i) model-free approaches and (ii) model-based approaches. Model-free approaches solve the Bellman equations without estimating a model, which is in the case of an MDP the transition kernel of the Markov process. On the other hand, model-based approaches typically first try to estimate the model and then solve the Bellman equation with the estimated model. Model-free approaches are usually less computationally expensive compared to model-based approaches and do not need a model. However, we will largely follow a model-based approach in this thesis, as model-based RL is favorable in sample efficiency, and we will show how to obtain scalable models for various settings.

3

BAYESIAN NON-PARAMETRIC INFERENCE FOR A CONTINUOUS-TIME POINT PROCESS MODEL

This chapter discusses how we can obtain a continuous-time model for event-based time-series data common in many applications. This can include recording the time stamps of events in social networks or computer networks. It is, for example, customary to model requests to a server as a point process. Often these point processes are non-stationary, and therefore the time between two events has a time-dependent distribution. Similar models are, for example, Markov modulated Poisson processes, which are used, e.g., in queueing theory [36]. However, classical models often do need substantial compute power to infer the latent Markov chain. Another line of work focuses on modeling the point process using a different more tractable latent stochastic process modulating the rate of the point process. This can, for example, be a Gaussian process (GP) as, e.g., in [37]. However, often it is sensible in many application scenarios to choose this process as piecewise constant as the latent process models some switching behavior between different regimes. Given this description, inference can be improved substantially as more data can be used to infer statistics of the time-dependent waiting time distribution. Though, a challenge is how to specify the number of regimes and the base distribution, which models the waiting time. In the following chapter, we, therefore, present a method that solves these two challenges. For this, we evaluate the presented method for synthetic data, which has similar behavior to packages arriving in a network system, see e.g., [6, 8, 9]. Additionally, we present an application that concerns the case of neuron-firings in the brain of mice, which have similar non-stationary behavior. Here, we consider the switching of recurring *neuronal states*. These neuronal states in the brain are essential for adapting to a changing environment, such as switching between inattentive and vigilant states depending on the task demand [38].

We will focus on the inference of such systems, and a discussion on decision-making is left for Chapter 6, where we will elaborate more on the general case of making decisions under uncertainty for similar continuous-time piecewise-constant processes. This chapter extends and contains parts and material from the published work [4].



(a) Piecewise constant process modulating the rate.

(b) Histogram of the holding times in state x_3.

FIGURE 3.1: A latent piecewise-constant rate function modulating the holding times. (a) A step rate with four change points $t_1, \ldots, t_4$ up to time T. The rate takes three values (states, x_1, x_2, x_3). (b) An example histogram for the holding times within state x_3, (i.e., in $[t_3, t_4]$).

Background

In the statistical analysis of series of events, the non-stationarity of point processes is a persistent problem. For this, consider the data for the point process described by the set

$$\mathcal{D} \coloneqq \{y_n \mid n = 1, \ldots, N\},$$

with N data points $y_n \in \mathbb{R}_{\geq 0}$. These data points can be interpreted as the sojourn times of a counting process $N(t) \coloneqq \sum_{n=1}^{N} \mathbb{1}(t_n \leq t)$, where t_n denote the event times; i.e., $y_n = t_{n+1} - t_n$ with $t \in \mathbb{R}_{\geq 0}$. A standard assumption is to model this counting processes as a Poisson counting process $N(t) \sim \mathcal{PP}(N(t) \mid \lambda)$, with rate λ. However, this assumption can be limiting in two ways (i) it does not capture non-stationary behavior, as the rate λ is a constant and (ii) it implicitly has an exponential distributional assumption on the waiting times; i.e., $y_n \sim \mathrm{Exp}(y_n \mid \lambda)$. This is emphasized in scenarios such as the one depicted in Fig. 3.1. The non-stationary behavior is also very challenging, as typical analysis methods assume constant parameters and may lead to erroneous conclusions when applied to non-stationary data [39, 40]. Therefore, one tries to identify changes in the firing rate (change points) to split up the *spike trains* into sections of approximately constant rate and apply the analyses in the individual sections separately. As this may reduce the sample size, it would be helpful if sections of similar rate could be joined in the analysis, see Fig. 3.1. Also, change points may indicate exciting aspects of the internal state. Secondly, assuming exponentially distributed waiting times can lead to wrong statistical conclusions. Therefore, we follow a different route and assume that the data points are gamma distributed, i.e., $y_n \sim \mathrm{Gam}(y_n \mid a_n, b_n)$, and have the shape parameter a_n and rate parameter b_n reoccurring.

To this end, we, therefore, present a model which deals with the two discussed limitations. We build upon the work of [41] and present a modeling framework for point processes with gamma-distributed increments, where a piecewise constant latent process controls the shape and scale of the distribution. By exploiting Bayesian non-parametrics, we utilize a Chinese restaurant process (CRP) [42] as a prior for the discrete number of states of the latent process. The inference is carried out using a tractable MCMC approach. This leads to a new model and inference method, which can be used in, for example, empirical spike

train recordings, which inherently possess non-stationary and non-Poissonian behavior. The accompanying code is publicly available via Git.[1]

Related Work

In the related discipline of change point detection, many methods have been proposed that are often based on moving or cumulative sums, Bayesian methods, or information criteria, for a review see, e.g., [43]. Recently, a method that has been designed specifically for the temporal properties and peculiarities of neuronal spiking patterns has been proposed, called multiple filter test (MFT) [44, 45]. Often, these methods allow the rate to follow an arbitrary step function. However, this assumption does not account for the mechanism of recurring states of firing rates between which, e.g., neurons in the brain may switch over time.

Methods assuming Markovian dynamics have been investigated for a fixed number of states. Many such methods inherently assume exponential distributed holding times. However, in many application scenarios, such as network modeling or in the case of neurons, this assumption does not hold. Consider, for example, the case of neuron firing; here, the neurons possess a stochastic refractory period, which prohibits them from firing in arbitrary short intervals. Therefore, some methods use more general assumptions on the distribution of holding times, such as the gamma distribution [46, 47]. These methods often require that the number of discrete states is defined before the analysis. However, this knowledge is not available in many scenarios. Therefore, we generalize the work in [41], which has a non-parametric assumption on the number of states using a CRP with exponentially distributed holding times, to gamma-distributed holding times, allowing the analysis of applications such as neuronal firing patterns.

3.1 A GENERATIVE MODEL FOR POINT PROCESSES

THE LIKELIHOOD. In this chapter, we assume the data $\mathcal{D}$ to be a realization of an inhomogeneous gamma process, with time-dependent shape parameter $a(t) \in \mathbb{R}_{\geq 0}$ and rate parameter $b(t) \in \mathbb{R}_{\geq 0}$. Additionally, we require that the latent process $\mathbf{x}(t) \coloneqq [a(t), b(t)]^\top$ is a piecewise constant process in $\mathbb{R}^2$. Hence, we have equal jump times in both $a(t)$ and $b(t)$. We define its path-wise realization as $\mathbf{x}_{[0,T]} \coloneqq \{\mathbf{x}(t) \mid t \in [0, T]\}$. As the latent process $\mathbf{x}(t)$ in the interval $[0, T]$ is piecewise constant, we can assume without loss of generality that it jumps c-times in the interval. Therefore, it can be fully described by its $c + 1$ holding times and the corresponding values during the intervals. Hence, the parameters of the gamma process can be written as a matrix $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m]^\top$, with m distinct states, $m \leq c + 1$ and the parameter $\boldsymbol{\theta}_i = [a_i, b_i]^\top$ of a state $i = 1, \ldots, m$ contains

1 https://github.com/bastianalt/non_param_bayes_change_points

the corresponding shape parameter a_i and rate parameter b_i. This yields the likelihood of the data $\mathcal{D}$ for a given path $\mathbf{x}_{[0,T]}$ as

$$p(\mathcal{D} \mid \mathbf{x}_{[0,T]}) = \prod_{i=1}^{m} \prod_{n=1}^{N} \left[\mathrm{Gam}(y_n \mid a_i, b_i)\right]^{\mathbb{1}(z_n=i)}. \tag{3.1.1}$$

Here, we use the assignment variables $\mathbf{z} \coloneqq [z_1, \ldots, z_N]^\top$, where the variable $z_n \in \{1, \ldots, m\}$ assigns the nth data samples y_n to its corresponding state z_n. Note that for a given path $\mathbf{x}_{[0,T]}$ the assignment variables $\mathbf{z}$ are entirely determined. For more information, see [22] where similar formulations are used, e.g., for Gaussian mixture models. The corresponding log-likelihood to Eq. (3.1.1) is therefore

$$\log p(\mathcal{D} \mid \mathbf{x}_{[0,T]}) = \sum_{i=1}^{m} n_i(-\log \Gamma(a_i) + a_i \log b_i) + (a_i - 1)s_i - b_i \tilde{s}_i,$$

where $\Gamma(\cdot)$ is the gamma function. Here, we define the number of times state $i \in \{1, \ldots, m\}$ is chosen for all data points as

$$n_i \coloneqq \sum_{n=1}^{N} \mathbb{1}(z_n = i),$$

and the sufficient statistics [23] of the corresponding gamma distributions are given as

$$s_i \coloneqq \sum_{n=1}^{N} \mathbb{1}(z_n = i) \log y_n$$
$$\tilde{s}_i \coloneqq \sum_{n=1}^{N} \mathbb{1}(z_n = i) y_n.$$

THE PRIOR DISTRIBUTION. Next, we describe the prior distribution of the latent process $\mathbf{x}(t)$. Here, we use a prior as presented in [41] and shortly recap the proposed approach.

For the jump times of the latent process $\mathbf{x}(t)$ we leverage a Poisson process, such that the $c+1$ holding times $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_{c+1}]^\top$ between two jumps of the piecewise constant process are exponentially distributed with rate f; i.e.,

$$p(\boldsymbol{\tau} \mid f) = \prod_{j=1}^{c+1} \mathrm{Exp}(\tau_j \mid f). \tag{3.1.2}$$

This yields the explicit expression $p(\boldsymbol{\tau} \mid f) = f^{c+1} e^{-fT}$, as the waiting times sum up to the length of the interval; i.e., $\sum_{j=1}^{c+1} \tau_j = T$.

For the assignment of the $c+1$ segments of the latent process $\mathbf{x}(t)$ to the parameters, we use as a prior distribution a Chinese restaurant process (CRP) [42]. Here, the assignment parameters $k_1, k_2, \ldots$ follow the CRP with concentration parameter α; i.e.,

$$p(k_j \mid \alpha, k_{j-1}, \ldots, k_1) = \mathrm{CRP}(k_j \mid \alpha, k_{j-1}, \ldots, k_1).$$

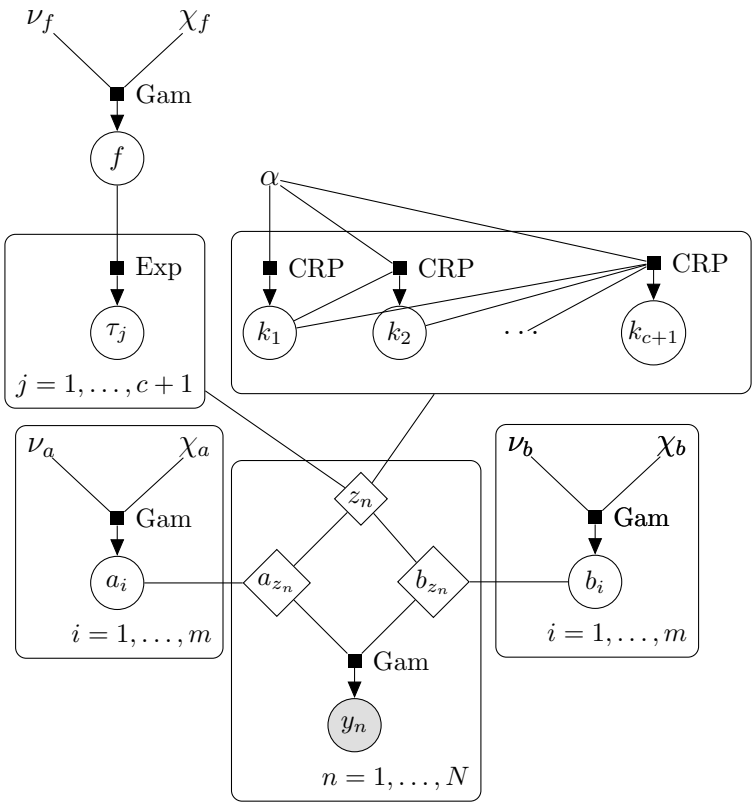


FIGURE 3.2: PGM for the generative point process model presented in Section 3.1.

The joint distribution for $\mathbf{k} = [k_1, \ldots, k_{c+1}]^\top \in \{1, \ldots, m\}^{c+1}$ can hence be derived as

$$p(\mathbf{k} \mid \alpha) = \alpha^m \frac{\prod_{i=1}^m (\Psi_i - 1)!}{\prod_{j=1}^{c+1} (\alpha + j - 1)},$$

where $\Psi_i = \sum_{j=1}^{c+1} \mathbb{1}(k_j = i)$ is the number of times state i has been chosen for all segments.

Therefore, we specify the prior distribution of a path $\mathbf{x}_{[0,T]}$ given the jump rate f, the concentration parameter α and a base distribution $p(\boldsymbol{\theta}_i)$ for the prior of the parameters as

$$p(\mathbf{x}_{[0,T]} \mid f, \alpha) = f^{c+1} e^{-fT} \alpha^m \frac{\prod_{i=1}^m p(\boldsymbol{\theta}_i)(\Psi_i - 1)!}{\prod_{j=1}^{c+1} (\alpha + j - 1)}.$$

Note that similar to the variable z_n, which assigns the data y_n to the corresponding parameter, the variables k_j assigns the segments of the latent process to its parameter. Hence, the variables $\mathbf{k}$ are entirely determined given a realization of a path $\mathbf{x}_{[0,T]}$.

A PGM can be found in Fig. 3.2, where we instantiated the prior distribution over the ith parameter $\boldsymbol{\theta}_i$ as $p(\boldsymbol{\theta}_i) = \mathrm{Gam}(a_i \mid \nu_a, \chi_a)\,\mathrm{Gam}(b_i \mid \nu_b, \chi_b)$ and the jump rate has a prior $p(f) = \mathrm{Gam}(f \mid \nu_f, \chi_f)$.

3.2 POSTERIOR INFERENCE WITH MARKOV CHAIN MONTE CARLO

Having specified the generative model in Section 3.1, we are now able to devise a posterior inference scheme for a given data set $\mathcal{D}$. As we deal with a gamma distribution for the likelihood, we extend the work of [41] and present a modified version of their MCMC sampler. For an overview on MCMC, see [25]. In order to sample from the posterior of a path $p(\mathbf{x}_{[0,T]} \mid \mathcal{D})$ given the data $\mathcal{D}$ we use a Metropolis-within-Gibbs sampler. The Metropolis-Hastings sampler generates a new sample path $\mathbf{x}_{[0,T]}$ conditioned on the parameters $\boldsymbol{\theta}$ and f. The Gibbs step then updates the parameters $\boldsymbol{\theta}$ and f given the path $\mathbf{x}_{[0,T]}$. This is done by specifying an additional hierarchy to estimate the rate f using a prior distribution $p(f)$. Note that the approach presented here, compared to [41], faces several technical obstacles, which arise in the case of the inhomogeneous gamma processes model. Next, we present how these can be tackled.

3.2.1 *Sampler Initialization*

First, for the initialization of the sampler, we draw a path from the prior $\mathbf{x}^{(0)}_{[0,T]} \sim p(\mathbf{x}_{[0,T]})$. This is done by sampling a rate f from the prior

$$f \sim p(f).$$

Conditioned on f we create $c+1$ holding times in the interval $[0, T]$ by sampling

$$\tau_j \sim \mathrm{Exp}(\tau_j \mid f), \quad j = 1, \ldots, c,$$

such that $\sum_{j=1}^{c} \tau_j \leq T$. We set the last holding time to $\tau_{c+1} = T - \sum_{j=1}^{c} \tau_j$.
Next, for all segments, we sample an assignment parameter from the CRP

$$k_j \sim \mathrm{CRP}(k_j \mid \alpha, k_{j-1}, \ldots, k_1), \quad j = 1, \ldots, c+1.$$

Here, whenever we introduce a new state i, we sample a new parameter set from the prior

$$\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}_i).$$

Hence, this creates the full description of the initial path $\mathbf{x}^{(0)}_{[0,T]}$. The corresponding pseudo-code is provided in Fig. 3.3.

3.2.2 *Updating the Path: Metropolis-Hastings Step*

In this step of the sampler, we sample a new path $\mathbf{x}^{(l+1)}_{[0,T]}$ by using a Metropolis-Hastings random walk. We propose a new path $\mathbf{x}^{*}_{[0,T]}$ by modifying the old path $\mathbf{x}^{(l)}_{[0,T]}$.[2] This modification of $\mathbf{x}^{(l)}_{[0,T]}$ is done by either: (i) adding a jump, (ii) shifting a jump time,

2 All parameters in relation to the proposal are denoted using *.

input : T: length of time interval
$f \sim p(f)$: rate prior sample
α: concentration parameter
$p(\boldsymbol{\theta}_i) = p(a_i, b_i)$: parameter prior
output : $\mathbf{x}_{[0,T]} \sim p(\mathbf{x}_{[0,T]})$: prior sample

`// Sample holding times`
Initialize time $t = 0$, initialize counter $j = 1$
while $t \leq T$ **do**
  Sample holding time $\tau_j \sim \mathrm{Exp}(\tau_j \mid f)$
  Update time $t \leftarrow t + \tau_j$, update counter $j \leftarrow j + 1$
end
Set number of intervals $c = j - 1$.
Set last holding time $\tau_{c+1} = T - \sum_{j=1}^{c} \tau_j$.

`// Sample assignments and parameters`
Initialize counter $j = 1$, initialize number of states $m = 0$
while $j \leq c + 1$ **do**
  Sample assignment $k_j \sim \mathrm{CRP}(k_j \mid \alpha, k_{j-1}, \ldots, k_1)$
  if k_j *is new* **then**
    Sample parameter $\boldsymbol{\theta}_{s+1} \sim p(\boldsymbol{\theta}_{s+1})$
    Update number of states $m = m + 1$
  end
  $j = j + 1$
end
return *Construct path* $\mathbf{x}_{[0,T]}$ *using* { $\boldsymbol{\tau}$*: holding times,* $\boldsymbol{\theta}$*: parameters,* $\mathbf{k}$*: assignments* }

FIGURE 3.3: Pseudo-Code for sampling from the Prior.

(iii) removing a jump, (iv) switching the assignment of a segment to another state, (v) joining two states or (vi) dividing a state into two states. This type of MCMC scheme is known as reversible-jump MCMC; for more information, see [41, 48].

The modification of an old path $\mathbf{x}^{(l)}_{[0,T]}$ to the new path proposal $\mathbf{x}^{*}_{[0,T]}$ is done by first choosing one of the proposal actions at random.

ADDING A JUMP. When choosing the add option, we add a jump at time t^* to the old path. We choose the jump time uniformly at random, $t^* \sim \mathrm{Uniform}(t^* \mid 0, T)$. We choose to either create a new parameter $\boldsymbol{\theta}^*$ with probability q_n or reuse an old parameter with probability $1 - q_n$.

SHIFTING A JUMP TIME. A jump is shifted by drawing the shifted jump time from a truncated Gaussian distribution, such that the shifted jump time is still between the two neighboring jumps.

REMOVING A JUMP. When the remove action is chosen, we draw one jump uniformly at random and remove it.

SWITCHING THE ASSIGNMENT. We switch the assignment of a segment j from an old state k_j to a new state k_j^*, by either creating a new parameter $\boldsymbol{\theta}^*$ with probability q_n or choosing an old one with probability $q_n - 1$.

JOINING TWO STATES. We join two states by selecting two states at random and then assigning a new parameter to this new state.

DIVIDING A STATE. For dividing a state into two states, we randomly select a state from the set of all states, which have at least two segments assigned to them. Then we draw two new states and assign the segments to one of the two new states.

3.2.2.1 *Proposal Parameter Sampler*

Whenever a new parameter is created by either adding a jump, switching a segment assignment, or in the process of joining or dividing states, we draw a new parameter for the proposal path $\mathbf{x}^*_{[0,T]}$. The Metropolis-Hastings algorithm, in general, achieves sampling from the posterior distribution by performing any completely random walk. Nevertheless, if the new parameter is drawn conditioned on the data, the mixing of the Markov Chain is expected to be significantly better in practice. For this reason, it is desirable to draw the new proposal parameter for the state $m+1$ from the posterior $p(\boldsymbol{\theta}^*_{m+1} \mid \mathcal{D}, \mathbf{x}^*_{[0,T]})$ given the proposal path and the data. Hence, the likelihood for the data given the new proposal parameter is

$$p(\mathcal{D} \mid \mathbf{x}^*_{[0,T]}, \boldsymbol{\theta}^*_{m+1}) = \prod_{n=1}^{N} [\mathrm{Gam}(y_n \mid a^*_{m+1}, b^*_{m+1})]^{\mathbb{1}(z^*_n = m+1)}.$$

Note that for this likelihood, a closed-form conjugate prior cannot be found, see [49]. This is problematic as we have to calculate the acceptance probability of the proposal path. For this, it is needed to have an expression for density of the posterior at the new parameter value $\boldsymbol{\theta}^*_{m+1}$. For this reason we add a new value $\boldsymbol{\theta}^*_{m+1}$ by fixing the corresponding shape parameter $a^*_{m+1} = 1$; i.e., $\boldsymbol{\theta}^*_{m+1} = [1, b^*_{m+1}]^\top$. This lets the likelihood collapse to a product of exponential distributions as

$$\begin{aligned} p(\mathcal{D} \mid \mathbf{x}^*_{[0,T]}, \boldsymbol{\theta}^*_{m+1}) &= \prod_{n=1}^{N} [\mathrm{Gam}(y_n \mid 1, b^*_{m+1})]^{\mathbb{1}(z^*_n = m+1)} \\ &= \prod_{n=1}^{N} [\mathrm{Exp}(y_n \mid b^*_{m+1})]^{\mathbb{1}(z^*_n = m+1)}. \end{aligned}$$

Now, a conjugate prior can be found to this likelihood and hence provides a closed-form posterior for the rate parameter b^*_{m+1}. Hence, we use a conjugate gamma prior as

$p(b^*_{m+1}) = \mathrm{Gam}(b^*_{m+1} \mid \nu_b, \chi_b)$ with hyper-parameters ν_b and χ_b. The parameter b^*_{m+1} can therefore be sampled from the resulting posterior distribution

$$p(b^*_{m+1} \mid \mathcal{D}, \mathbf{x}^*_{[0,T]}) \propto p(\mathcal{D} \mid \mathbf{x}^*_{[0,T]}, \boldsymbol{\theta}^*_{m+1}) p(b^*_{m+1}) \propto \mathrm{Gam}(b^*_{m+1} \mid \nu_b + n^*_{m+1}, \chi_b + \tilde{s}^*_{m+1}). \tag{3.2.3}$$

Here, the sufficient statistics are calculated using the assignments z^*_n to the new state $m+1$ of the proposal path as

$$n^*_{m+1} \coloneqq \sum_{n=1}^{N} \mathbb{1}(z^*_n = m+1)$$
$$\tilde{s}^*_{m+1} \coloneqq \sum_{n=1}^{N} \mathbb{1}(z^*_n = m+1) y_n.$$

This new proposal parameter $\boldsymbol{\theta}^*_{m+1} = [a^*_{m+1}, b^*_{m+1}]^T$ has always a shape parameter of $a^*_{m+1} = 1$. However, a^*_{m+1} is only needed for the proposal of a new path, and the parameters are corrected in the Gibbs-step of the sampler.

Another benefit of the proposed closed-form expression is that if we choose to reuse an existing value, for example, in the process of adding a jump, we can sample an old parameter $\boldsymbol{\theta}^*_i$ with probability proportional to the posterior in Eq. (3.2.3). Here, the posterior density $p(b^*_i \mid \mathcal{D}, \mathbf{x}^*_{[0,T]})$ is evaluated at the rate parameter b^*_i, which is calculated by moment matching the parameters of the gamma distribution to the parameters of the exponential distribution; i.e., $b^*_i = b_i / a_i$.

Finally, we can calculate the acceptance probability of the Metropolis-Hastings step as

$$p^{\mathrm{MH}}_x = \min(1, \Lambda_x \Phi_x \Omega_x), \tag{3.2.4}$$

with likelihood ratio $\Lambda_x = \frac{p(\mathcal{D} \mid \mathbf{x}^*_{[0,T]})}{p(\mathcal{D} \mid \mathbf{x}_{[0,T]})}$, prior ratio $\Phi_x = \frac{p(\mathbf{x}^*_{[0,T]} \mid f, \alpha)}{p(\mathbf{x}_{[0,T]} \mid f, \alpha)}$ and proposal ratio $\Omega_x = \frac{q(\mathbf{x}_{[0,T]} \mid \mathbf{x}^*_{[0,T]})}{q(\mathbf{x}^*_{[0,T]} \mid \mathbf{x}_{[0,T]})}$, which dependents on the proposal action that has been chosen.

3.2.3 *Updating the Parameters: Gibbs Step*

In the Gibbs step, we sample the jump rate f and the parameters $\boldsymbol{\theta}$ from the corresponding distribution conditioned on the path $\mathbf{x}_{[0,T]}$.

JUMP RATE. For the full conditional of the jump rate parameter f we use a conjugate prior $p(f) = \mathrm{Gam}(f \mid \nu_f, \chi_f)$ to the likelihood Eq. (3.1.2), with hyper-parameters ν_f and χ_f. This yields the full conditional

$$p(f \mid \boldsymbol{\tau}) \propto p(\boldsymbol{\tau} \mid f) p(f) \propto \mathrm{Gam}(f \mid \nu_f + c + 1, \chi_f + T).$$

PARAMETERS. The full conditional of the parameters $\boldsymbol{\theta}$ given a path can be calculated by noting that it factorizes as

$$p(\boldsymbol{\theta} \mid \mathcal{D}, \mathbf{x}_{[0,T]}) = \prod_{i=1}^{m} p(\boldsymbol{\theta}_i \mid \mathcal{D}, \mathbf{x}_{[0,T]}).$$

Hence, we can use for the likelihood term of the ith state, i.e.,

$$p(\mathcal{D} \mid \mathbf{x}_{[0,T]}, \boldsymbol{\theta}_i) = \prod_{n=1}^{N} \left[\mathrm{Gam}(y_n \mid a_i, b_i)\right]^{\mathbb{1}(z_n=i)},$$

a gamma prior for a_i and b_i; i.e.,

$$p(\boldsymbol{\theta}_i) = \mathrm{Gam}(a_i \mid \nu_a, \chi_a)\,\mathrm{Gam}(b_i \mid \nu_b, \chi_b),$$

with the hyper-parameters ν_a, χ_a, ν_b and χ_b. Therefore, we compute the full conditional of each parameter $\boldsymbol{\theta}_i$ up to a normalizing constant using

$$p(\boldsymbol{\theta}_i \mid \mathcal{D}, \mathbf{x}_{[0,T]}) \propto p(\mathcal{D} \mid \mathbf{x}_{[0,T]}, \boldsymbol{\theta}_i) p(\boldsymbol{\theta}_i). \tag{3.2.5}$$

Since the normalizing constant of Eq. (3.2.5) cannot be computed analytically, as discussed in [49], we use an additional Metropolis sampler for Eq. (3.2.5).

The Metropolis sampler is initialized by choosing a starting point close to the mode of Eq. (3.2.5) with $\boldsymbol{\theta}^{(0)} = [a^{(0)}, b^{(0)}]^T$, where we choose $a^{(0)} = 1$ and $b^{(0)} = \arg\max_b p(b \mid \mathcal{D}, \mathbf{x}_{[0,T]}, a^{(0)})$. In practice, this ensures a faster convergence of the Markov chain. We perform the random walk in the Metropolis algorithm by drawing $r_a, r_b \sim \mathcal{N}(r \mid 0, \sigma_\theta^2)$ and then setting $a^* = \exp(r_a + \log a)$ and $b^* = \exp(r_b + \log b)$, where σ_θ^2 is a parameter of the sampler. This yields the proposal distribution for $\theta^* = [a^*, b^*]^T$ as

$$q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = \mathrm{Lognorm}(a^* \mid \log a, \sigma_\theta^2)\,\mathrm{Lognorm}(b^* \mid \log b, \sigma_\theta^2).$$

Hence, the acceptance probability of a proposal is

$$p_\theta^{\mathrm{MH}} = \min\left(1, \Lambda_\theta \Phi_\theta \Omega_\theta\right),$$

with $\Lambda_\theta = \frac{p(\mathcal{D}|\mathbf{x}_{[0,T]}, \boldsymbol{\theta}^*)}{p(\mathcal{D}|\mathbf{x}_{[0,T]}, \boldsymbol{\theta})}$, $\Phi_\theta = \frac{p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta})}$ and $\Omega_\theta = \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}$.

3.2.4 *Path Point Estimate*

Here, we note that it is possible to find a point estimate. We can achieve this by calculating the maximum a posteriori (MAP) estimate over all possible paths

$$\hat{\mathbf{x}}_{[0,T]} = \arg\max_{\mathbf{x}_{[0,T]}} p(\mathbf{x}_{[0,T]} \mid \mathcal{D}). \tag{3.2.6}$$

An approximate solution to the intractable optimization problem in Eq. (3.2.6) can be computed by simulated annealing. The reader is referred to, e.g., [25]. For simulated

annealing, we decrease the acceptance probability of the Metropolis-Hastings sampler for $\mathbf{x}_{[0,T]}$, by introducing an inverse cooling schedule β_l which increases over the number of iteration steps l of the sampler, with $\lim_{l\to\infty} \beta_l \to \infty$. The annealed acceptance probability is found by modifying Eq. (3.2.4) to

$$p_x^{\mathrm{MH}}(\beta_l) = \min(1, [\Lambda_x \Phi_x]^{\beta_l} \, \Omega_x).$$

Using this acceptance probability in the sampler yields a Markov Chain, where its stationary state approximates the solution of Eq. (3.2.6).

3.3 EVALUATION OF THE METHOD: SIMULATIONS AND REAL-WORLD EXPERIMENTS

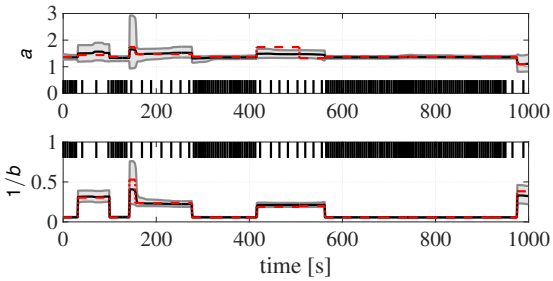
Next, we present the evaluation of the proposed inference method. The method is tested using the modeling assumption as well as a real-data experiment. For the experiment under the modeling assumption, we explicitly evaluate two simulation scenarios. For the real-data experiment, we consider a data set recorded from neuronal firing of mice. All hyper-parameters are set to yield vague prior distributions, and we use $\alpha = 3$ for the CRP as proposed in [41]. For annealing, we use an inverse temperature grid from 10^0 to 10^{10} on the logarithmic scale. We perform 10^5 iterations for burn-in and in total sample 10^6 paths. We conservatively discard 99% samples to ensure the remaining samples are independent. Consequently, we use 10^4 samples for the analysis of the posterior.

3.3.1 *Simulation Results*

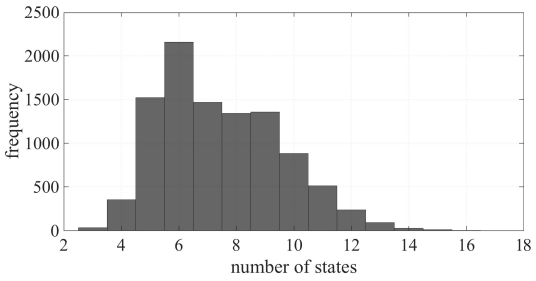
We draw two simulation data sets assuming the model presented in Section 3.1 by sampling two ground truth paths from the prior, see Section 3.2.1. Further, the observation data $\mathcal{D}$ is sampled using the ground truth paths. Note that this can be seen as a synthetic simulator for network systems, where the holding times are the times between customers in a queue, see, e.g., [6]. The inference algorithm then uses the simulated observation data.

Figures 3.4 and 3.5 show the results. In the first experiment, displayed in Fig. 3.4a, we can observe that the posterior distribution nicely infers the ground-truth data. Additionally, it gives an uncertainty estimate over the parameters. Since we use a non-parametric method, we can also plot the distribution over the number of latent states, which are represented in the different rate regimes of the observed data. In Fig. 3.4b we display the corresponding histogram for the posterior samples. We also use the same experiment to compare our gamma process assumption against the Poisson process assumption as presented in [41]. In Fig. 3.5a we can see that the Poisson approach tracks the ground truth, but the posterior mean is not as sharp as in the gamma process case, and the estimate has a higher posterior uncertainty.

As a second experiment we compare methods for point estimates as displayed in Fig. 3.5b. Here, we compare the point estimate using simulated annealing and the MFT method from [44]. Both methods perform similarly and reasonably well. However, our method



(a) Posterior inference for the shape path $a(t)$ (upper graph) and scale path $1/b(t)$ (lower graph).



(b) Posterior inference for the total number of states.

FIGURE 3.4: Posterior inference results under the model assumption. The ground truth trajectory (dashed red lines) in (a) is latent. The observations are given as a point process depicted as the small dashes in the plot. Here, the posterior inference scheme faithfully recovers the path displayed by the posterior distribution over shape and scale parameters. The black lines denote posterior mean, and shaded areas are the quantiles between $q_{0.05}$ and $q_{0.95}$. In (b) a corresponding histogram for the posterior samples is displayed, which depicts the posterior distribution over the number of states.

focuses explicitly on the estimation of recurring discrete states, which is helpful in cases where reoccurring states have an inherent meaning, such as physiologically relevant neuronal states or regimes of burstiness as, e.g., presented in [6].

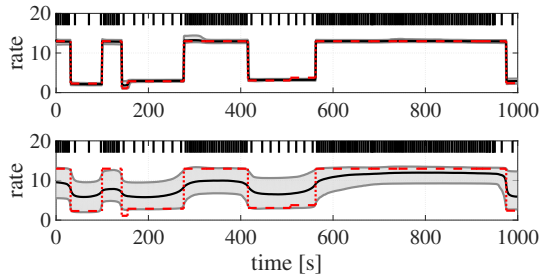
3.3.2 *Real Data Results*

The results for the experimental data are shown in Figs. 3.6 and 3.7.

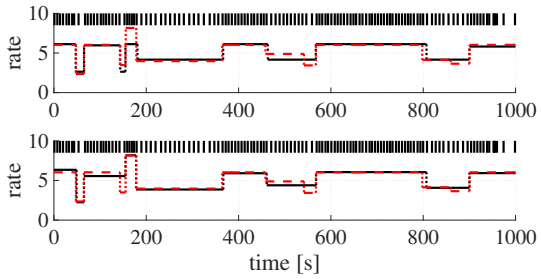
First, we notice in Fig. 3.6a that the shape parameter during periods with high rates is somewhat larger than one, which indicates that a gamma process assumption is more reasonable than the Poisson process assumption. Additionally, we see multiple possible explanations for the number of states, as depicted in Fig. 3.6b. However, the point estimate using annealing estimates just three different states of the process. Note that the MAP estimate is over the joint posterior distribution, which differs from the mode of the marginal posterior as depicted in the histogram in Fig. 3.6b. In Fig. 3.7b, we see a comparison between the MAP estimate and the posterior distribution over the time-dependent rate. Here, we observe that, as it is intuitive, the mean rate has higher uncertainty during rate changes than during periods with a rather constant rate. Additionally, the approximate MAP estimate is close to the posterior mean.

3.4 SUMMARY

In this chapter, we developed a model for continuous-time point processes with a non-stationary rate. For this, we used an inhomogeneous gamma process model with a CRP on the state assignment, which copes with the unknown number of states of the inhomogeneous gamma process. For inference, we used an MCMC scheme by building on the work of [41]. Here, we gave an extension using a Metropolis-within-Gibbs sampler to deal with the case



(a) Comparison of posterior inference using an inhomogeneous gamma process (upper graph) and using an inhomogeneous Poisson process [41] (lower graph).

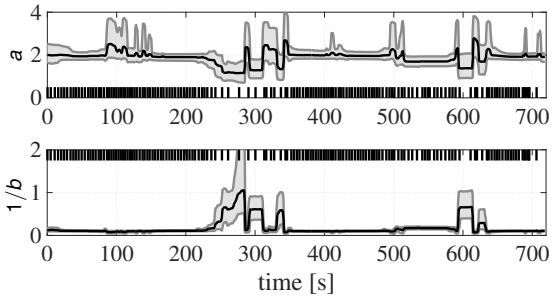


(b) Comparison of point estimates using simulated annealing (upper graph) and the MFT method [44] (lower graph).

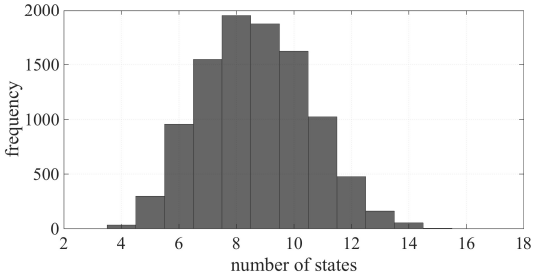
FIGURE 3.5: Comparison of the proposed inference scheme. The observations are given as a point process depicted as the small dashes in the plots. Black lines denote posterior mean, and shaded areas are the quantiles between $q_{0.05}$ and $q_{0.95}$. In (a) we compare two Bayesian inference schemes. The upper graph displays the presented scheme with a time-dependent posterior rate corresponding to the gamma distribution assumption; i.e., $b(t)/a(t)$. The lower Graph uses a reference method under a Poisson assumption with a time-dependent rate. The resulting estimation method tracks the ground truth sharper in the gamma case compared to the Poisson case. In (b) we compare the point estimation method using simulated annealing in the upper graph against the MFT method in the lower graph. The resulting estimation is similar, though the MFT method fails to recover the task of estimating reoccurring latent states, which generate the data.

of gamma-distributed holding times. For sampling a new parameter in the proposal of the Metropolis-Hastings random walk, we compensate the lack of a closed-form posterior for gamma likelihoods by fixing the shape parameter; this is then corrected in the Gibbs step. An additional Metropolis sampler, therefore, draws parameters of the posterior. We present a method to find an approximate solution for the MAP estimate of a path using annealing, which delivers an easily interpretable point estimate.

This Bayesian model is of use in areas such as network modeling as well as for neuronal spike trains, which often violate the Poisson assumption as shown in Section 3.3.2. In the case of neuronal spike trains, our results suggest that the analysis of discrete states can be particularly interesting for identifying hidden physiologically relevant neuronal states. For decision-making in these types of continuous-time discrete state space problems, a solution method is discussed in Chapter 6.

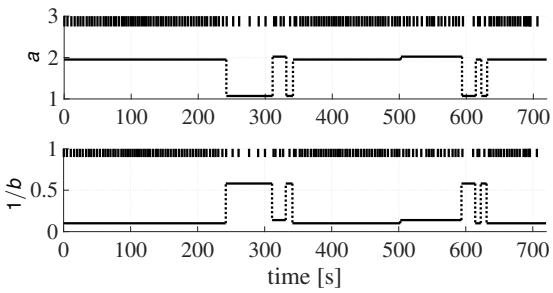


(a) Posterior inference for the shape path $a(t)$ (upper graph) and scale path $1/b(t)$ (lower graph).

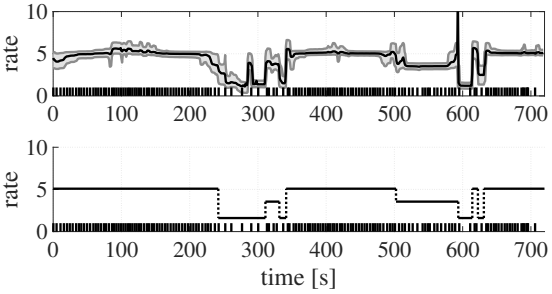


(b) Posterior inference for the total number of states.

FIGURE 3.6: Posterior inference results for real data of the neuronal activity in mice. The observations corresponding to firing neurons are given as a point process depicted in (a) as the small dashes in the plot. The posterior inference scheme estimates the means (black lines) of the shape path $a(t)$ and scale path $1/b(t)$ parameters of the corresponding gamma distribution, the shaded areas are the quantiles between $q_{0.05}$ and $q_{0.95}$. In (b) the corresponding histogram for the posterior samples over the number of latent states under the CRP prior is displayed. This information helps interpret regimes of brain activity, which switch over time.



(a) Point estimate using annealing for the shape path $a(t)$ (upper graph) and scale path $1/b(t)$ (lower graph).



(b) Posterior inference for the rate path $b(t)/a(t)$ (upper graph) and point estimate of the rate path using annealing (lower graph).

FIGURE 3.7: Point estimates and comparison for real data of the neuronal activity in mice. The observations denoted as black dashes in (a) are used in the simulated annealing procedure to estimate a point estimate for the shape path $a(t)$ and scale path $1/b(t)$. A comparison of the Bayesian estimate and the point estimate using simulated annealing is shown in (b). Here, we compare the posterior distribution for the rate $b(t)/a(t)$ with posterior mean (black lines) and Bayesian credible interval of quantiles between $q_{0.05}$ and $q_{0.95}$ (shaded areas) to the MAP point estimate for the rate using simulated annealing.

4

DECISION-MAKING WITH A SPARSE BAYESIAN MODEL IN VIDEO STREAMING

This chapter shows how Bayesian models, similar to the one developed in the previous chapter, can be used in decision-making. In particular, we present a Bayesian model designed for the use of adaptive bitrate (ABR) video-streaming. In ABR video-streaming, the streaming client has to choose among a set of bitrates to achieve a high-quality experience for the user watching a video. This can be seen as a prime example for making decisions under uncertainty. One significant difficulty is that decisions have to be typically made in order of milliseconds. This poses a problem for solving the underlying OC exactly. We, therefore, develop a method using a bandit strategy, which is a form of RL. These types of heuristics are often faster than comparably more sophisticated model-based RL algorithms (see Chapter 5), which solve the underlying OC problem with fewer data points, but typically require a lot more computing power and are therefore often hard to use in time-critical applications. This chapter extends and contains parts and material from the published works [3, 50].

From an application standpoint, quality adaptation algorithms for ABR streaming face an inherent problem when determining sustainable video quality. In this setting, they may either rely on the information gathered at the client vantage point or on the server and network assistance. The adaptation problem becomes particularly hard in future internet settings such as named data networking (NDN), where the notion of a network connection does not exist. The fundamental problem here is to determine how valuable either information is for the adaptation decision. Hence, we closely look in this chapter at the problem of using context information available to the client for video quality adaptation. Note that the problem description is agnostic to the underlying networking paradigm, making it an excellent fit for traditional internet protocol (IP)-based video streaming as well as NDN. In essence, we consider the fundamental problem of sequential decision-making

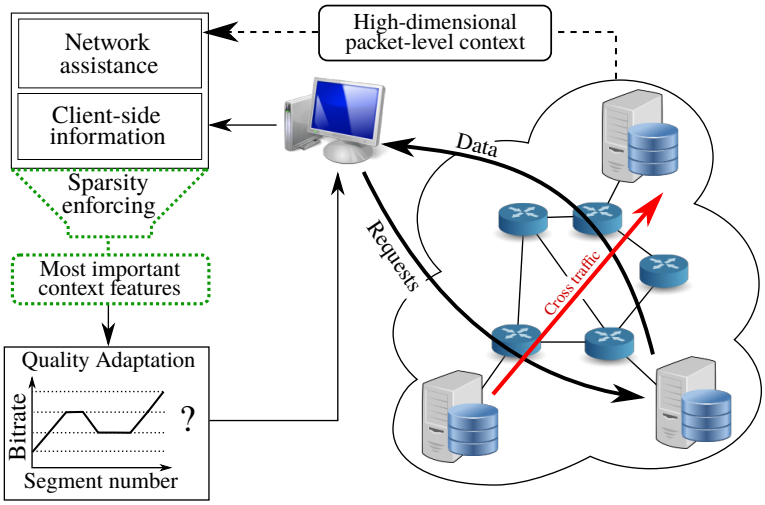


FIGURE 4.1: A standard client-based and/or network-assisted ABR streaming model (black) with the proposed contextual-based adaptation (CBA) (dotted). In CBA, high-dimensional context features from the network and client-side information undergo sparsity enforcement to shrink the impact of unimportant features.

under uncertainty where the client uses network context information received with every fetched video segment. In Fig. 4.1 we show a sketch where the client adaptation algorithm decides on the quality of the next segment based on a high-dimensional network context. We model the client's decision on a video segment quality as a contextual multi-armed bandit problem aiming to optimize an objective quality of experience (QoE) metric that comprises (i) the average video quality bitrate, (ii) the quality degradation, and (iii) the video stalling. For this, we use a *sparse* Bayesian model, which allows taking high-dimensional streaming context information, including client-measured variables and network assistance, for finding *online* the most valuable information for the quality adaptation. For the decision-making, we opt for the ABR quality adaptation for QoE maximization, which we formalize as a decision problem under uncertainty. Since the required approximate Bayesian inference can be computationally expensive, we develop a new fast inference scheme to support online video adaptation and contribute a sparse Bayesian contextual bandit algorithm denoted contextual-based adaptation (CBA). We perform an extensive evaluation of our adaptation algorithm in the particularly challenging setting of NDN, where we use an emulation testbed to demonstrate the efficacy of CBA compared to state-of-the-art algorithms. The accompanying code is publicly available via Git.[1]

Background

Video streaming services such as Netflix, YouTube, and Twitch, which constitute an overwhelming share of current internet traffic, use ABR streaming algorithms that try to find the most suitable video quality representation given the client's networking conditions. Current architectures use dynamic adaptive streaming over HTTP (DASH) in conjunction

1 https://github.com/bastianalt/cba-pipeline-public

with client-driven algorithms to adjust the quality bitrate of each video segment based on various signals, such as measured throughput, buffer filling, and derivatives thereof. In contrast, new architectures such as server and network assisted DASH (SAND) [51] introduce network-assisted streaming via DASH-enabled network elements that guide the client, such as accurate throughput measurements and source recommendations. Given the various adaptation algorithms that exist in addition to client-side and network-assisted information, a fundamental question arises on the importance of this context information for the QoE of the video stream.

The problem of video quality adaptation is aggravated in future internet architectures such as NDN. In NDN, content is requested by name rather than location, and each node within the network will either return the requested content or forward the request. Routers are equipped with caches to hold frequently requested content, thereby reducing the round trip time (RTT) of the request while simultaneously saving other network links from redundant content requests. Several attempts to make DASH-style streaming possible over NDN exist, e.g., [52], for which the critical difficulty is that traditional algorithms rarely play to the strengths of NDN where the notion of a connection does not exist. Throughput, for example, is not a trivial signal in NDN as data may not be coming from the same source.

One major challenge with incorporating high-dimensional network context information in video quality adaptation is extracting the most relevant information to the sought QoE metric. We note that the interactions within this context space become complicated given the NDN architecture, where the network topology and cache states influence the streaming session. Our approach introduces a sparse Bayesian contextual bandit algorithm that is fast enough to run online during video playback. The rationale behind the sparsity is that the given information, including network-assisted and client-side measured signals such as buffer filling and throughput, constitutes a high-dimensional context that is difficult to model in detail. Our intuition is that, depending on the client's network context, only a few input variables significantly impact on QoE.

Related Work

In the following, we split the state-of-the-art related work into two categories; i.e., work on ABR quality adaptation, especially in NDN, and related work on contextual bandit algorithms with high-dimensional covariates.

Significant amounts of research have been given to finding streaming architectures capable of satisfying high bitrate and minimal rebuffering requirements at scale. Content delivery network (CDN) brokers such as Conviva[2] allow content producers to use multiple CDNs easily and are becoming crucial to meet user demand [53]. Furthermore, the use of network assistance in CDNs has received significant attention recently as a method of directly providing network details to DASH players. SAND [51] is an international organization for standardization (ISO) standard that permits DASH-enabled in-network entities to communicate with clients and offer them quality of service (QoS) information. SDNDASH [54] is another such architecture aiming to maintain QoE stability across clients, as clients without

2 https://www.conviva.com

network assistance information are prone to misjudge current network conditions, causing QoE to oscillate. Beyond hypertext transfer protocol (HTTP), the capabilities of promising new network paradigms such as NDN pose challenges to video streaming. The authors of [52] compare three state-of-the-art DASH adaptation algorithms over NDN and TCP/IP, finding NDN performance to notably exceed that of TCP/IP given certain network conditions. New adaptation algorithms specific to NDN have also been proposed, such as NDNLive [55], which uses a simple RTT mechanism to stream live content with minimal rebuffering.

In this chapter, we model the video quality adaptation problem as a contextual bandit problem assuming a linear parametrization, which has successfully been used, e.g., for ad placement [56]. Another promising approach is based on cost-sensitive classification in the bandit setting [57]. Recently, [58] has discussed the use of VI in the bandit setting; wherein Thompson sampling is considered to cope with the exploration-exploitation trade-off. By assuming a high-dimensional linear parametrization, we make use of sparse estimation techniques. High-dimensional information arises in video streaming due to the network context. Sparsity has been an essential topic in statistical modeling, and both frequentist and Bayesian approaches have been discussed numerously, e.g., for some prior work discussing robust sparse frequentist estimators see [10, 11]. For computing sparse Bayesian estimates, traditionally double exponential priors which correspond to ℓ_1-regularization have been used. However, these priors often fail due to limited flexibility in their shrinkage behavior. Other approaches that induce sparsity include 'spike-and-slab' priors [59] and continuous shrinkage priors. Between these two, continuous shrinkage priors have the benefit of often being computationally faster [60]. For our approach, we use the three parameter beta normal (TPBN) continuous shrinkage prior introduced by [60], which generalizes diverse shrinkage priors, e.g., the horseshoe prior [61], the Strawderman-Berger prior, the normal-exponential-gamma prior, and the normal-gamma prior. Next, we present the entire model using this sparsity prior.

4.1 A GENERATIVE MODEL FOR DECISION-MAKING WITH SPARSE COVARIATES

We consider a discrete-time setting where at each time step $t \in \mathbb{N}$ an agent can take a decision $u(t) \in \mathcal{U}$. Here, the agent can choose among a finite set of K possible actions; i.e., $\mathcal{U} = \{u^{(i)} \mid i = 1, \dots, K\}$. The decisions can be made based on a set of context variables $\mathbf{x}(t) \coloneqq \{\mathbf{x}(t, u) \mid u \in \mathcal{U}\}$, where the different context variables $\mathbf{x}(t, u) \in \mathbb{R}^n$ at time t are given for each action. The agent decides on an action $u(t)$ and observes for this a noisy reward signal $r(t) \in \mathbb{R}$, which is statistically dependent on the chosen action and the corresponding context as

$$r(t) \sim p(r(t) \mid \mathbf{x}(t, u(t)), u(t)).$$

The goal of the decision-making agent is to get a high cumulative reward over a finite but typically unknown horizon T; i.e., $t \in \{1, \dots, T\}$. Therefore, formally the agent tries to solve the problem

$$\begin{aligned} \underset{u_{[1,T]}}{\text{maximize}} \quad & J\left[u_{[1,T]}\right] = \sum_{t=1}^{T} \mathsf{E}[r(t)] \\ \text{subject to} \quad & r(t) \sim p(r(t) \mid \mathbf{x}(t, u(t)), u(t)), \end{aligned}$$

where $u_{[1,T]} \coloneqq \{u(1), \dots, u(T)\}$ is the action trajectory. What makes this problem especially hard is that the agent does not know how the different actions influence the reward; i.e., it does not know the probability distribution $p(r(t) \mid \mathbf{x}(t, u(t)), u(t))$. However, at a time step t the agent can use the data based on its previous decisions; i.e., $\mathcal{D}(t) \coloneqq \{\mathbf{x}(1), u(1), r(1), \dots \mathbf{x}(t-1), u(t-1), r(t-1), \mathbf{x}(t)\}$. This data can hence be used to learn a reward model, which is presented next. After this, we will discuss the problem of optimal decision-making given this learned model.

THE LIKELIHOOD. As a reward model, we assume a linear model subject to Gaussian noise as

$$p(r(t) \mid u(t), \mathbf{x}(t)) = \mathcal{N}(r(t) \mid \mathbf{x}^\top(t, u(t))\boldsymbol{\beta}(u(t)), \sigma^2(u(t))). \tag{4.1.1}$$

Here, we introduced the regression coefficients $\boldsymbol{\beta} = \{\boldsymbol{\beta}(u) \mid u \in \mathcal{U}\}$, with $\boldsymbol{\beta}(u) \in \mathbb{R}^n$ and the precision parameters $\boldsymbol{\sigma}^{-2} = \{\sigma^{-2}(u) \mid u \in \mathcal{U}\}$, with $\sigma^{-2}(u) \in \mathbb{R}_{\geq 0}$. Hence, we use a linear parameterization for each action independently. This yields the likelihood for the data set $\mathcal{D}(t)$ and parameters $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \boldsymbol{\sigma}^{-2}\}$ as

$$p(\mathcal{D}(t) \mid \boldsymbol{\theta}) = \prod_{s=1}^{t-1} \mathcal{N}(r(s) \mid \mathbf{x}^\top(s, u(s))\boldsymbol{\beta}(u(s)), \sigma^2(u(s))).$$

This likelihood can then be rewritten as

$$\begin{aligned} p(\mathcal{D}(t) \mid \boldsymbol{\theta}) &= \prod_{u \in \mathcal{U}} \prod_{s=1}^{t-1} [\mathcal{N}(r(s) \mid \mathbf{x}^\top(s, u)\boldsymbol{\beta}(u), \sigma^2(u))]^{\mathbb{1}(u(s)=u)} \\ &= \prod_{u \in \mathcal{U}} \prod_{m=1}^{M(u)} \mathcal{N}(r_m(u) \mid \mathbf{x}_m^\top(u)\boldsymbol{\beta}(u), \sigma^2(u)), \end{aligned}$$

where $M(u) \coloneqq \sum_{s=1}^{t-1} \mathbb{1}(u(s) = u)$ is the total number of times action u is chosen. Here, $r_m(u)$ and $\mathbf{x}_m(u)$ are the reward sample and context variable of action u, respectively, when action u was chosen the mth time; i.e.,

$$r_m(u) \coloneqq r(s''), \quad \mathbf{x}_m(u) \coloneqq \mathbf{x}(s'', u), \quad s'' = \max \left\{ s' : \sum_{s=1}^{s'} \mathbb{1}(u(s) = u) \leq m \right\}.$$

Finally, this can be written in vector form as

$$p(\mathcal{D}(t) \mid \boldsymbol{\theta}) = \prod_{u \in \mathcal{U}} \mathcal{N}(\mathbf{r}(u) \mid \mathbf{X}(u)\boldsymbol{\beta}(u), \sigma^2(u)\mathbf{I}),$$

with reward data $\mathbf{r}(u) = [r_1(u), \dots, r_{M(u)}(u)]^\top$, context data $\mathbf{X}(u) = [\mathbf{x}_1(u), \dots, \mathbf{x}_{M(u)}(u)]^\top$, and $\mathbf{I}$ denotes the identity matrix.

THE PRIOR DISTRIBUTION. To learn the model parameters, we use a Bayesian approach. Therefore, we specify a prior distribution, where we use independent priors for all actions, as

$$\begin{aligned} p(\boldsymbol{\beta}) &= \prod_{u \in \mathcal{U}} p(\boldsymbol{\beta}(u)) \\ p(\boldsymbol{\sigma}^2) &= \prod_{u \in \mathcal{U}} p(\sigma^2(u)). \end{aligned}$$

As we will deal with high-dimensional context information, we use a sparsity inducing prior over the regression coefficients $\boldsymbol{\beta}$ to find the most valuable context information. For this, we leverage the three parameter beta normal (TPBN) continuous shrinkage prior [60], which puts on each regression coefficient $\boldsymbol{\beta}(u) = [\beta_1(u), \dots, \beta_n(u)]^\top$, the following hierarchical prior

$$\begin{aligned} \lambda_j(u) &\sim \mathrm{Gam}(\lambda_j(u) \mid b_0, \phi(u)) \\ \tau_j(u) &\sim \mathrm{Gam}(\tau_j(u) \mid a_0, \lambda_j(u)) \qquad j = 1, \dots, n \\ \beta_j(u) &\sim \mathcal{N}(\beta_j(u) \mid 0, \tau_j(u)/\sigma^{-2}(u)). \end{aligned}$$

Here, $\boldsymbol{\tau}(u) = [\tau_1(u), \dots, \tau_n(u)]^\top$ are gamma distributed continuous shrinkage parameters which shrink the corresponding regression coefficient $\beta_j(u)$, as the component $\tau_j(u)$ gets small. The components of the parameters $\boldsymbol{\lambda}(u) = [\lambda_1(u), \dots, \lambda_n(u)]^\top$ control the components of $\boldsymbol{\tau}(u)$ via a global shrinkage parameter parameter $\phi(u)$. For appropriate hyper-parameter choices of a_0 and b_0, different known shrinkage priors are obtained. For example, we use $a_0 = 1/2$, $b_0 = 1/2$, which corresponds to the horseshoe prior [61].

For the estimation of the global shrinkage parameter $\phi(u)$ an additional hierarchy is used as

$$\begin{aligned} \omega(u) &\sim \mathrm{Gam}(\omega(u) \mid 1/2, 1) \\ \phi(u) &\sim \mathrm{Gam}(\phi(u) \mid 1/2, \omega(u)). \end{aligned}$$

For the noise precision, a gamma prior is used as

$$\sigma^{-2}(u) \sim \mathrm{Gam}(\sigma^{-2}(u) \mid c_0/2, d_0/2),$$

with hyper-parameters c_0 and d_0. The PGM [16] of this generative Bayesian model is depicted in Fig. 4.2.

4.2 APPROXIMATE POSTERIOR INFERENCE

Given the generative model, one goal of the decision-making agent is to infer the posterior distribution over the model parameters $\boldsymbol{\theta}$ and prior parameters $\boldsymbol{\Psi} = \{\boldsymbol{\tau}, \boldsymbol{\lambda}, \boldsymbol{\phi}, \boldsymbol{\omega}\}$, with the shorthands

$$\begin{aligned} \boldsymbol{\tau} &= \{\boldsymbol{\tau}(u) \mid u \in \mathcal{U}\} \\ \boldsymbol{\lambda} &= \{\boldsymbol{\lambda}(u) \mid u \in \mathcal{U}\} \\ \boldsymbol{\phi} &= \{\phi(u) \mid u \in \mathcal{U}\} \\ \boldsymbol{\omega} &= \{\omega(u) \mid u \in \mathcal{U}\}. \end{aligned}$$

However, for the posterior distribution

$$p(\boldsymbol{\theta}, \boldsymbol{\Psi} \mid \mathcal{D}(t)) = \frac{p(\mathcal{D}(t) \mid \boldsymbol{\theta}, \boldsymbol{\Psi}) p(\boldsymbol{\theta}, \boldsymbol{\Psi})}{\int p(\mathcal{D}(t) \mid \boldsymbol{\theta}, \boldsymbol{\Psi}) p(\boldsymbol{\theta}, \boldsymbol{\Psi}) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{\Psi}}$$

the evidence in the denominator for the specified model in Section 4.1 is intractable to compute. Therefore, we resort to approximate inference using a variational approach. First, we show how a mean-field variational Bayes estimate can be obtained as proposed in [60]. After this, we will present a new inference scheme for this model leveraging stochastic variational inference (SVI) [62], which can be used to reduce the computation time. This is especially useful in the context of sequential decision-making.

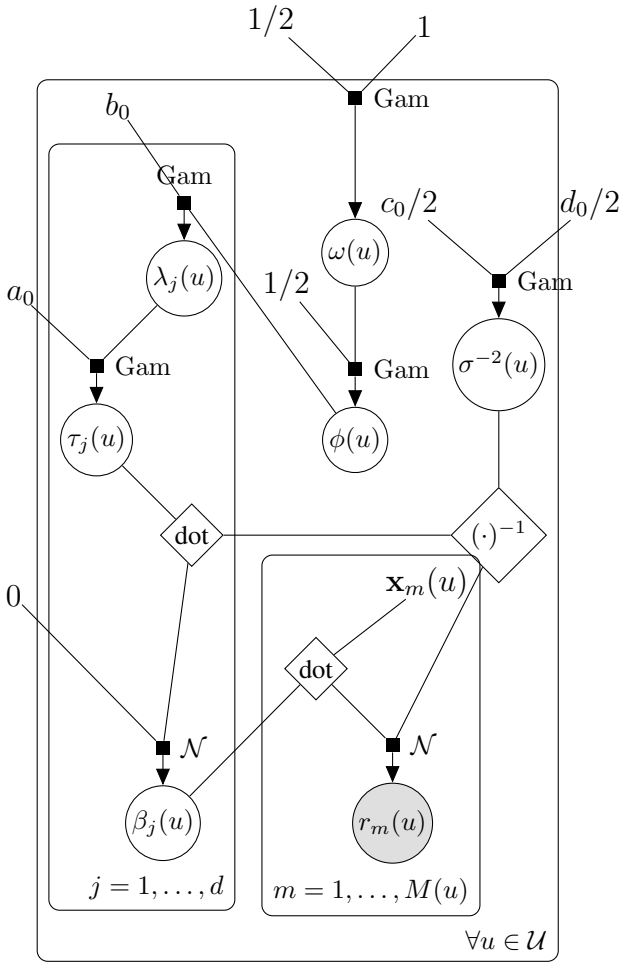


FIGURE 4.2: PGM for the reward model. The Figure shows the generative Bayesian regression model with TPBN prior. Here, 'dot' denotes the inner product.

4.2.1 *Mean-Field Variational Bayesian Inference*

Since exact inference of the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\Psi} \mid \mathcal{D}(t))$ is intractable [22], we apply approximate inference in form of variational Bayes (VB) for posterior inference. We use a mean-field variational approximation, with

$$q(\boldsymbol{\theta}, \boldsymbol{\Psi}) = \prod_{u \in \mathcal{U}} q(\boldsymbol{\beta}(u)) q(\sigma^{-2}(u)) q(\boldsymbol{\tau}(u)) q(\boldsymbol{\lambda}(u)) q(\phi(u)) q(\omega(u)) \tag{4.2.2}$$

for the approximate distribution. The variational distributions are obtained by minimizing the KL divergence between the variational distribution, and the intractable posterior distribution over the class $\mathcal{Q}$ of mean-field approximations; i.e.,

$$\underset{q(\boldsymbol{\theta}, \boldsymbol{\Psi}) \in \mathcal{Q}}{\text{minimize}} \quad \mathsf{KL}(q(\boldsymbol{\theta}, \boldsymbol{\Psi}) \parallel p(\boldsymbol{\theta}, \boldsymbol{\Psi} \mid \mathcal{D}(t))).$$

This yields the equivalent tractable maximization problem over the ELBO $\mathsf{L}[q] \leq \log p(\mathcal{D}(t))$ as

$$\underset{q(\boldsymbol{\theta}, \boldsymbol{\Psi}) \in \mathcal{Q}}{\text{maximize}} \quad \mathsf{L}[q] = \mathsf{E}[\log p(\boldsymbol{\theta}, \boldsymbol{\Psi}, \mathcal{D}(t))] - \mathsf{E}[\log q(\boldsymbol{\theta}, \boldsymbol{\Psi})], \tag{4.2.3}$$

where the expectation is carried out w.r.t. the variational distribution $q(\boldsymbol{\theta}, \boldsymbol{\Psi})$. Using calculus of variations [22, 31], the maximizer of the optimization problem in Eq. (4.2.3) is derived. This yields the optimal variational distribution for $\boldsymbol{\beta}(u)$ with its parameters [60]

$$\begin{aligned} q(\boldsymbol{\beta}(u)) &= \mathcal{N}(\boldsymbol{\beta}(u) \mid \boldsymbol{\mu}_{\boldsymbol{\beta}}(u), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)) \\ \boldsymbol{\mu}_{\boldsymbol{\beta}}(u) &= \left(\mathbf{X}^\top(u)\mathbf{X}(u) + \operatorname{diag}\left(\mathsf{E}\left[\boldsymbol{\tau}^{\circ -1}(u)\right]\right)\right)^{-1} \mathbf{X}^\top(u)\mathbf{r}(u) \\ \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u) &= \mathsf{E}\left[\sigma^{-2}(u)\right]^{-1} \left(\mathbf{X}^\top(u)\mathbf{X}(u) + \operatorname{diag}\left(\mathsf{E}\left[\boldsymbol{\tau}^{\circ -1}(u)\right]\right)\right)^{-1}, \end{aligned} \tag{4.2.4}$$

where $(\cdot)^{\circ -1}$ denotes the Hadamard inverse (element-wise inversion). The optimal variational distribution for $\sigma^{-2}(u)$ is given as

$$\begin{aligned} q(\sigma^{-2}(u)) &= \operatorname{Gam}(\sigma^{-2}(u) \mid c^*(u), d^*(u)) \\ c^*(u) &= \frac{M(u) + n + c_0}{2} \\ d^*(u) &= \frac{1}{2}\Bigg(\mathbf{r}^\top(u)\mathbf{r}(u) - 2\mathbf{r}^\top(u)\mathbf{X}(u)\,\mathsf{E}[\boldsymbol{\beta}(u)] + \sum_{m=1}^{M(u)} \mathbf{x}_m^\top(u)\,\mathsf{E}\left[\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\right]\mathbf{x}_m(u) \\ &\qquad + \sum_{j=1}^{n} \mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\tau_j^{-1}(u)\right] + d_0\Bigg). \end{aligned} \tag{4.2.5}$$

For the shrinkage parameters $\boldsymbol{\tau}(u)$ we have

$$\begin{aligned} q(\boldsymbol{\tau}(u)) &= \prod_{j=1}^{n} \mathcal{GIG}(\tau_j(u) \mid p, a_{\tau,j}(u), b_{\tau,j}(u)) \\ p = a_0 - 1/2, \quad a_{\tau,j}(u) &= 2\,\mathsf{E}[\lambda_j(u)], \quad b_{\tau,j}(u) = \mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\sigma^{-2}(u)\right], \end{aligned} \tag{4.2.6}$$

where $\mathcal{GIG}(x \mid p, a, b)$ denotes the generalized inverse Gaussian (GIG) distribution, see Section A.2.13. Additionally, we have

$$\begin{aligned} q(\boldsymbol{\lambda}(u)) &= \prod_{j=1}^{n} \operatorname{Gam}(\lambda_j(u) \mid a_{\lambda,j}(u), b_{\lambda,j}(u)) \\ a_{\lambda,j}(u) = a_0 + b_0, \quad b_{\lambda,j}(u) &= \mathsf{E}[\tau_j(u)] + \mathsf{E}[\phi(u)]. \end{aligned} \tag{4.2.7}$$

The variational distribution for $\phi(u)$ is

$$\begin{aligned} q(\phi(u)) &= \operatorname{Gam}(\phi(u) \mid a_\phi(u), b_\phi(u)) \\ a_\phi(u) = nb_0 + 1/2, \quad b_\phi(u) &= \mathsf{E}[\omega(u)] + \sum_{j=1}^{n} \mathsf{E}[\lambda_j(u)] \end{aligned} \tag{4.2.8}$$

and finally the variational distribution

$$\begin{aligned} q(\omega(u)) &= \operatorname{Gam}(\omega(u) \mid a_\omega(u), b_\omega(u)) \\ a_\omega(u) = 1, \quad b_\omega(u) &= \mathsf{E}[\phi(u)] + 1. \end{aligned} \tag{4.2.9}$$

In Fig. 4.3 the corresponding PGM of the mean-field approximation is depicted. Note the factorization of the RVs, which enables tractable posterior inference in comparison to the

PGM for the coupled Bayesian regression in Fig. 4.2. The moments of these variational distributions are given as

$$\begin{aligned}
\mathsf{E}[\boldsymbol{\beta}(u)] &= \boldsymbol{\mu}_{\boldsymbol{\beta}}(u), \quad \mathsf{E}\big[\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\big] = \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u) + \boldsymbol{\mu}_{\boldsymbol{\beta}}(u)\boldsymbol{\mu}_{\boldsymbol{\beta}}^\top(u), \\
\mathsf{E}\big[\sigma^{-2}(u)\big] &= c^*(u)/d^*(u), \quad \mathsf{E}[\lambda_j(u)] = a_{\lambda,j}(u)/b_{\lambda,j}(u), \\
\mathsf{E}[\phi(u)] &= a_\phi(u)/b_\phi(u), \quad \mathsf{E}[\omega(u)] = a_\omega(u)/b_\omega(u), \\
\mathsf{E}[\tau_j(u)] &= \left(\frac{b_{\tau,j}(u)}{a_{\tau,j}(u)}\right)^{1/2} \frac{\mathrm{K}_{p+1}(v_j(u))}{\mathrm{K}_p(v_j(u))}, \quad \mathsf{E}\big[\tau_j^{-1}(u)\big] = \left(\frac{a_{\tau,j}(u)}{b_{\tau,j}(u)}\right)^{1/2} \frac{\mathrm{K}_{1-p}(v_j(u))}{\mathrm{K}_{-p}(v_j(u))},
\end{aligned} \tag{4.2.10}$$

with $v_j(u) = \sqrt{2a_{\tau,j}(u)b_{\tau,j}(u)}$ and $\mathrm{K}_p(\cdot)$ is the modified Bessel function of second kind.

4.2.1.1 *Calculation of the Evidence Lower Bound*

Further, we newly derive an expression for the ELBO in Eq. (4.2.3) under the optimal variational distributions.

For the derivation, we note that the joint distribution $p(\boldsymbol{\theta}, \boldsymbol{\Psi}, \mathcal{D}(t))$ over parameters and data factorizes as

$$\begin{aligned}
p(\boldsymbol{\theta}, \boldsymbol{\Psi}, \mathcal{D}(t)) = \prod_{u \in \mathcal{U}} & p(\mathbf{r}(u) \mid \mathbf{X}(u), \boldsymbol{\beta}(u), \sigma^{-2}(u)) p(\boldsymbol{\beta}(u) \mid \sigma^{-2}(u), \boldsymbol{\tau}(u)) p(\sigma^{-2}(u)) \\
& \cdot p(\boldsymbol{\tau}(u) \mid \boldsymbol{\lambda}(u)) p(\boldsymbol{\lambda}(u) \mid \phi(u)) p(\phi(u) \mid \omega(u)) p(\omega(u)).
\end{aligned} \tag{4.2.11}$$

Using the mean-field variational distribution as given in Eq. (4.2.2), the ELBO in Eq. (4.2.3) can be calculated as

$$\begin{aligned}
\mathsf{L}[q] = \sum_{u \in \mathcal{U}} & \mathsf{E}\big[\log p(\mathbf{r}(u) \mid \mathbf{X}(u), \boldsymbol{\beta}(u), \sigma^{-2}(u))\big] + \mathsf{E}\big[\log p(\boldsymbol{\beta}(u) \mid \sigma^{-2}(u), \boldsymbol{\tau}(u))\big] \\
& + \mathsf{E}\big[\log p(\sigma^{-2}(u))\big] + \mathsf{E}[\log p(\boldsymbol{\tau}(u) \mid \boldsymbol{\lambda}(u))] + \mathsf{E}[\log p(\boldsymbol{\lambda}(u) \mid \phi(u))] \\
& + \mathsf{E}[\log p(\phi(u) \mid \omega(u))] + \mathsf{E}[\log p(\omega(u))] - \mathsf{E}[\log q(\boldsymbol{\beta}(u))] \\
& - \mathsf{E}\big[\log q(\sigma^{-2}(u))\big] - \mathsf{E}[\log q(\boldsymbol{\tau}(u))] - \mathsf{E}[\log q(\boldsymbol{\lambda}(u))] \\
& - \mathsf{E}[\log q(\phi(u))] - \mathsf{E}[\log q(\omega(u))].
\end{aligned} \tag{4.2.12}$$

Next, we calculate the expectations w.r.t. the variational distribution in Eq. (4.2.2) for the log-factorized joint distribution $p(\boldsymbol{\theta}, \boldsymbol{\Psi}, \mathcal{D}(t))$ in Eq. (4.2.11). This yields the following expressions

$$\begin{aligned}
&\mathsf{E}\big[\log p(\mathbf{r}(u) \mid \mathbf{X}(u), \boldsymbol{\beta}(u), \sigma^{-2}(u))\big] = \mathsf{E}\big[\log \mathcal{N}(\mathbf{r}(u) \mid \mathbf{X}(u)\boldsymbol{\beta}(u), \sigma^2(u)\mathbf{I})\big] \\
&= -\frac{M(u)}{2}\log(2\pi) + \frac{M(u)}{2}\mathsf{E}\big[\log \sigma^{-2}(u)\big] - \frac{1}{2}\mathsf{E}\big[\sigma^{-2}(u)\big]\Big(\mathbf{r}^\top(u)\mathbf{r}(u) \\
&\qquad -2\mathbf{r}^\top(u)\,\mathsf{E}[\boldsymbol{\beta}(u)] + \sum_{m=1}^{M(u)} \mathbf{x}_m^\top(u)\,\mathsf{E}\big[\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\big]\mathbf{x}_m(u)\Big)
\end{aligned} \tag{4.2.13}$$

$$\begin{aligned}\mathsf{E}\big[\log p(\boldsymbol{\beta}(u) \mid \sigma^{-2}(u), \boldsymbol{\tau}(u))\big] &= \mathsf{E}\left[\log\left(\prod_{j=1}^{n} \mathcal{N}(\beta_j(u) \mid 0, \tau_j(u)/\sigma^{-2}(u))\right)\right] \\ &= -\frac{n}{2}\log(2\pi) + \frac{n}{2}\,\mathsf{E}\big[\log\sigma^{-2}(u)\big] - \frac{1}{2}\sum_{j=1}^{n}\mathsf{E}[\log\tau_j(u)] - \frac{1}{2}\sum_{j=1}^{n}\mathsf{E}\big[\sigma^{-2}(u)\big] \\ &\quad \cdot \mathsf{E}\big[\beta_j^2(u)\big]\,\mathsf{E}\big[\tau_j^{-1}(u)\big]\end{aligned} \tag{4.2.14}$$

$$\begin{aligned}\mathsf{E}\big[\log p(\sigma^{-2}(u))\big] &= \mathsf{E}\big[\log \mathrm{Gam}(\sigma^{-2}(u) \mid c_0/2, d_0/2)\big] \\ &= \left(\frac{c_0}{2} - 1\right)\mathsf{E}\big[\log\sigma^{-2}(u)\big] - \frac{d_0}{2}\,\mathsf{E}\big[\sigma^{-2}(u)\big] + \frac{c_0}{2}\log\frac{d_0}{2} - \log\Gamma\left(\frac{c_0}{2}\right)\end{aligned} \tag{4.2.15}$$

$$\begin{aligned}\mathsf{E}[\log p(\boldsymbol{\tau}(u) \mid \boldsymbol{\lambda}(u))] &= \mathsf{E}\left[\log\left(\prod_{j=1}^{n}\mathrm{Gam}(\tau_j(u) \mid a_0, \lambda_j(u))\right)\right] \\ &= -n\log\Gamma(a_0) + a_0\sum_{j=1}^{n}\mathsf{E}[\log\lambda_j(u)] + (a_0 - 1)\sum_{j=1}^{n}\mathsf{E}[\log\tau_j(u)] - \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)] \\ &\quad \cdot \mathsf{E}[\tau_j(u)]\end{aligned} \tag{4.2.16}$$

$$\begin{aligned}\mathsf{E}[\log p(\boldsymbol{\lambda}(u) \mid \phi(u))] &= \mathsf{E}\left[\log\left(\prod_{j=1}^{n}\mathrm{Gam}(\lambda_j(u) \mid b_0, \phi(u))\right)\right] \\ &= -n\log\Gamma(b_0) + b_0 n\,\mathsf{E}[\log\phi(u)] + (b_0 - 1)\sum_{j=1}^{n}\mathsf{E}[\log\lambda_j(u)] - \mathsf{E}[\phi(u)] \\ &\quad \cdot \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)]\end{aligned} \tag{4.2.17}$$

$$\begin{aligned}\mathsf{E}[\log p(\phi(u) \mid \omega(u))] &= \mathsf{E}[\log \mathrm{Gam}(\phi(u) \mid 1/2, \omega(u))] \\ &= \frac{1}{2}\,\mathsf{E}[\log\omega(u)] - \log\Gamma\left(\frac{1}{2}\right) - \frac{1}{2}\,\mathsf{E}[\log\phi(u)] - \mathsf{E}[\omega(u)]\,\mathsf{E}[\phi(u)]\end{aligned} \tag{4.2.18}$$

$$\begin{aligned}\mathsf{E}[\log p(\omega(u))] &= \mathsf{E}[\log \mathrm{Gam}(\omega(u) \mid 1/2, 1)] \\ &= \frac{1}{2}\log\frac{1}{2} - \log\Gamma\left(\frac{1}{2}\right) - \frac{1}{2}\,\mathsf{E}[\log\omega(u)] - \frac{1}{2}\,\mathsf{E}[\omega(u)].\end{aligned} \tag{4.2.19}$$

Additionally, the entropy terms, i.e., $\mathsf{E}[\log q(\boldsymbol{\theta}, \boldsymbol{\Psi})] \equiv -\,\mathsf{H}[\boldsymbol{\theta}, \boldsymbol{\Psi}]$, in Eq. (4.2.3) calculate to

$$\begin{aligned}\mathsf{E}[\log q(\boldsymbol{\beta}(u))] &= \mathsf{E}\big[\log\mathcal{N}(\boldsymbol{\beta}(u) \mid \boldsymbol{\mu}_{\boldsymbol{\beta}}(u), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u))\big] \\ &= -\frac{n}{2}\log\frac{2}{\pi} - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)|\end{aligned} \tag{4.2.20}$$

$$\begin{aligned}\mathsf{E}\big[\log q(\sigma^{-2}(u))\big] &= \mathsf{E}\big[\log\mathrm{Gam}(\sigma^{-2}(u) \mid c^*(u), d^*(u))\big] \\ &= c^*(u)\log d^*(u) - \log\Gamma(c^*(u)) + (c^*(u) - 1)\,\mathsf{E}\big[\log\sigma^{-2}(u)\big] - d^*(u)\,\mathsf{E}\big[\sigma^{-2}(u)\big]\end{aligned} \tag{4.2.21}$$

$$\mathsf{E}[\log q(\boldsymbol{\tau}(u))] = \mathsf{E}\left[\log\left(\prod_{j=1}^{n} \mathcal{GIG}(\tau_j(u) \mid p, a_{\tau,j}(u), b_{\tau,j}(u))\right)\right]$$
$$= \left(\frac{a_0}{2} - \frac{1}{2}\right)\left[p\log 2 - p\log \mathsf{E}\left[\sigma^{-2}(u)\right] + \sum_{j=1}^{n}\log \mathsf{E}[\lambda_j(u)] - \sum_{j=1}^{n}\log \mathsf{E}\left[\beta_j^2(u)\right]\right]$$
$$-p\log 2 - \sum_{j=1}^{n}\log \mathrm{K}_{a_0-\frac{1}{2}}(v_j) + \left(a_0 - \frac{3}{2}\right)\sum_{j=1}^{n}\mathsf{E}[\log \tau_j(u)] - \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)]$$
$$\cdot\, \mathsf{E}[\tau_j(u)] - \frac{1}{2}\,\mathsf{E}\left[\sigma^{-2}(u)\right]\sum_{j=1}^{n}\mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\tau_j^{-1}(u)\right] \tag{4.2.22}$$

$$\mathsf{E}[\log q(\boldsymbol{\lambda}(u))] = \mathsf{E}\left[\log\left(\prod_{j=1}^{n}\mathrm{Gam}(\lambda_j(u) \mid a_{\lambda,j}(u), b_{\lambda,j}(u))\right)\right]$$
$$= -n\log\Gamma(a_0+b_0) + (a_0+b_0)\sum_{j=1}^{n}\log\left(\mathsf{E}[\tau_j(u)] + \mathsf{E}[\phi(u)]\right) + (a_0+b_0-1)$$
$$\cdot\sum_{j=1}^{n}\mathsf{E}[\log\lambda_j(u)] - \sum_{j=1}^{n}\mathsf{E}[\tau_j(u)]\,\mathsf{E}[\lambda_j(u)] - \mathsf{E}[\phi(u)]\sum_{j=1}^{n}\mathsf{E}[\tau_j(u)] \tag{4.2.23}$$

$$\mathsf{E}[\log q(\phi(u))] = \mathsf{E}[\log \mathrm{Gam}(\phi(u) \mid a_\phi(u), b_\phi(u))]$$
$$= \left(nb_0 + \frac{1}{2}\right)\log\left(\mathsf{E}[\omega(u)] + \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)]\right) - \log\Gamma\left(nb_0 + \frac{1}{2}\right) + \left(nb_0 - \frac{1}{2}\right)$$
$$\cdot\,\mathsf{E}[\log\phi(u)] - \mathsf{E}[\omega(u)]\,\mathsf{E}[\phi(u)] - \mathsf{E}[\phi(u)]\sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)] \tag{4.2.24}$$

$$\mathsf{E}[\log q(\omega(u))] = \mathsf{E}[\log \mathrm{Gam}(\omega(u) \mid a_\omega(u), b_\omega(u))]$$
$$= \log\left(\mathsf{E}[\phi(u)] + 1\right) - \mathsf{E}[\omega(u)]\,\mathsf{E}[\phi(u)] - \mathsf{E}[\omega(u)]. \tag{4.2.25}$$

Using Eqs. (4.2.13) to (4.2.25) in Eq. (4.2.12), yields the expression for the ELBO for the optimal variational distributions as

$$\begin{aligned}
\mathsf{L}[q] = \sum_{u\in\mathcal{U}} & -\frac{M(u)}{2}\log(2\pi) + \frac{c_0}{2}\log\frac{d_0}{2} - \log\Gamma\left(\frac{c_0}{2}\right) - c^*(u)\log d^*(u) \\
& + \log\Gamma(c^*(u)) + \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)]\,\mathsf{E}[\tau_j(u)] + \mathsf{E}[\phi(u)]\sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)] \\
& - n\log\Gamma(a_0) - n\log\Gamma(b_0) - 2\log\Gamma\left(\frac{1}{2}\right) + \log 2\left(\left(\frac{5}{4}-\frac{a_0}{2}\right)n - \frac{1}{2}\right) \\
& + n\log\Gamma(a_0+b_0) + \log\Gamma\left(nb_0+\frac{1}{2}\right) + \frac{1}{2}\mathsf{E}[\omega(u)] + \mathsf{E}[\omega(u)]\,\mathsf{E}[\phi(u)] \\
& + \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)| + \left(\frac{a_0}{2}-\frac{1}{4}\right)\left(n\log\mathsf{E}\left[\sigma^{-2}(u)\right] - \sum_{j=1}^{n}\log\mathsf{E}[\lambda_j(u)]\right) \\
& + \left(\frac{a_0}{2}-\frac{1}{4}\right)\sum_{j=1}^{n}\log\mathsf{E}\left[\beta_j^2(u)\right] - \sum_{j=1}^{n}\log \mathrm{K}_{a_0-\frac{1}{2}}(v_j(u)) - \frac{1}{2}\mathsf{E}\left[\sigma^{-2}(u)\right] \\
& \cdot \sum_{j=1}^{n}\mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\tau_j^{-1}(u)\right] - \left(nb_0+\frac{1}{2}\right)\log(\mathsf{E}[\omega(u)] + \sum_{j=1}^{n}\mathsf{E}[\lambda_j(u)]) \\
& - (a_0+b_0)\sum_{j=1}^{n}\log(\mathsf{E}[\tau_j(u)] + \mathsf{E}[\phi(u)]) - \log(\mathsf{E}[\phi(u)]+1).
\end{aligned} \tag{4.2.26}$$

Finally, optimizing Eq. (4.2.26) can be achieved by cycling through the coupled moments in Eq. (4.2.10) and the previously presented parameters of the variational distributions, see Eqs. (4.2.4) to (4.2.9), which corresponds to a coordinate ascent algorithm on the ELBO $\mathsf{L}[q]$. This yields a local optimum of the objective in Eq. (4.2.3). The full algorithm is shown in Fig. 4.4b.

4.2.2 *Stochastic Variational Inference*

The VB update as presented in Section 4.2.1 requires cycling through the coupled moments in Eq. (4.2.10) until convergence. Since this computation can be costly, we propose a new scheme based on SVI [62]. Here, the ELBO $\mathsf{L}[q]$ is optimized by using stochastic approximation [63], where we use a natural gradient descent algorithm [64]. The natural gradient w.r.t. the mean-field variational distributions can be obtained utilizing the natural parameterization of the corresponding exponential family distribution [23].

For this, we use the shorthand $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \boldsymbol{\Psi}\}$ for all parameters. In the presented mean-field approximation in Section 4.2.1, the variational distribution factorizes over all single parameters as $q(\boldsymbol{\Theta}) = \prod_i q(\Theta_i)$. For the presented model, each factor $q(\Theta_i)$ is a member of the exponential family. This means that the factors can be written as a probability density of the form

$$q(\Theta_i) \equiv q(\Theta_i \mid \boldsymbol{\eta}_i) = h(\Theta_i)\exp\{\boldsymbol{\eta}_i^\top \mathbf{s}(\Theta_i) - A(\boldsymbol{\eta}_i)\}.$$

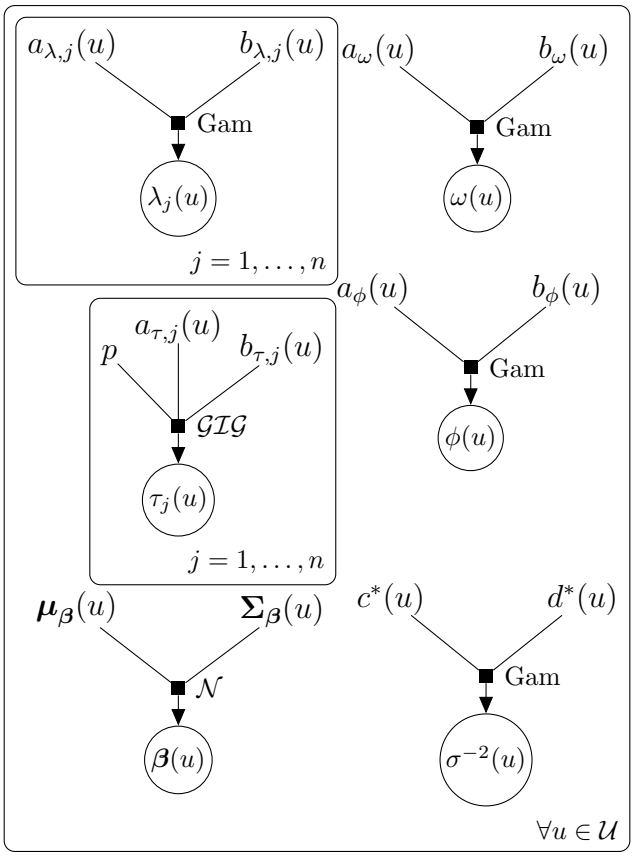


FIGURE 4.3: PGM for the optimal mean-field approximation. The factorized distribution is an approximation to the true posterior. The optimal variational parameters are learned to provide the data dependency.

Here, $h(\Theta_i)$ denotes the base measure, $\boldsymbol{\eta}_i$ are the natural parameters, $\mathrm{s}(\Theta_i)$ are the sufficient statistics of the natural parameters and $A(\boldsymbol{\eta}_i)$ is the log-normalizer.

Next, we compute the natural gradient of the ELBO $\mathsf{L}[q] \equiv \mathsf{L}(\boldsymbol{\eta})$ with respect to the natural parameters of the factorized variational distributions for each variational factor $q(\Theta_i)$. Note that the natural gradient of a function $f(\cdot)$ w.r.t. a variable $\boldsymbol{\eta}$ is given as

$$\hat{\nabla}_{\boldsymbol{\eta}} f(\boldsymbol{\eta}) = G(\boldsymbol{\eta})^{-1} \nabla_{\boldsymbol{\eta}} f(\boldsymbol{\eta}),$$

where $G(\boldsymbol{\eta}) = \mathsf{E}\Big[\left(\nabla_{\boldsymbol{\eta}} \log q(\Theta \mid \boldsymbol{\eta})\right) \left(\nabla_{\boldsymbol{\eta}} \log q(\Theta \mid \boldsymbol{\eta})\right)^\top\Big]$ is the Fisher information matrix of the RV Θ which depends on the parameters $\boldsymbol{\eta}$; i.e., $q(\Theta \mid \boldsymbol{\eta})$. For the natural gradient of the ELBO in Eq. (4.2.26) it can be shown that the natural gradient w.r.t. to a natural parameter $\boldsymbol{\eta}$ computes to

$$\hat{\nabla}_{\boldsymbol{\eta}} \mathsf{L}(\boldsymbol{\eta}) = \mathsf{E}[\boldsymbol{\eta}'] - \boldsymbol{\eta},$$

where the parameters $\boldsymbol{\eta}'$ are the natural parameters of the full conditional distribution $p(\Theta_m \mid \Theta_{-m}, \mathcal{D}(t))$, with Θ_{-m} denoting the tuple of all variables but Θ_m. Using a gradient update the variational approximation can be found as

$$\boldsymbol{\eta}^{(l+1)} = \boldsymbol{\eta}^{(l)} + \gamma_l \hat{\nabla}_{\boldsymbol{\eta}} \mathsf{L}(\boldsymbol{\eta}^{(l)}), \tag{4.2.27}$$

where l is the iteration step of the algorithm and γ_l is a step size parameter, we choose as $\gamma_l = 1/l$. Given the vanilla natural gradient update in Eq. (4.2.27), we use random sub-sampling of the data, which enables constructing a stochastic approximation algorithm. For this algorithm, $\mathsf{E}[\boldsymbol{\eta}']$ is replaced by an unbiased estimate $\hat{\boldsymbol{\eta}}$, which yields a stochastic natural gradient ascent algorithm on the ELBO in the form

$$\boldsymbol{\eta}^{(l+1)} = (1 - \gamma_l)\boldsymbol{\eta}^{(l)} + \gamma_l \hat{\boldsymbol{\eta}}. \tag{4.2.28}$$

4.2.2.1 *Calculating the Natural Parameter Updates*

In the case of the regression problem presented in Section 4.1, we can sample one data point $(u, \mathbf{x}_m(u), r_m(u))$ from the set of observed data points and replicate it $M(u)$ times to calculate the estimates $\{\hat{\boldsymbol{\eta}}_{\boldsymbol{\beta}}(u), \hat{\boldsymbol{\eta}}_{\sigma^{-2}}(u), \hat{\boldsymbol{\eta}}_{\tau,j}(u), \hat{\boldsymbol{\eta}}_{\lambda,j}(u), \hat{\boldsymbol{\eta}}_{\phi}(u), \hat{\boldsymbol{\eta}}_{\omega}(u)\}$. To derive the intermediate estimates, we need to compute the expectation $\mathsf{E}[\boldsymbol{\eta}']$ for the natural parameters of the full conditional distributions. The full conditional distributions [60] are given as

$$p(\boldsymbol{\beta}(u) \mid \mathbf{r}(u), \mathbf{X}(u), \sigma^{-2}(u), \boldsymbol{\tau}(u)) = \mathcal{N}(\boldsymbol{\beta}(u) \mid \boldsymbol{\mu}'_{\boldsymbol{\beta}}(u), \boldsymbol{\Sigma}'_{\boldsymbol{\beta}}(u))$$
$$\boldsymbol{\mu}'_{\boldsymbol{\beta}}(u) = \left(\mathbf{X}^\top(u)\mathbf{X}(u) + \operatorname{diag}\left(\boldsymbol{\tau}^{\circ -1}(u)\right)\right)^{-1} \mathbf{X}^\top(u)\mathbf{r}(u)$$
$$\boldsymbol{\Sigma}'_{\boldsymbol{\beta}}(u) = \sigma^{-2}(u)^{-1}\left(\mathbf{X}^\top(u)\mathbf{X}(u) + \operatorname{diag}\left(\boldsymbol{\tau}^{\circ -1}(u)\right)\right)^{-1},$$

$$p(\sigma^{-2}(u) \mid \mathbf{r}(u), \mathbf{X}(u), \boldsymbol{\beta}(u), \boldsymbol{\tau}(u)) = \operatorname{Gam}(\sigma^{-2}(u) \mid c'(u), d'(u))$$
$$c'(u) = \frac{M(u) + n + c_0}{2}$$
$$d'(u) = \frac{1}{2}\left(\mathbf{r}^\top(u)\mathbf{r}(u) - 2\mathbf{r}^\top(u)\mathbf{X}(u)\boldsymbol{\beta}(u) + \sum_{m=1}^{M(u)} \mathbf{x}_m^\top(u)\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\mathbf{x}_m(u) + \sum_{j=1}^{n} \beta_j^2(u)\tau_j^{-1}(u) + d_0\right),$$

$$p(\boldsymbol{\tau}(u) \mid \boldsymbol{\beta}(u), \sigma^{-2}(u), \boldsymbol{\lambda}(u)) = \prod_{j=1}^{n} \mathcal{GIG}(\tau_j(u) \mid p', a'_{\tau,j}(u), b'_{\tau,j}(u))$$
$$p' = a_0 - 1/2, \quad a'_{\tau,j}(u) = 2\lambda_j(u), \quad b'_{\tau,j}(u) = \beta_j^2(u)\sigma^{-2}(u),$$

$$p(\boldsymbol{\lambda}(u) \mid \boldsymbol{\tau}(u), \phi(u)) = \prod_{j=1}^{n} \operatorname{Gam}(\lambda_j(u) \mid a'_{\lambda,j}(u), b'_{\lambda,j}(u))$$
$$a'_{\lambda,j}(u) = a_0 + b_0, \quad b'_{\lambda,j}(u) = \tau_j(u) + \phi(u),$$

$$p(\phi(u) \mid \boldsymbol{\lambda}(u), (u)) = \operatorname{Gam}(\phi(u) \mid a'_{\phi}(u), b'_{\phi}(u))$$
$$a'_{\phi}(u) = nb_0 + 1/2, \quad b'_{\phi}(u) = \sum_{j=1}^{n} \lambda_j(u) + \omega(u)$$

and

$$p(\omega(u) \mid \phi(u)) = \mathrm{Gam}(\omega(u) \mid a'_\omega(u), b'_\omega(u)).$$
$$a'_\omega(u) = 1, \quad b'_\omega(u) = \phi(u) + 1.$$

Next, the transformation of the parameters of the full conditional distributions into their exponential family parameterization can be obtained via

$$\begin{aligned}
\boldsymbol{\eta}'_{\boldsymbol{\beta}}(u) &= \left({\boldsymbol{\Sigma}'_{\boldsymbol{\beta}}}^{-1}(u)\boldsymbol{\mu}'_{\boldsymbol{\beta}}(u), -\frac{1}{2}{\boldsymbol{\Sigma}'_{\boldsymbol{\beta}}}^{-1}(u) \right) \\
\boldsymbol{\eta}'_{\sigma^{-2}}(u) &= (c'(u) - 1, -d'(u)) \\
\boldsymbol{\eta}'_{\tau,j}(u) &= \left(p' - 1, a'_{\tau,j}(u)/2, b'_{\tau,j}(u)/2\right) \\
\boldsymbol{\eta}'_{\lambda,j}(u) &= \left(a'_{\lambda,j}(u) - 1, -b'_{\lambda,j}(u)\right) \\
\boldsymbol{\eta}'_{\phi}(u) &= \left(a'_{\phi}(u) - 1, -b'_{\phi}(u)\right) \\
\boldsymbol{\eta}'_{\omega}(u) &= \left(a'_{\omega}(u) - 1, -b'_{\omega}(u)\right).
\end{aligned}$$

Now, we can carry out the expectation w.r.t. the variational distributions as

$$\begin{aligned}
\mathsf{E}\left[\boldsymbol{\eta}'_{\boldsymbol{\beta}}(u)\right] &= \Bigg(\mathsf{E}\left[\sigma^{-2}(u)\right]\mathbf{X}^\top(u)\mathbf{r}(u), -\frac{1}{2}\,\mathsf{E}\left[\sigma^{-2}(u)\right]\left(\mathbf{X}^\top(u)\mathbf{X}(u)\right. \\
&\qquad \left. + \operatorname{diag}\left(\mathsf{E}\left[\boldsymbol{\tau}^{\circ -1}(u)\right]\right)\right)\Bigg) \\
\mathsf{E}[\boldsymbol{\eta}'_{\sigma^{-2}}(u)] &= \Bigg(\frac{M(u) + n + c_0}{2} - 1, -\frac{1}{2}\left(\mathbf{r}^\top(u)\mathbf{r}(u) - 2\mathbf{r}^\top(u)\mathbf{X}(u)\,\mathsf{E}[\boldsymbol{\beta}(u)]\right. \\
&\quad \left. + \sum_{m=1}^{M(u)} \mathbf{x}_m^\top(u)\,\mathsf{E}\left[\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\right]\mathbf{x}_m(u) + \sum_{j=1}^{n} \mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\tau_j^{-1}(u)\right] + d_0 \right) \Bigg) \\
\mathsf{E}\left[\boldsymbol{\eta}'_{\tau,j}(u)\right] &= \left(a_0 - \frac{3}{2}, -\mathsf{E}[\lambda_j(u)], \mathsf{E}\left[\beta_j^2(u)\right]\mathsf{E}\left[\sigma^{-2}(u)\right]/2 \right) \\
\mathsf{E}\left[\boldsymbol{\eta}'_{\lambda,j}(u)\right] &= (a_0 + b_0 - 1, -\mathsf{E}[\tau_j(u)] - \mathsf{E}[\phi(u)]) \\
\mathsf{E}\left[\boldsymbol{\eta}'_{\phi}(u)\right] &= \left(nb_0 - \frac{1}{2}, -\mathsf{E}[\omega(u)] - \sum_{j=1}^{n} \mathsf{E}[\lambda_j(u)] \right) \\
\mathsf{E}[\boldsymbol{\eta}'_{\omega}(u)] &= (0, -\mathsf{E}[\phi(u)] - 1).
\end{aligned}$$

Replicating one data point $(u, \mathbf{x}_m(u), r_m(u))$ $M(u)$-times yields the intermediate estimates of the natural parameters as

$$\begin{aligned}
\hat{\boldsymbol{\eta}}_{\boldsymbol{\beta}}(u) &= \Bigg(\mathsf{E}\big[\sigma^{-2}(u)\big] M(u)\mathbf{x}_m(u) r_m(u), -\frac{1}{2}\,\mathsf{E}\big[\sigma^{-2}(u)\big]\,\big(M(u)\mathbf{x}_m(u)\mathbf{x}_m^\top(u) \\
&\qquad\qquad + \operatorname{diag}\big(\mathsf{E}\big[\boldsymbol{\tau}^{\circ -1}(u)\big]\big)\big)\Bigg) \\
\hat{\boldsymbol{\eta}}_{\sigma^{-2}}(u) &= \Bigg(\frac{M(u)+n+c_0}{2} - 1, -\frac{1}{2}\,\big(M(u) r_m^2(u) - 2M(u) r_m(u)\mathbf{x}_m^\top(u)\,\mathsf{E}[\boldsymbol{\beta}(u)] \\
&\qquad + M(u)\mathbf{x}_m^\top(u)\,\mathsf{E}\big[\boldsymbol{\beta}(u)\boldsymbol{\beta}^\top(u)\big]\mathbf{x}_m(u) + \sum_{j=1}^{n} \mathsf{E}\big[\beta_j^2(u)\big]\,\mathsf{E}\big[\tau_j^{-1}(u)\big] + d_0\big)\Bigg) \\
\hat{\boldsymbol{\eta}}_{\tau,j}(u) &= \left(a_0 - \frac{3}{2}, -\mathsf{E}[\lambda_j(u)], \mathsf{E}\big[\beta_j^2(u)\big]\,\mathsf{E}\big[\sigma^{-2}(u)\big]/2 \right) \\
\hat{\boldsymbol{\eta}}_{\lambda,j}(u) &= (a_0 + b_0 - 1, -\mathsf{E}[\tau_j(u)] - \mathsf{E}[\phi(u)]) \\
\hat{\boldsymbol{\eta}}_{\phi}(u) &= \left(n b_0 - \frac{1}{2}, -\mathsf{E}[\omega(u)] - \sum_{j=1}^{n} \mathsf{E}[\lambda_j(u)] \right) \\
\hat{\boldsymbol{\eta}}_{\omega}(u) &= (0, -\mathsf{E}[\phi(u)] - 1)\,.
\end{aligned} \tag{4.2.29}$$

The intermediate estimates in Eq. (4.2.29) can be calculated using the moments in Eq. (4.2.10), where the transformation from the natural parametrization to the variational parametrization can be calculated via

$$\begin{aligned}
\big(\boldsymbol{\mu}_{\boldsymbol{\beta}}(u), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)\big) &= \left(-\frac{1}{2}\Big(\boldsymbol{\eta}_{\boldsymbol{\beta}}^{(2)}(u)\Big)^{-1} \boldsymbol{\eta}_{\boldsymbol{\beta}}^{(1)}(u), -\frac{1}{2}\Big(\boldsymbol{\eta}_{\boldsymbol{\beta}}^{(2)}(u)\Big)^{-1} \right) \\
(c^*(u), d^*(u)) &= \Big(\eta_{\sigma^{-2}}^{(1)}(u) + 1, -\eta_{\sigma^{-2}}^{(2)}(u)\Big) \\
(p, a_{\tau,j}(u), b_{\tau,j}(u)) &= \Big(\eta_{\tau,j}^{(1)}(u) + 1, -2\eta_{\tau,j}^{(2)}(u), 2\eta_{\tau,j}^{(3)}(u)\Big) \\
(a_{\lambda,j}(u), b_{\lambda,j}(u)) &= \Big(\eta_{\lambda,j}^{(1)}(u) + 1, -\eta_{\lambda,j}^{(2)}(u)\Big) \\
(a_{\phi}(u), b_{\phi}(u)) &= \Big(\eta_{\phi}^{(1)}(u) + 1, -\eta_{\phi}^{(2)}(u)\Big) \\
(a_{\omega}(u), b_{\omega}(u)) &= \big(\eta_{\omega}^{(1)}(u) + 1, -\eta_{\omega}^{(2)}(u)\big)\,.
\end{aligned} \tag{4.2.30}$$

Here, we denote by $\eta^{(i)}$ or $\boldsymbol{\eta}^{(i)}$ the ith variable of the tuple of corresponding natural parameters $\boldsymbol{\eta}$. The gradient update as in Eq. (4.2.28), with random sub-sampling, is performed until the ELBO L converges. For a pseudo-code of this SVI algorithm see Fig. 4.4c.

4.2.2.2 *Using One Gradient Step in Stochastic Variational Inference*

Since optimizing the ELBO $\mathsf{L}[q]$ until convergence with both VB and SVI is computationally expensive, we present a new inference method called one-step stochastic variational inference (OS-SVI), which scales significantly in a sequential decision-making setting.

Here, at the tth time step in the sequential setting of Section 4.1, the learner has access to a new context $\mathbf{x}_{M(u)}(u)$ and a reward $r_{M(u)}(u)$ based on the taken action $u(t) = u$. Therefore, OS-SVI uses this data point by updating the uth variational parameters by going one step in the direction of the natural gradient of the ELBO L. For OS-SVI, we calculate the intermediate estimates in Eq. (4.2.29) based on t replicates of the observed data point $(\mathbf{x}(t), u(t), r(t))$. Thereafter, the stochastic gradient update is performed with Eq. (4.2.28), by transforming the natural parameters back to their corresponding parametric form as in Eq. (4.2.30). This updating step is computationally significantly faster than using VB or SVI until convergence. The OS-SVI pseudo-code is given in Fig. 4.4d.

4.3 DECISION-MAKING USING A BANDIT ALGORITHM

After finding some tractable posterior inference schemes, we now tackle the problem, how an agent can make optimal decisions to solve the desired problem

$$\begin{aligned} \underset{u_{[1,T]}}{\text{maximize}} \quad & J\left[u_{[1,T]}\right] = \sum_{t=1}^{T} \mathsf{E}[r(t)] \\ \text{subject to} \quad & r(t) \sim p(r(t) \mid \mathbf{x}(t, u(t)), u(t)). \end{aligned}$$

The complete optimization problem is often still intractable even if the posterior distribution of the reward model can be computed. This is because the agent faces an inherent problem of deciding when to learn more about the model and gather data and when to act greedily w.r.t. the current knowledge of the system at hand. This exploration-exploitation dilemma is discussed under the term multi-armed bandit problem, which dates back to [65]. In our setting with context information, the contextual bandit problem [66], which is an extension to the classic problem, can be used. In the bandit setting, it is often helpful to evaluate the performance of a bandit algorithm in terms of its regret. We introduce the regret for the linear model in Eq. (4.1.1) as

$$R(T) = \sum_{t=1}^{T} r^*(t) - \sum_{t=1}^{T} r(t). \tag{4.3.31}$$

Here, the optimal reward $r^*(t) \sim p(r^*(t) \mid \mathbf{x}(t, u^*(t)), u^*(t))$ is generated for the optimal action $u^*(t) = \arg\max_{u \in \mathcal{U}} \mathbf{x}(t, u)^\top \boldsymbol{\beta}^*(u)$ under the true regression parameters $\boldsymbol{\beta}^*(u)$. This is compared to the reward obtained for an algorithm which chooses the action trajectory $u_{[1,T]}$ and generates the rewards $r(t) \sim p(r(t) \mid \mathbf{x}(t, u(t)), u(t))$. Hence, the regret in Eq. (4.3.31) compares the cumulative reward of the algorithm against the cumulative reward with hindsight. To develop algorithms with a small regret for the linear setting, many strategies have been proposed. Such algorithms include techniques based on forced sampling [67], Thompson sampling [68], and the upper confidence bound (UCB) [56, 69–71]. Here, we use the Bayes upper confidence bound (Bayes-UCB) algorithm as introduced in [72], and we develop a version that fits the given model from Section 4.1.

4.3.1 *The Bayes Upper Confidence Bound Algorithm*

First, let $Q_x(q, p(x))$ denote the quantile function associated to the distribution $p(x)$; i.e.,

$$\mathsf{P}(X \leq Q_x(q, p(x))) = q,$$

with RV $X \sim p(x)$. Given this implicit definition, the Bayes-UCB algorithm of [72] selects in each round the action u, which maximizes the acquisition function

$$A(t, u) \coloneqq Q_{\bar{r}}\left(1 - \frac{1}{\alpha t}, \pi(\bar{r}, t, u)\right), \tag{4.3.32}$$

where $\alpha \in \mathbb{R}_{>0}$ is an exploration parameter for the UCB algorithm. Here,

$$\pi(x, t, u) \coloneqq \frac{\partial}{\partial x} \mathsf{P}(\bar{R}(t, u) \leq x \mid \mathcal{D}(t)) \tag{4.3.33}$$

denotes the posterior density of the mean reward under the model in Eq. (4.1.1); i.e.,

$$\bar{R}(t, u) \coloneqq \mathsf{E}[r(t) \mid u, \mathbf{x}(t)] = \mathbf{x}^\top(t, u)\boldsymbol{\beta}(u), \tag{4.3.34}$$

where the regression coefficients $\boldsymbol{\beta}(u)$ under the conditional probability measure in Eq. (4.3.33) follow the the posterior distribution as $\boldsymbol{\beta}(u) \sim p(\boldsymbol{\beta}(u) \mid \mathcal{D}(t))$. The acquisition function in Eq. (4.3.32) uses as a strategy a one-step look ahead by exploiting an upper confidence bound for the posterior density of the mean reward. This means that a decision is made based on a $1 - \frac{1}{\alpha t}$ sized upper Bayesian credible interval. This strategy is sensible because it is optimistic w.r.t. the agent's reward in the next decision step. Being optimistic is reasonable, as it leads to exploration. However, the credible interval is scaled down with the number of iteration steps and with the posterior distribution getting sharper as more data is available. This leads to a greedier exploitation behavior the longer the algorithm runs. For an analysis of the Bayes-UCB algorithm, see [72], where some convergence results, by bounding the random regret as in Eq. (4.3.31), are shown.

4.3.2 *The Contextual-Based Adaptation Algorithm*

We now calculate the acquisition function in Eq. (4.3.32) for the derived posterior from Section 4.2. Using the mean-field approximation, as in Section 4.2.1, a Gaussian approximate posterior for $\boldsymbol{\beta}(u)$ was derived as

$$p(\boldsymbol{\beta}(u) \mid \mathcal{D}(t)) \approx q(\boldsymbol{\beta}(u)) = \mathcal{N}(\boldsymbol{\beta}(u) \mid \boldsymbol{\mu}_{\boldsymbol{\beta}}(u), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)).$$

Since, Eq. (4.3.34) is a linear equation in $\boldsymbol{\beta}(u) \sim p(\boldsymbol{\beta}(u) \mid \mathcal{D}(t))$, the mean reward $\bar{R}(t, u)$ under the mean-field approximation is also Gaussian distributed, and we can calculate its mean as

$$\begin{aligned}
\mathsf{E}\big[\bar{R}(t, u) \;\big|\; \mathcal{D}(t)\big] &= \mathsf{E}\big[\mathbf{x}^\top(t, u)\boldsymbol{\beta}(u) \;\big|\; \mathcal{D}(t)\big] \\
&= \mathbf{x}^\top(t, u)\, \mathsf{E}[\boldsymbol{\beta}(u) \mid \mathcal{D}(t)] \\
&= \mathbf{x}^\top(t, u)\boldsymbol{\mu}_{\boldsymbol{\beta}}(u)
\end{aligned}$$

and its variance as

$$\begin{aligned}\mathsf{Var}[\bar{R}(t,u)|\mathcal{D}(t)] &= \mathsf{Var}[\mathbf{x}^\top(t,u)\boldsymbol{\beta}(u)|\mathcal{D}(t)] \\ &= \mathbf{x}^\top(t,u)\,\mathsf{Cov}[\boldsymbol{\beta}(u)|\mathcal{D}(t)]\mathbf{x}(t,u) \\ &= \mathbf{x}^\top(t,u)\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)\mathbf{x}(t,u).\end{aligned}$$

This yields the posterior density of the mean reward as

$$\pi(\bar{r},t,u) = \mathcal{N}(\bar{r} \mid \mathbf{x}^\top(t,u)\boldsymbol{\beta}(u), \mathbf{x}^\top(t,u)\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)\mathbf{x}(t,u)).$$

Using this, the acquisition function in Eq. (4.3.32) calculates to

$$A(t,u) = \mathbf{x}^\top(t,u)\boldsymbol{\mu}_{\boldsymbol{\beta}}(u) + Q_{\bar{r}}\left(1-\frac{1}{\alpha t}, \mathcal{N}(\bar{r} \mid 0,1)\right)\sqrt{\mathbf{x}^\top(t,u)\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)\mathbf{x}(t,u)},$$

where we can compute the quantile function explicitly as

$$Q_{\bar{r}}\left(1-\frac{1}{\alpha t}, \mathcal{N}(\bar{r} \mid 0,1)\right) = \sqrt{2}\mathrm{erf}^{-1}\left(1-\frac{2}{\alpha t}\right),$$

with the inverse error function $\mathrm{erf}^{-1}(\cdot)$. Finally, we use the policy

$$u(t) = \arg\max_{u\in\mathcal{U}} A(t,u)$$

as a contextual bandit algorithm. Therefore, the algorithm we call CBA is given as

$$u(t) = \arg\max_{u\in\mathcal{U}} \mathbf{x}^\top(t,u)\boldsymbol{\mu}_{\boldsymbol{\beta}}(u) + \sqrt{2}\mathrm{erf}^{-1}\left(1-\frac{2}{\alpha t}\right)\sqrt{\mathbf{x}^\top(t,u)\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(u)\mathbf{x}(t,u)}. \quad (4.3.35)$$

A pseudo-code for this bandit strategy is given in Fig. 4.4a, and the full CBA algorithm is summarized in Fig. 4.4.

4.4 EVALUATION UNDER THE MODEL ASSUMPTION

We first create data based on the model assumption for the numerical evaluation of CBA exploiting the TPBN prior. For this, we evaluate two simulation scenarios. Here, we use a problem with $n = 20$ context dimensions, $K = 20$ actions, and a decision horizon of $T = 1000$. For both scenarios, we use a linear ground truth model for the reward generation as

$$r(t) \sim \mathcal{N}(r(t) \mid \mathbf{x}^\top(t,u)\boldsymbol{\beta}^*(u), \sigma^{2^*}(u)),$$

with true parameters $\boldsymbol{\beta}^*(u)$ and $\sigma^{2^*}(u)$, which are unknown to the bandit algorithm.

SCENARIO 1: SPARSE REGRESSION. We consider a scenario with sparse regression coefficients, where only five regression coefficients per action are not equal to zero; i.e.,

$$\sum_{j=1}^{n} \mathbb{1}(\beta_j^*(u) \neq 0) = 5, \quad \forall u \in \mathcal{U}.$$

SCENARIO 2: DENSE REGRESSION. We consider a dense scenario for the regression coefficients; i.e.,

$$\beta_j^*(u) \neq 0, \quad \forall (j,u) \in \{1,\dots,n\} \times \mathcal{U}.$$

4.4.1 *Evaluation of the Accuracy*

We compare the developed CBA algorithm with the different posterior inference schemes VB, SVI, and OS-SVI against two baseline algorithms: LinUCB [56] and contextual Gaussian process upper confidence bound (CGP-UCB) [71]. For the CGP-UCB algorithm, we use independent linear kernels for every action, which fits the linear ground truth model assumption. Figure 4.5 shows the evaluation of the regret Eq. (4.3.31), averaged over multiple Monte Carlo runs, where Fig. 4.5a considers the sparse setting and Fig. 4.5b considers the dense setting. For the sparse setting in Fig. 4.5a, which is usually omnipresent in high-dimensional problems, CBA with VB yields the smallest regret. We observe in Fig. 4.5b that in the dense setting, CGP-UCB achieves the best performance, which is closely followed by CBA with VB. Note that it is expected for CGP-UCB to perform well, as GP regression with a linear kernel corresponds to a dense Bayesian regression with marginalized regression coefficients [73], and therefore matches the model under which the dense data has been created.

Algorithm 4.4: Main-routine CBA

input : T: horizon
α: exploration parameter
$\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$: initial parameters

Initialize data set $\mathcal{D}(0) = \{\}$
for $t = 1$ ***to*** T **do**
  observe contexts $\mathbf{x}(t)$
  play action $u(t)$ using Eq. (4.3.35)
  observe reward $r(t)$
  append data $\mathcal{D}(t) = \mathcal{D}(t-1) \cup \{\mathbf{x}(t), u(t), r(t)\}$
  update estimates for $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$ using a sub-routine with $\mathcal{D}(t)$
end

(a) The pseudo-code for the decision-making algorithm.

Algorithm 4.4: Sub-routine VB for CBA

input : $\mathcal{D}(t)$: data set
a_0, b_0, c_0, d_0: hyper-parameters
output : $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$: updated parameters

Initialize variational moments in Eq. (4.2.10)
while *ELBO in Eq.* (4.2.26) *not converged* **do**
  Update variational parameters, Eqs. (4.2.4) to (4.2.9)
  Update the variational moments with Eq. (4.2.10)
end
return $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$

(b) The pseudo-code for posterior inference with VB.

FIGURE 4.4: Pseudo code for proposed CBA algorithm (cont. next page).

Algorithm 4.4: Sub-routine SVI for CBA

input : $\mathcal{D}(t)$: data set
a_0, b_0, c_0, d_0: hyper-parameters
γ_l: step size schedule
output : $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$: updated parameters

Initialize natural parameters $\boldsymbol{\eta}_{\boldsymbol{\beta}}$, $\boldsymbol{\eta}_{\sigma^{-2}}$, $\boldsymbol{\eta}_{\tau,j}$, $\boldsymbol{\eta}_{\lambda,j}$, $\boldsymbol{\eta}_{\phi}$ and $\boldsymbol{\eta}_{\omega}$
Initialize iteration counter $l = 1$
while *ELBO in Eq.* (4.2.26) *not converged* **do**
  Draw random sample $(u, \mathbf{x}_m(u), r_m(u))$ from $\mathcal{D}(t)$
  Calculate intermediate parameters with Eq. (4.2.29)
  Do gradient update with Eq. (4.2.28) and step size γ_l
  Update variational parametrization with Eq. (4.2.30)
  Update moments with Eq. (4.2.10)
  Update iteration counter $l = l + 1$
end
return $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$

(c) The pseudo-code for posterior inference with SVI.

Algorithm 4.4: Sub-routine OS-SVI for CBA

input : $\mathcal{D}(t)$: data set
a_0, b_0, c_0, d_0: hyper-parameters
γ: step size
output : $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$: updated parameters

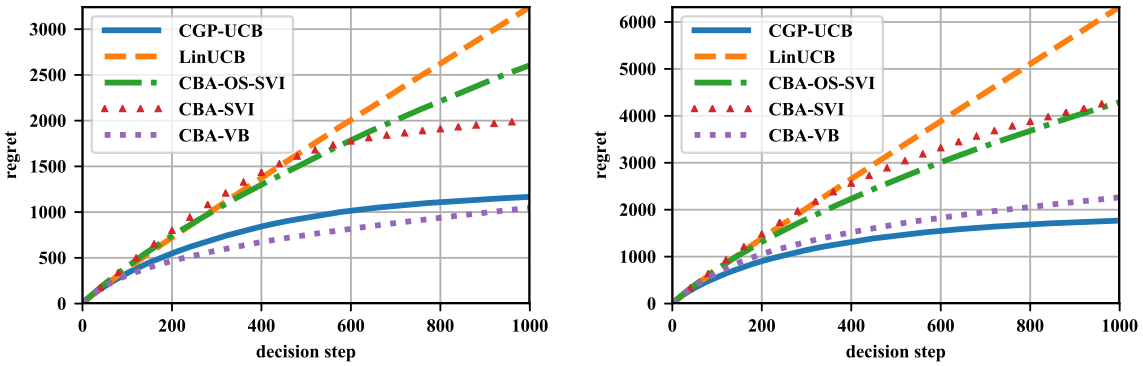
Extract last data point $(u, \mathbf{x}_m(u), r_m(u))$ from $\mathcal{D}(t)$ Calculate intermediate parameters with Eq. (4.2.29) and $M(u) = t$
Do gradient update with Eq. (4.2.28) and step size γ
Update variational parametrization with Eq. (4.2.30)
return $\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}$

(d) The pseudo-code for posterior inference with OS-SVI.

FIGURE 4.4: Pseudo code for proposed CBA algorithm (cont.). In (a) the decision-making algorithm is shown. The different possible posterior inference schemes are given in (b) (VB), (c) (SVI) and (d) (OS-SVI).

4.4.2 *Evaluation of the Computation Run-Time*

In Table 4.1 we show the run-times of the algorithms, where we observe that the run-times for CBA with VB and SVI, and the CGP-UCB baseline can be impractically high in time-critical applications. Further, this run-time performance scales with certain parameters as depicted in Fig. 4.6. This can be seen, e.g., for a scaling of the context dimension n, since the computational bottleneck of both posterior inference with VB and posterior inference with SVI are multiple matrix inversions of size $n \times n$, see Fig. 4.6a. Additionally, we evaluate the effect of the decision horizon T on the run-time. Here, Fig. 4.6b shows the scaling using the identical setup as in Table 4.1. It can be observed that CGP-UCB scales poorly with T, as the kernel matrix of size $M(u) \times M(u)$ is inverted at every time step. Since, for many scenarios, decision-making has to be made in the order of a few hundred milliseconds, neither CBA with VB nor CGP-UCB can be computed under time-critical conditions. However, we observe that the OS-SVI variant fulfills timing restrictions and



(a) Scenario 1: Sparse regression.

(b) Scenario 2: Dense regression.

FIGURE 4.5: Results under the model assumption. Depicted is the average regret for the proposed CBA algorithm with VB, SVI and OS-SVI posterior inference compared to the baseline algorithms CGP-UCB and LinUCB, for (a) a sparse regression scenario and for (b) a dense regression scenario.

Algorithm	Sparse Setting	Dense Setting
CGP-UCB	638.68 s	643.44 s
LinUCB	31.24 s	30.70 s
CBA-OS-SVI	91.40 s	89.56 s
CBA-SVI	3784.00 s	4081.74 s
CBA-VB	1434.11 s	1760.83 s

TABLE 4.1: Run-times for $N = 100$ simulations of the CBA algorithms compared to the baseline algorithms CGP-UCB and LinUCB. Simulations executed on an Intel® Xeon® E5-2680 v3 @2.5GHz machine.

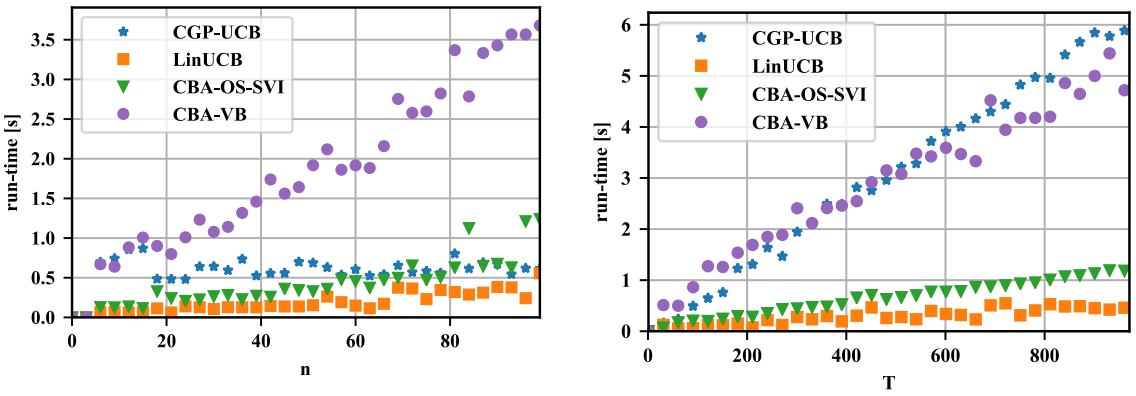
obtains a much smaller average regret than the fast LinUCB baseline algorithm. The reason for the increased speed-up of these two methods is that they both have to invert an $n \times n$ matrix only once after a decision.

4.5 APPLICATION SCENARIO: VIDEO STREAMING

After presenting a decision-making algorithm, which can perform its necessary inference in a small amount of time, we look at a scenario from video streaming. Here, we describe how a problem in this context, namely ABR streaming, can be modeled as a decision-making problem, which fits our derived CBA algorithm. In this setting, we look at a video streaming scenario within NDN, where consumers or clients issue interests, which are forwarded to content producers, i.e., origin servers, via caching-enabled network routers.

4.5.1 *Adaptive Bitrate Video Streaming*

The scenario of video streaming is an excellent example of the problem of making decisions under uncertainty. Modern applications often rely for video streaming on ABR streaming.



(a) Run-time vs. context dimensions n for a sparse linear model. (CBA with SVI not shown for clarity.)

(b) Run-time vs. decision horizon T for a sparse linear model. (CBA with SVI not shown for clarity.)

FIGURE 4.6: Run-time comparisons for sparse linear model with $K = 20$ actions. In (a) the context dimension n is varied with fixed decision horizon of $T = 100$. In (b) the decision horizon T is varied with fixed number of context dimensions $n = 20$.

Here, the content provider offers multiple qualities of the same video content to the client. The client can then decide which quality to pick based on its own client-side logic. In the context of NDN these interests are then eventually answered with data provided by the producer or an intermediary router cache.

To decide which quality to pick, each video is divided into consecutive segments, representing some fixed number of seconds of content, we denote as l. These segment lengths are, in practice, often chosen to be two to ten seconds [74], with several distinct quality levels to choose from, such as 720p and 1080p. Here, the segments are encoded at multiple bitrates, directly mapping to the perceived average segment quality. Hence, from a decision-making point of view, the set of actions for the agent to choose from are the set of all available video qualities. Therefore, we denote the set of video qualities as

$$\mathcal{U} = \{u^{(i)} \mid i = 1, \dots, K\},$$

where we let $u^{(i)} > u^{(j)}$ for $\forall i > j$; i.e., a higher index indicates a higher bitrate and therefore better quality. Additionally, we denote by $s(t, u)$ the tth segment encoded in the uth quality. Out of the set of possible quality levels $\mathcal{U}$ a client can then select for each segment, one of the possible quality levels.

In NDN a consumer will first issue an interest for a media presentation description (MPD), a extensible markup language (XML)-like file with information on the available video segments and quality levels of the video, during session initialization. After obtaining the MPD, the client may begin to request each segment according to its adaptation algorithm. Here, each segment $s(t, u)$ is given a name in the MPD, e.g., of the form `/video ID/quality level/segment number`. Afterwards, the client issues an interest for each data packet when requesting a particular segment.

After the client decides for a video quality $u \in \mathcal{U}$, a video segment is received and placed into a playback buffer containing downloaded unplayed video segments. A simple queueing relation can describe this playback buffer. For this, we denote by $b(t)$ the number of

seconds in the buffer at the tth decision epoch; i.e., at the time point when segment t is received. The playback buffer size is denoted by B and is the maximum allowed number of seconds of video in the buffer. This playback buffer size B is usually chosen between ten and thirty seconds [74]. By convention we let $b(0) = 0$. The queueing recursion of the buffer filling can then be written as

$$b(t) = [b(t-1) + l - \xi(t,u)]_0^B \,,$$

where $[\cdot]_0^B \coloneqq \min(\max(\cdot, 0), B)$, denotes the truncation operation and $\xi(t,u)$ is the fetch time for the tth segment with quality u.

Given this recursion we can ascribe a stalling event for the tth segment when $\xi(t,u) > b(t-1)$. Note that in general, uncertainty exists over the segment fetch time $\xi(t,u)$ and the recursion above holds only if $b(t-1) + l < B$; i.e., the client is blocked from fetching new segments if the playback buffer is full. If this occurs, the client idles for exactly l seconds before resuming segment fetching. Since NDN data packets are of small and fixed size, higher-quality video segments will require more data packets to encode. We note that we here do not permit the client to drop frames, so all data packets belonging to some segment $s(t,u)$ must be in the playback buffer to watch that segment.

The most prevalent quality adaptation algorithms take throughput estimates [75] or the current buffer filling $b(t)$ [76], or combinations and functions thereof to decide on the quality of the next segment $s(t+1,u)$. However, overall the goal of the decision-maker is to find the segment quality which maximizes a QoE metric, such as the average video bitrate, or compound metrics taking the bitrate, bitrate variations, and stalling events into account. Hence, for optimizing QoE, the presented application can directly be mapped to the CBA algorithm presented earlier, as we now show.

4.5.2 *Using the Contextual-Based Adaptation Algorithm for Video Streaming*

We model ABR streaming as a contextual bandit problem, where we use the presented CBA algorithm.

Here, we use, as discussed before, as the action set, the set of available bitrates such that action $u(t) = u$ represents the decision to request the quality u for the tth segment; i.e., to request the segment $s(t,u)$.

Furthermore, we let $\mathbf{x}(t,u)$ represent the network context vector corresponding to an available action u at segment t. At each t, therefore, there will be K unique context vectors available. There are no constraints on the contents of the context vectors, allowing CBA to learn with any information available in the networking environment. Moreover, each context feature may be global or action-specific, such as the current buffer filling percentage or the last fifty packet RTTs at bitrate u, respectively. In this network-assisted video streaming scenario, it is hence sensible to use a high-dimensional context vector for which it is natural to assume sparse regression coefficients $\boldsymbol{\beta}(u)$ in the CBA algorithm. This automatically leads to feature selection since we used for the posterior inference in CBA the TPBN shrinkage prior.

In order to use CBA in this video streaming setting, we formulate a real-valued segment-based QoE function to represent the reward $r(t) \sim p(r(t) \mid \mathbf{x}(t, u(t)), u(t))$ obtained by performing $u(t)$. Here, the calculated QoE metric is the *feedback* used by CBA to optimize the quality adaptation strategy. As QoE scores for a video segment may vary among users, we resort in this work to an objective QoE metric similar to ,[77] which is derived from the following set of factors: (i) video quality, (ii) decline in video quality and (iii) rebuffering time.

First, for the video quality, we consider the bitrate of the segment; i.e., $u(t) \in \mathcal{U}$. Second, for the decline in video quality we measure if the current segment is at a lower bitrate than the previous one, $[u(t-1) - u(t)]_0$ for two back to back segments, with $[\cdot]_0 \coloneqq \max(\cdot, 0)$. The rationale behind using the decline in quality, in contrast to the related work that counts quality variations, is that we do not want to penalize CBA if the player strives for higher qualities without the risk of rebuffering. Finally, we consider the amount of time spent with an empty buffer after choosing $u(t)$ to evaluate the rebuffering time, where we denote the amount of time spent rebuffering after choosing $u(t)$ as $G(u(t))$.

The importance of these three components may vary based on the specific user or context, so, similar to [77], we define the QoE of a segment as a weighted sum of the above factors. Hence, the reward or QoE is calculated as

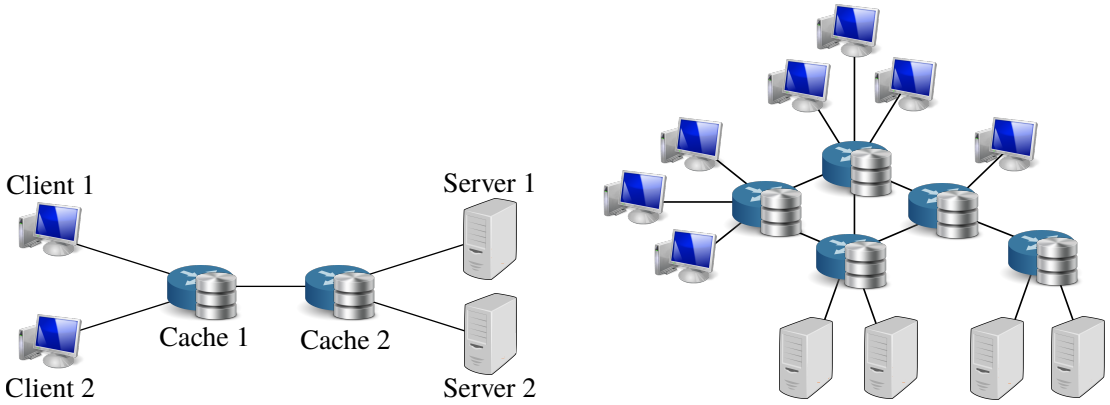
$$r(t) \equiv \mathrm{QoE}(t) = w_1 u(t) - w_2 [u(t-1) - u(t)]_0 - w_3 G(u(t)), \tag{4.5.36}$$

where w_1, w_2, and w_3 are non-negative weights corresponding to the importance of the video quality, decline in quality, and rebuffer time, respectively. For a comparison of several instantiations of these weights, see [77]. Note that this particular reward function is sensible, though arbitrary. For other purposes, other QoE metrics can be specified for CBA as long as these comprise a scalar function to produce the reward metric. For example, it might be of interest to use other low-level metrics such as fetching time or to use subjective quality evaluation tests for different users to map the weights to a QoE metric, e.g., via the mean opinion score (MOS) [78].

4.5.3 *Evaluation of the Quality Adaptation in Named Data Networking*

Next, we evaluate the performance of CBA for video streaming within NDN and compare it with throughput-based adaptation (TBA) and buffer-based adaptation (BBA) peers by emulating two NDN topologies as depicted in Fig. 4.7. For the evaluation, we consider (i) the *doubles* topology, shown in Fig. 4.7a and (ii) the *full* topology, shown in Fig. 4.7b. These topologies are built using an extension of the containernet project[3] which allows the execution of Docker[4] hosts as nodes in the Mininet[5] emulator.

3 https://github.com/containernet/containernet
4 https://www.docker.com/
5 http://mininet.org/



(a) Emulation testbed for the *doubles* topology. The client link capacities follow random bandwidth traces.

(b) Emulation testbed for the *full* topology. The client links have a fixed capacity of 20 Mbps.

FIGURE 4.7: Emulation testbeds for video streaming in an NDN. The *doubles* topology in (a) and the *full* topology in (b) both have server links with a capacity of 20 Mbps and the internal cache links have a capacity of 1000 Mbps. In both topologies, the caches can store up to 1500 data chunks.

VIDEO STREAMING SETUP. The NDN clients use a DASH player implemented with *libdash*, based on code from [52]. We utilize the interest control protocol (ICP), where we set the parameters to $\gamma_{\text{ICP}} = 2$, $\beta_{\text{ICP}} = 0.5$ and `initialWindow` $= 300$. We note that traffic burstiness can vary significantly depending on the ICP parameters used.

The clients begin playback simultaneously, where they stream the first two hundred seconds of a video encoded in two-second *H.264-AVC* segments offered at the $K = 5$ quality bitrates $\mathcal{U} = \{1\,\text{Mbps}, 1.5\,\text{Mbps}, 2.1\,\text{Mbps}, 3\,\text{Mbps}, 3.5\,\text{Mbps}\}$, with a playback buffer size of thirty seconds; i.e., $B = 30\,\text{s}$. All containers run instances of the NDN forwarding daemon (NFD) with the *access* strategy, and *repo-ng* is used to host the video on the servers and caches.

In the following, we compare the performance of CBA with VB and OS-SVI, in addition to the baseline algorithm LinUCB [56]. We also examine the performance of two state-of-the-art BBA and TBA algorithms; i.e., the buffer occupancy-based Lyapunov algorithm (BOLA) [76] and the probe and adapt (PANDA) algorithm [75], respectively. Note that there are many other adaptation algorithms in the literature, some of which use BBA and TBA simultaneously [77, 79–81]. However, BOLA and PANDA were chosen because they are widely used and achieve state-of-the-art performance in standard HTTP environments. Buffer filling percentage and quality-specific segment packet RTTs are provided to the client as context. Furthermore, we added a *numHops* tag to each data packet to track the number of hops from the data origin to the consumer.

For CBA, we track the RTTs and number of hops of the last fifty packets of each segment received by the client following measurements from [82]. If a segment does not contain fifty packets, results from existing packets are resampled. As a result, each CBA algorithm variant is given an $n = 101$ dimensional context vector $\mathbf{x}(t, u)$, containing the buffer fill percentage, packet RTTs, and *numHops* for each of the $K = 5$ available qualities. Additionally, we choose for CBA the following hyper-parameters. We use an exploration of

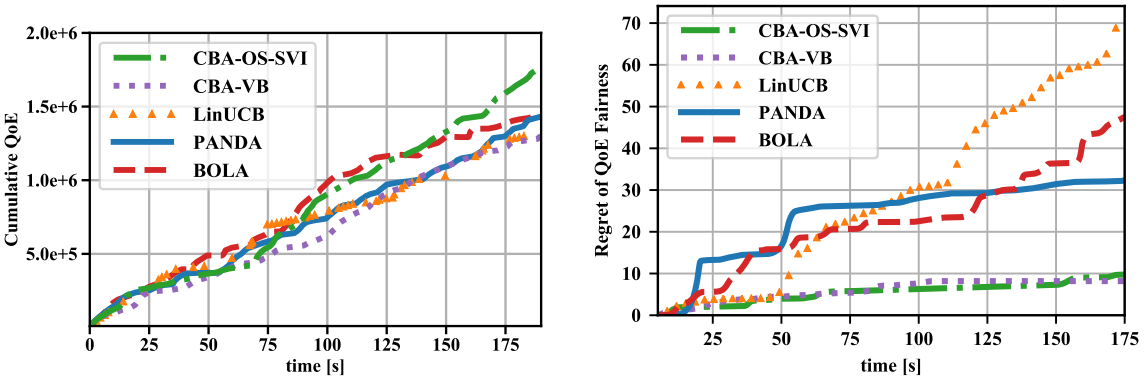
$\alpha = 1$ as it was shown to yield good results under similar models [72]. For the TPBN prior, we use $a_0 = b_0 = 1/2$ to obtain the horseshoe shrinkage prior. We let $c_0 = d_0 = 10^{-6}$ such that a vague prior is obtained.

4.5.4 *Evaluation of the Doubles Topology*

SETUP. For the *doubles* topology, we modulate the capacity of the bottleneck link using a truncated normal distribution, where the link capacity is drawn with a mean of $7\,\mathrm{Mbps}$. The capacity stays unchanged for a random period length, which is drawn from another truncated normal distribution with a mean of $5\,\mathrm{s}$. For the weights in Eq. (4.5.36) we use $w_1 = 6$, $w_2 = 2$ and $w_3 = 2$, emphasizing the importance of the average quality bitrate without allowing a large amount of rebuffering to take place.

Algorithm	Bitrate [Mbps]	Quality switches [#]	Switch magnitude [Mbps]	Parameter update time [ms]
CBA-OS-SVI	**3.10**	**6**	**0.57**	15
CBA-VB	2.58	**6**	0.65	325
LinUCB	2.24	14	1.07	**6**
BOLA	2.63	36	1.19	–
PANDA	2.51	16	1.00	–

TABLE 4.2: Streaming statistics for client 1 on the *doubles* topology.



(a) Client 1 cumulative QoE over playback on the *doubles* topology.

(b) QoE fairness evaluation on the *doubles* topology.

FIGURE 4.8: Results for different QoE-based metrics on the *doubles* topology. In (a) we evaluate the cumulative reward optimization criterion, which in the case of the CBA optimizes for cumulative QoE. In (b) we compare the regret of QoE fairness for CBA and different baseline methods.

RESULTS. The results are depicted in Table 4.2 and Fig. 4.8. Examining Table 4.2, we see that the CBA with OS-SVI yields a significantly higher average bitrate. This is expected based on the QoE definition Eq. (4.5.36), but we might expect CBA with VB to pick high

bitrates as well. However, we observe that the parameter update time for the VB variant is twenty times greater than that of the OS-SVI variant; this puts a delay of one-sixth of each segment length on average between receiving one segment and requesting another. Looking at CBA with VB in Fig. 4.8a we see that it accumulates a much larger rebuffer time than other methods. Hence, CBA with VB is forced to request lower bitrates to cope with the extra rebuffer time incurred by updating its parameters.

In addition, note that LinUCB fails to select high bitrates despite having a very short parameter update time, implying that LinUCB is not adequately fitting the context to the QoE and is instead accumulating a large amount of regret. This is corroborated by its cumulative QoE depicted in Fig. 4.8a, which performs nearly as poorly as CBA with VB. By inducing sparsity on the priors and using just one sample, CBA with OS-SVI successfully extracts the most salient features quickly enough to obtain the highest cumulative QoE of all algorithms tested.

Concerning the rebuffering behavior, we observe rebuffering ratios of {4.5%, 8.4%, 11.4%, 17.6%, 32.9%} for LinUCB, BOLA, PANDA, CBA with OS-SVI and CBA with VB, respectively. We trace some of the rebuffering events to the ICP congestion control in NDN. Note that tuning the impact of rebuffering on the adaptation decision is not a trivial task [52]. Fortunately, this is not hardwired in CBA but instead given through Eq. (4.5.36). Hence, in contrast to state-of-the-art adaptation algorithms, CBA could learn to filter the contextual information that is most important for rebuffering by tweaking the QoE metric used.

Interestingly, the CBA approaches shown in Table 4.2 also result in the lowest number of quality switches, though our QoE metric does not severely penalize quality variation. We see that the magnitude of their quality switches is also nearly half that of the other algorithms.

An important consideration when choosing a quality adaptation algorithm is fairness among clients while simultaneously streaming over common links. While this is taken care of in DASH by the underlying transmission control protocol (TCP) congestion control, we empirically show here how the on-off segment request behavior, when paired with the considered quality adaptation algorithms, impacts the QoE fairness in NDN. This is fundamentally different from considering bandwidth sharing fairness in NDN, e.g., in [52]. Here we are interested in QoE fairness since the QoE metric and not the bandwidth share is the primary driver of the quality adaptation algorithm.

For the evaluation, we use as a metric the regret of QoE fairness between both clients. The regret is calculated w.r.t. the optimal fairness. Here, we choose following the discussion in [83] the binary entropy as a fairness measure. For the fairness f_{qoe} of the algorithm, we calculate the relative QoE of the two clients as

$$f_{\mathrm{qoe}}(t) = H_B\left(\frac{\mathrm{QoE}_{\text{client 1}}(t)}{\mathrm{QoE}_{\text{client 1}}(t) + \mathrm{QoE}_{\text{client 2}}(t)}\right),$$

where $H_B(\cdot)$ denotes the binary entropy function and the QoE is given by Eq. (4.5.36). Further we consider the optimal fairness between two clients as

$$f^*_{\mathrm{qoe}}(t) = H_B\left(\frac{\mathrm{QoE}^*_{\text{client1}}(t)}{\mathrm{QoE}^*_{\text{client1}}(t) + \mathrm{QoE}^*_{\text{client2}}(t)}\right) = 1,$$

Algorithm	Bitrate [Mbps]	Quality switches [#]	Switch magnitude [Mbps]	Parameter update time [ms]
High quality setting				
CBA-OS-SVI	1.55	**5**	0.82	53
CBA-VB	1.52	15	1.16	1254
LinUCB	1.27	17	1.01	**11**
BOLA	**1.96**	8	0.63	–
PANDA	1.15	18	**0.56**	–
Low rebuffering setting				
CBA-OS-SVI	1.55	**6**	0.93	55
CBA-VB	1.68	12	1.08	1362
LinUCB	1.43	22	1.04	**16**
BOLA	**1.92**	12	0.71	–
PANDA	1.13	17	**0.70**	–

TABLE 4.3: Streaming statistics for client 1 on the *full* topology.

where the optimal fairness is achieved for the ratio $1/2$; i.e., $H_B\left(1/2\right) = 1$. Finally, the regret of QoE fairness R_{qoe} is then defined as a cumulative metric similar to Eq. (4.3.31) as

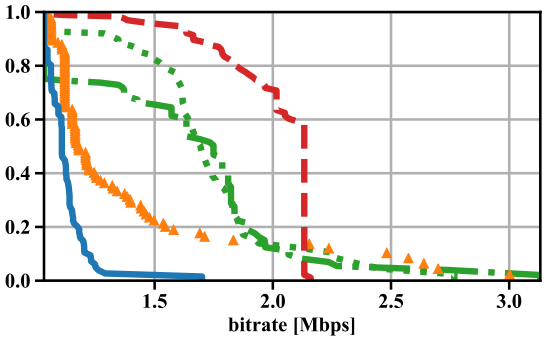
$$R_{\mathrm{qoe}}(T) = \sum_{t=1}^{T} f^{*}_{\mathrm{qoe}}(t) - \sum_{t=1}^{T} f_{\mathrm{qoe}}(t).$$

Figure 4.8b shows the regret of QoE fairness between both clients, where a larger regret indicates a greater difference in QoE between both clients up to a particular segment. Here, we observe that the CBA algorithms attain a significantly lower QoE fairness regret than other techniques.

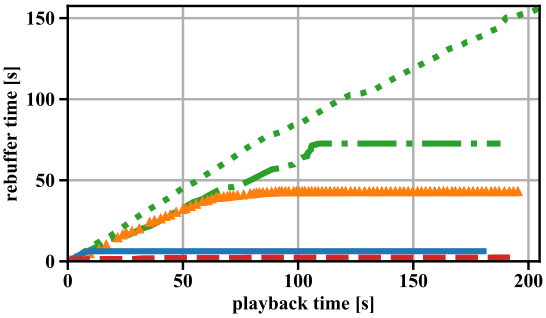
4.5.5 *Evaluation of the Full Topology*

SETUP. To evaluate the capacity of CBA to adapt to different reward functions in complex environments, we compare performance with the *full* topology on two sets of weights in Eq. (4.5.36). For the evaluation of the *full* topology, we consider (i) a *high quality setting* with weights $w_1 = 6$, $w_2 = 2$ and $w_3 = 2$, identical to those used in the evaluation on the *doubles* topology and (ii) a *low rebuffering setting* with weights $w_1 = 1$, $w_2 = 1$ and $w_3 = 3$, placing greater importance on continuous playback at the expense of video quality.

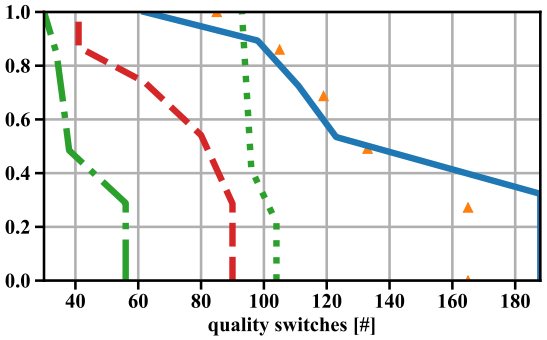
We evaluate each algorithm with each weighting scheme for thirty episodes, where one episode corresponds to streaming two hundred seconds of the video. All clients use the same adaptation algorithm and weighting scheme within an episode, and the bandit algorithms are initialized for each episode with no prior context information.



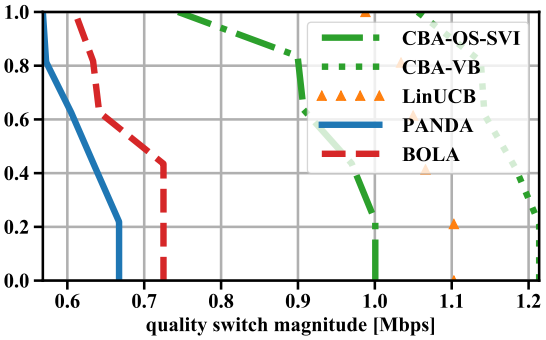
(a) CCDF of average bitrate chosen per episode.



(b) Average cumulative rebuffer time during playback.



(c) CCDF of the number of quality switches per episode.

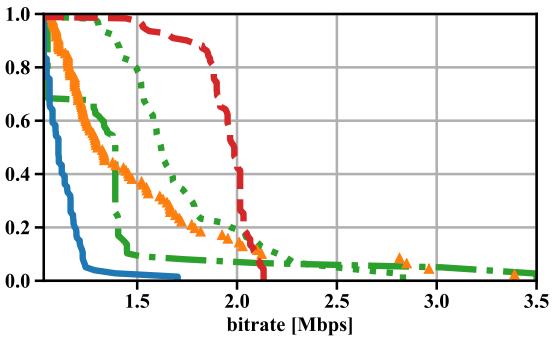


(d) CCDF of the average magnitude of quality switches per episode.

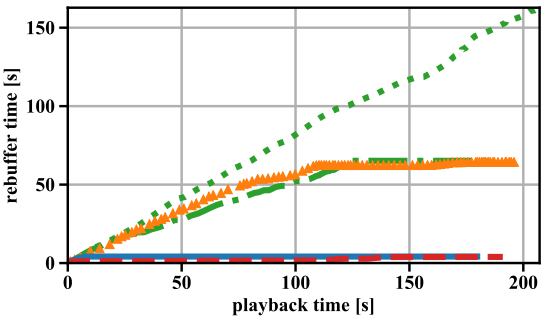
FIGURE 4.9: Results for the *high-quality setting* on the *full* topology.

RESULTS. The results are depicted in Table 4.3 and Figs. 4.9 and 4.10. Inspecting Table 4.3, we observe that the performance statistics among algorithms, even with different weighting schemes, are much closer than for the *doubles* topology. We attribute this to using a more complicated topology in which many more clients are sharing network resources, resulting in fewer and less predictable resources for each client. Furthermore, the average bitrate for the bandit algorithms does not change significantly across weighting schemes and either stays the same or increases when using the *low rebuffering setting* for the weights. This may seem contradictory, but, analyzing Figs. 4.9a and 4.10a, we note that CBA with OS-SVI tended to choose much lower bitrates for the *low rebuffering setting* and therefore accruing less rebuffer time in Fig. 4.10b, than for the *high quality setting* in Fig. 4.9b, indicating that CBA with OS-SVI successfully adapted to either weighting scheme within the playback window. Similar to the *doubles* topology, LinUCB failed to map the context to either weighting scheme, selecting higher bitrates and rebuffering longer for the *low rebuffering setting*. Note that, for either CBA with OS-SVI or LinUCB, the cumulative rebuffer time in Figs. 4.9b and 4.10b tapers off roughly halfway through the video, as either algorithm learns to request more appropriate bitrates.

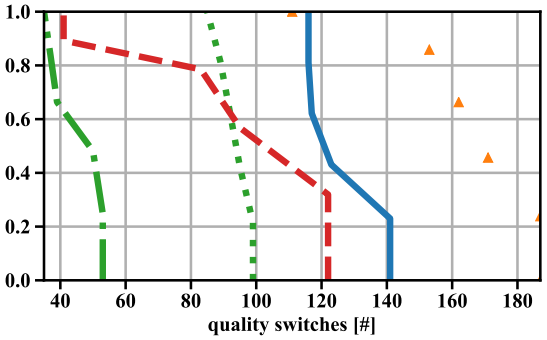
Interestingly, CBA with VB also fails to adapt to either weighting scheme, performing nearly identically in either case. This is a byproduct of the excessive parameter update time for CBA with VB in Table 4.3, which stems from the unpredictable nature of a larger network and the computational strain of performing up to seven VB parameter updates simultaneously on the test machine. CBA with VB is therefore spending over half of the length of each segment



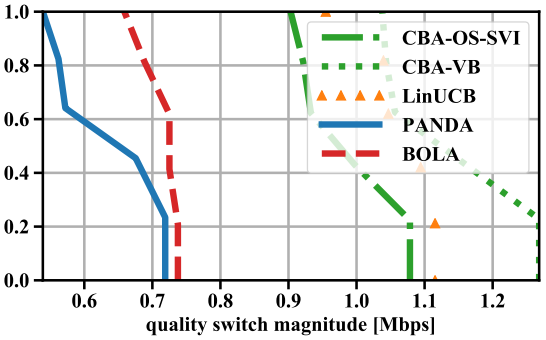
(a) CCDF of average bitrate chosen per episode.



(b) Average cumulative rebuffer time during playback.



(c) CCDF of the number of quality switches per episode.



(d) CCDF of the average magnitude of quality switches per episode.

FIGURE 4.10: Results for the *low rebuffering setting* on the *full* topology.

deciding on which segment to request next, causing long rebuffering times in Figs. 4.9b and 4.10b, culminating in very low QoE scores regardless of the weighting scheme used. This obfuscates the underlying QoE function, preventing CBA with VB from differentiating between the weights in either case within the time allotted. In a real-world scenario, where each client is an independent machine, we expect that CBA with VB and CBA with OS-SVI and LinUCB, to a lesser extent, would have parameter update times comparable to those on the *doubles* topology, resulting in better performance. However, we leave this evaluation in such an environment for future work.

Again, we see in Table 4.2 that CBA with OS-SVI switches qualities least frequently despite neither weighting scheme explicitly penalizing quality variation. Furthermore, according to Figs. 4.9c and 4.9d and Figs. 4.10c and 4.10d, CBA with OS-SVI and CBA with VB are both stable in the number of quality switches and the quality switch magnitude across episodes, even under different weighting schemes, as opposed to the other algorithms tested.

4.6 SUMMARY

In this chapter, we discussed a sparse Bayesian model for adaptive video streaming. Further, we developed a contextual bandit algorithm in this setting denoted contextual-based adaptation (CBA). In contrast to state-of-the-art adaptation algorithms, we take high-dimensional video streaming context information and enforce sparsity to shrink the impact

of unimportant features. In this setting, streaming context information includes client-measured variables, such as throughput and buffer filling, as well as network assistance information. Since sparse Bayesian estimation is computationally expensive, we developed a fast new inference scheme to support online video quality adaptation. For this, we gave a stochastic variational inference algorithm with a computationally easy one-step gradient scheme, which enables fast inference in the sequential decision-making setting.

Furthermore, we evaluated the algorithm extensively under the model assumption. For the application scenario of video streaming, the provided algorithm is naturally applicable to different adaptive video streaming settings, such as DASH over NDN. Finally, we provided NDN emulation results showing that CBA yields a higher QoE objective value and better QoE fairness score between simultaneous streaming sessions compared to known throughput- and buffer-based video quality adaptation algorithms.

5

EXPLOITING BAYESIAN CORRELATION MODELS IN DECISION-MAKING

Up until this point, we did model in Chapter 4 only the reward function explicitly. However, we are frequently interested in a description of the state or action dynamics of a decision-making agent. By using a model for these types of dynamics, the performance of the developed algorithms can be substantially increased. For this, we note that many decision-making problems naturally exhibit pronounced structures inherited from the characteristics of the underlying environment. In an MDP model, for example, two distinct states can have inherently related semantics or encode resembling physical state configurations. This often implies locally correlated transition dynamics among the states. To complete a particular task in such environments, the operating agent must execute a series of temporally and spatially correlated actions. Though a variety of approaches exist to capture these correlations in continuous state-action domains, a principled solution for discrete environments is missing. Therefore, we present in this chapter a Bayesian learning framework based on Pólya-Gamma augmentation that enables analogous reasoning in such cases. We demonstrate the framework on several common decision-making-related problems, such as imitation learning, subgoal extraction, system identification, and BRL. By explicitly modeling the underlying correlation structures of these problems, the proposed approach yields superior predictive performance than correlation-agnostic models, even when trained on data sets that are an order of magnitude smaller in size. This chapter extends and contains parts and material from the published work [2].

Background

Correlations arise naturally in many aspects of decision-making. The reason for this phenomenon is that decision-making problems often exhibit pronounced *structures*, which substantially influence the strategies of an agent. Examples of correlations are even found in

stateless decision-making problems, such as multi-armed bandits, where prominent patterns in the reward mechanisms of different arms can translate into correlated action choices of the operating agent [84, 85]. However, these statistical relationships become more pronounced in the case of contextual bandits, where effective decision-making strategies not only exhibit a temporal correlation but also take into account the state context at each time point, introducing a second source of correlation as discussed in Chapter 4 or in the related work [71].

In more general decision-making models, such as MDPs, the agent can directly affect the state of the environment through its action choices. The effects caused by these actions often share common patterns between different states of the process, e.g., because the states have inherently related semantics or encode similar physical state configurations of the underlying system. Examples of this general principle are omnipresent in all disciplines and range from robotics, where similar actuator outputs result in similar kinematic responses for similar states of the robot's joints, to networking applications, where the servicing of a particular queue affects the surrounding network state (Section 5.4.2.3). The typical consequence is that the structures of the environment are usually reflected in the decisions of the operating agent, who needs to execute a series of temporally and spatially correlated actions to complete a specific task. This is particularly true when two or more agents interact with each other in the same environment and need to coordinate their behavior [86].

Focusing on rational behavior, correlations can manifest themselves even in unstructured domains, though at a higher level of abstraction of the decision-making process. This is because rationality itself implies the existence of an underlying objective optimized by the agent that represents the agent's intentions and incentivizes them to choose one action over another. Typically, these goals persist at least for a short period of time, causing dependencies between consecutive action choices (Section 5.4.1.2).

Therefore, we propose a learning framework that offers a direct way to model such correlations in *finite* decision-making problems; i.e., involving systems with discrete state and action spaces. A key feature of our framework is that it allows capturing correlations at any level of the process; i.e., in the system environment, at the intentional level, or directly at the level of the executed actions. We encode the underlying structure in a hierarchical Bayesian model, for which we derive a tractable VI method based on PG augmentation that allows a fully probabilistic treatment of the learning problem. Results on common benchmark problems and a queueing network simulation demonstrate the advantages of the framework. The accompanying code is publicly available via Git.[1]

Related Work

Modeling correlations in decision-making is a common theme in RL and related fields. GPs offer a flexible tool for this purpose and are widely used in a wide variety of contexts. Moreover, movement primitives [87] provide an effective way to describe temporal relationships in control problems. However, the natural problem domain of both is a continuous state-action environment, which is not the focus of this chapter.

1 https://github.com/bastianalt/correlation_priors_for_rl

Inferring correlation structure from count data has been discussed extensively in topic modeling [88, 89] and factor analysis [90]. Recently, a GP classification algorithm with a scalable variational approach based on PG augmentation was proposed [91]. Though these approaches are promising, they do not address the problem-specific modeling aspects of decision-making.

Several customized solutions exist for agents acting in discrete environments that allow modeling specific characteristics of a decision-making problem. A broad class of methods that specifically target temporal correlations rely on hidden Markov models. Many of these approaches operate on the intentional level, modeling the temporal relationships of the different goals followed by the agent [92]. However, there also exist several approaches to capture spatial dependencies between these goals. For a recent overview, see [93] and the references therein. Dependencies on the action level have also been considered in the past, but, like most intentional models, existing approaches primarily focus on the temporal correlations in action sequences (such as probabilistic movement primitives [87]), or they are restricted to the particular case of deterministic policies [94]. A probabilistic framework to capture correlations between discrete action distributions is described in [95].

When it comes to modeling transition dynamics, most existing approaches rely on GP models [96, 97]. In the Texplore method of [98], correlations within the transition dynamics are modeled with the help of a random forest, creating a mixture of decision tree outcomes. Yet, a full Bayesian description in the form of an explicit prior distribution is missing in this approach. For behavior acquisition, prior distributions over transition dynamics are advantageous since they can easily be used in BRL algorithms such as Bayesian exploration exploitation tradeoff in learning (BEETLE) [99] or Bayes-adaptive Monte Carlo planning (BAMCP) [100]. A particular example of a prior distribution over transition probabilities is given in [101] in the form of a Dirichlet mixture. However, the incorporation of prior knowledge expressing a particular correlation structure is difficult in this model. To the best of our knowledge, no principled method exists to explicitly model correlations in the transition dynamics of discrete environments. Also, a universally applicable inference tool for discrete environments, comparable to GPs, has not yet emerged. In this chapter, we fill this gap by providing a flexible inference framework for such cases.

5.1 A GENERATIVE MODEL FOR CORRELATED COUNT DATA IN MARKOV DECISION PROCESSES

Similar to Chapter 4, we consider a discrete-time setting $t \in \mathbb{N}$, where an agent can make decision $u(t) \in \mathcal{U}$ based on a finite action set $\mathcal{U} = \{u^{(i)} \mid i = 1, \dots, m\}$, with m distinct actions. However, in this chapter we specifically focus on the setting, where the state of the agent $x(t) \in \mathcal{X}$, takes values from a finite state space $\mathcal{X} = \{x^{(i)} \mid i = 1, \dots, n\}$ of size n. Further, compared to Chapter 4 we consider a dynamical model for the states in the form of a Markovian transition kernel. Here, the agent can select an action $u(t) \in \mathcal{U}$ for each time point t. When the action $u(t)$ is executed by the agent, it lets it switch its current state $x(t) \in \mathcal{X}$ stochastically to a new state $x(t+1) \in \mathcal{X}$ according to

$$x(t+1) \sim p(x(t+1) \mid x(t), u(t)).$$

For optimal decision-making, we hence consider an MDP with a reward function $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. Given this reward function, the agent tries to solve the OC problem

$$\begin{aligned} \underset{u_{[0,\infty]}}{\text{maximize}} \quad & J\left[u_{[0,\infty]}\right] = \sum_{t=0}^{\infty} \gamma^t \, \mathsf{E}[R(x(t), u(t))] \\ \text{subject to} \quad & x(t+1) \sim p(x(t+1) \mid x(t), u(t)), \end{aligned}$$

over the action trajectory $u_{[0,\infty]} \coloneqq \{u(t) \mid t \in \mathbb{N}_{\geq 0}\}$, where we consider an infinite horizon setting with discount factor $\gamma \in (0, 1) \subset \mathbb{R}$. Additionally, we denote a deterministic policy $\mu : \mathcal{X} \to \mathcal{U}$ as a function, which maps a particular state $x \in \mathcal{X}$ to an action $u \in \mathcal{U}$; i.e., $u = \mu(x)$. For a stochastic policy we use $\mu : \mathcal{U} \times \mathcal{X} \to [0, 1]$, which maps a given state $x \in \mathcal{X}$ to the probability distribution over actions $u \in \mathcal{U}$; i.e., $\mu(u, x) \equiv p(u \mid x)$.

THE LIKELIHOOD. One overarching theme in these types of decision-making problems with discrete state and action spaces is that the data $\mathcal{D}$ consists of integer-valued quantities. Hence, we can model these quantities as draws from multinomial distributions as

$$\mathbf{s}_c \sim \mathrm{Mult}(\mathbf{s}_c \mid N_c, \mathbf{p}_c),$$

with number of trials $N_c \in \mathbb{N}$, event probabilities $\mathbf{p}_c = [p_{c1}, \ldots, p_{cK}]^\top \in \Delta^K$ and K denotes the number of categories. Here, $c \in \mathcal{C}$ indexes some finite covariate space $\mathcal{C} = \{c^{(i)} \mid i = 1, \ldots, C\}$ with cardinality C. We choose an abstract formulation for this covariate space $\mathcal{C}$ as the count vectors $\mathbf{s}_c \in \{0, \ldots, N_c\}^K$, with $\|\mathbf{s}_c\|_1 = N_c$ can either represent actual count data observed during an experiment or describe some latent variable of our model. Generally, the likelihood for the data $\mathcal{D} = \{\mathbf{s}_c \mid c \in \mathcal{C}\}$ and parameters $\boldsymbol{\theta} = \{\mathbf{p}_c \mid c \in \mathcal{C}\}$ can be written as

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{c \in \mathcal{C}} \mathrm{Mult}(\mathbf{s}_c \mid N_c, \mathbf{p}_c). \tag{5.1.1}$$

To concretize this type of model, we now give three examples in the context of MDPs.

EXAMPLE 1. As a first example, consider the case where the agent produces a state-action trajectory as

$$\mathcal{D} = \{x(1), u(1), \ldots, x(T-1), u(T-1), x(T)\}$$

and we want to model the state transition probabilities from a state $x(t) = x$ to a state $x(t+1) = x'$ under action $u(t) = u$; i.e., $p_{xux'} \coloneqq p(x' \mid x, u)$.

Here, we can represent the data $\mathcal{D} = \{\mathbf{s}_c \mid c \in \mathcal{C}\}$ using a product covariate space as $\mathcal{C} = \mathcal{X} \times \mathcal{U}$ with $K = n \equiv |\mathcal{X}|$ categories. Additionally, the elements of the count vector $\mathbf{s}_c \equiv \mathbf{s}_{xu} = [s_{xu1}, \ldots, s_{xun}]^\top$ are counting outgoing transitions from some state x for a given action u; i.e.,

$$s_{xux'} \coloneqq \sum_{t=1}^{T-1} \mathbb{1}(x(t) = x \wedge u(t) = u \wedge x(t+1) = x'), \quad \forall (x, u, x') \in \mathcal{X} \times \mathcal{U} \times \mathcal{X}.$$

Furthermore, the total number of transitions from state x under action u is $N_{xu} = \sum_{x' \in \mathcal{X}} s_{xux'}$. This yields the likelihood for the data $\mathcal{D}$ and parameters $\boldsymbol{\theta} = \{\mathbf{p}_{xu} \mid (x, u) \in \mathcal{X} \times \mathcal{U}\}$ as

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{x \in \mathcal{X}} \prod_{u \in \mathcal{U}} \mathrm{Mult}(\mathbf{s}_{xu} \mid N_{xu}, \mathbf{p}_{xu}).$$

EXAMPLE 2. Another example for the multinomial model is the case when modeling a stochastic policy of an agent in an MDP, i.e., $p_{xu} \coloneqq \mu(u, x) = p(u \mid x)$, given a state-action trajectory

$$\mathcal{D} = \{x(1), u(1), \ldots, x(T-1), u(T-1), x(T), u(T)\}.$$

Here, we consider covariate space $\mathcal{C} = \mathcal{X}$ with $K = m \equiv |\mathcal{U}|$ categories. The elements of the vector $\mathbf{s}_c \equiv \mathbf{s}_x = [s_{x1}, \ldots, s_{xm}]^\top$ are counting the number of times action u is observed at a particular state x; i.e.,

$$s_{xu} \coloneqq \sum_{t=1}^{T} \mathbb{1}(x(t) = x \wedge u(t) = u).$$

Therefore, the likelihood is given as

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{x \in \mathcal{X}} \mathrm{Mult}(\mathbf{s}_x \mid N_x, \mathbf{p}_x),$$

where $N_x = \sum_{u \in \mathcal{U}} s_{xu}$ is the total number of times we observe the agent choosing an action at state x.

EXAMPLE 3. As the last example, we consider the case when modeling the agent's intentions. In this case, we choose as the covariate space the state space, i.e., $\mathcal{C} = \mathcal{X}$, and we model $K = G$ distinct goals, which itself might be unobservable (Section 5.4.1.2). For this, we consider the count vector $\mathbf{s}_c \equiv \mathbf{s}_x = [s_{x1}, \ldots, s_{xG}]^\top$, where the element s_{xg} describes the number of times the agent follows the particular goal g at state x.

THE PRIOR DISTRIBUTION. As a model prior for the probability vectors $\boldsymbol{\theta} = \{\mathbf{p}_c \mid c \in \mathcal{C}\}$ a computationally straightforward approach is to use independent Dirichlet distributions for all covariate values; i.e.,

$$p(\boldsymbol{\theta}) = \prod_{c \in \mathcal{C}} \mathrm{Dir}(\mathbf{p}_c \mid \boldsymbol{\alpha}_c),$$

where $\boldsymbol{\alpha}_c \in \mathbb{R}^K_{>0}$ is a local concentration parameter for covariate value c. This approach is computationally appealing as the Dirichlet distribution is conjugate to the multinomial likelihood in Eq. (5.1.1). Hence, posterior inference can be calculated by updating the posterior parameters using the sufficient statistics of the model. However, the resulting model is agnostic to the rich correlation structure present in most MDPs and thus ignores much prior information about the underlying decision-making problem.

Therefore, a more informative approach is to model the probability vectors $\boldsymbol{\theta} = \{\mathbf{p}_c \mid c \in \mathcal{C}\}$ jointly using a common prior model to capture their dependency structure. Unfortunately,

this often yields a non-conjugate prior distribution to the multinomial likelihood and hence, leads to intractable exact posterior inference as the evidence of the distribution contains an intractable integral over the parameters $\boldsymbol{\theta}$. Additionally, even when using approximate inference schemes, such as MCMC or mean-field VI [26], a non-conjugacy leads to problems as the full conditional distributions needed for both methods can not be computed in closed form.

However, to introduce a dependency on the multinomial likelihood, using a shared prior over all parameters, a tractable method for approximate inference in dependent multinomial models has been developed to account for the inherent correlation of the probability vectors [89]. To this end, the following prior model was introduced,

$$\begin{aligned} \boldsymbol{\psi}_{\cdot k} &\sim \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \quad k = 1, \dots, K-1 \\ \mathbf{p}_c &= \boldsymbol{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{c\cdot}), \qquad\quad \forall c \in \mathcal{C}, \end{aligned} \tag{5.1.2}$$

with latent Gaussian variables $\boldsymbol{\psi}_{\cdot k} \in \mathbb{R}^C$ and hyper-parameters for the mean $\boldsymbol{\mu}_k \in \mathbb{R}^C$ and the positive-semidefinite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{C \times C}$. Herein, $\boldsymbol{\Pi}_{\mathrm{SB}} : \mathbb{R}^{K-1} \to \Delta^K$ is the *logistic stick-breaking transformation*, which is defined component wise for an input $\boldsymbol{\zeta} = [\zeta_1, \dots, \zeta_{K-1}]^\top \in \mathbb{R}^{K-1}$ as

$$\begin{aligned} \boldsymbol{\Pi}_{\mathrm{SB}}\boldsymbol{\zeta}) &= [\Pi_{\mathrm{SB}}^{(1)}(\boldsymbol{\zeta}), \dots, \Pi_{\mathrm{SB}}^{(K)}(\boldsymbol{\zeta})]^\top \\ \Pi_{\mathrm{SB}}^{(k)}(\boldsymbol{\zeta}) &= \sigma(\zeta_k) \prod_{j=1}^{k-1} (1 - \sigma(\zeta_j)), \quad k = 1, \dots, K-1 \\ \Pi_{\mathrm{SB}}^{(K)}(\boldsymbol{\zeta}) &= 1 - \sum_{k=1}^{K-1} \Pi_{\mathrm{SB}}^{(k)}(\boldsymbol{\zeta}) \end{aligned}$$

and $\sigma(\cdot)$ is the logistic function. The purpose of the *logistic stick-breaking transformation* in Eq. (5.1.2) is to map the real-valued Gaussian variables $\{\boldsymbol{\psi}_{\cdot k} \mid k = 1, \dots, K-1\}$ to the simplex by passing each entry through the sigmoid function and subsequently applying a regular stick-breaking construction [102]. Through the correlation structure $\boldsymbol{\Sigma}$ of the prior variables $\boldsymbol{\Psi} = [\boldsymbol{\psi}_{\cdot 1}, \dots, \boldsymbol{\psi}_{\cdot K-1}]$, the transformed probability vectors $\{\mathbf{p}_c \mid c \in \mathcal{C}\}$ become dependent. The corresponding PGM is depicted in Fig. 5.1.

It can be shown that by introducing a set of auxiliary PG variables $\boldsymbol{\Omega}$, a conjugate model in the augmented space can be established; i.e., the prior distribution $p(\boldsymbol{\Psi})$ has the same Gaussian distributional form as the full conditional distribution $p(\boldsymbol{\Psi} \mid \boldsymbol{\Omega}, \mathcal{D})$ [89]. Though the posterior distribution is still intractable under the augmented model, it enables a simple inference procedure based on blocked Gibbs sampling. Here, the now in a closed form, available full conditional distributions can be sampled in alternating order. As such, the Gaussian variables $\boldsymbol{\Psi}$ and the PG variables $\boldsymbol{\Omega}$ are sampled in turn, conditionally on each other and the count data $\mathcal{D}$.

The following section presents a different approximate posterior inference method based on VI utilizing this augmentation trick, which establishes a closed-form approximation scheme for the posterior distribution. Moreover, we present a hyper-parameter optimization method based on variational expectation-maximization (VEM) that allows us to calibrate our model to a particular problem type, avoiding the need for manual parameter tuning. Applied in

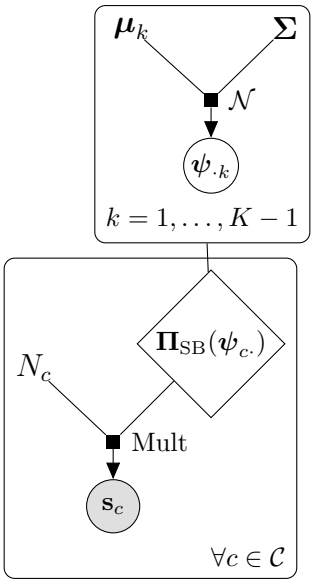


FIGURE 5.1: PGM for the generative model of the dependent multinomial model. The event probabilities of the multinomial distribution get correlated through the real-valued latent variables $\{\boldsymbol{\psi}_{\cdot k} \mid k = 1, \ldots, K\}$. These latent variables are mapped to the probability simplex by the *logistic stick-breaking transformation* $\boldsymbol{\Pi}_{\mathrm{SB}}(\cdot)$.

combination, this takes us beyond existing sampling-based approaches, providing a fast and automated inference algorithm for correlated count data.

5.2 VARIATIONAL INFERENCE FOR DEPENDENT MULTINOMIAL MODELS

Consider the generative model from Section 5.1, with the likelihood model in Eq. (5.1.1) and the correlation prior in Eq. (5.1.2). Here, we aim to compute the posterior distribution $p(\boldsymbol{\Psi} \mid \mathcal{D})$, where $\boldsymbol{\Psi} = [\boldsymbol{\psi}_{\cdot 1}, \ldots, \boldsymbol{\psi}_{\cdot K-1}]$ is the matrix of real-valued latent parameters, and the data $\mathcal{D} = \{\mathbf{s}_c \mid c \in \mathcal{C}\}$ contains the count vectors. Unfortunately, exact posterior inference is intractable since the calculation of $p(\boldsymbol{\Psi} \mid \mathcal{D})$ requires marginalization over the joint parameter space of all variables $\boldsymbol{\Psi}$. Hence, the goal of our inference procedure is to find an approximate posterior distribution. Instead of following a Monte Carlo approach as in [89], we resort to a variational approximation. Within this framework, we consider the optimization problem

$$\underset{q(\boldsymbol{\Psi}) \in \mathcal{Q}_{\boldsymbol{\Psi}}}{\text{minimize}} \quad \mathsf{KL}(q(\boldsymbol{\Psi}) \parallel p(\boldsymbol{\Psi} \mid \mathcal{D})), \tag{5.2.3}$$

which searches for the best approximating distribution

$$q^*(\boldsymbol{\Psi}) \approx p(\boldsymbol{\Psi} \mid \mathcal{D})$$

from a family of distributions $\mathcal{Q}_{\boldsymbol{\Psi}}$. It is hard to carry out this optimization under a multinomial likelihood because it involves intractable expectations over the variational distribution. However, in the following, we show that analogously to the inference scheme

of [89], a PG augmentation of $\mathbf{\Psi}$ makes the optimization tractable. To this end, we introduce a family of augmented posterior distributions $\mathcal{Q}_{\mathbf{\Psi},\mathbf{\Omega}}$ and instead consider the problem

$$\underset{q(\mathbf{\Psi},\mathbf{\Omega})\in\mathcal{Q}_{\mathbf{\Psi},\mathbf{\Omega}}}{\text{minimize}} \quad \mathsf{KL}(q(\mathbf{\Psi},\mathbf{\Omega}) \parallel p(\mathbf{\Psi},\mathbf{\Omega} \mid \mathcal{D})), \tag{5.2.4}$$

where we aim to find the optimal variational posterior in the augmented space

$$q^*(\mathbf{\Psi},\mathbf{\Omega}) \approx p(\mathbf{\Psi},\mathbf{\Omega} \mid \mathcal{D})$$

and $\mathbf{\Omega} = [\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_{K-1}] \in \mathbb{R}^{C\times K-1}$ denotes the matrix of auxiliary variables. Notice that the desired posterior can be recovered as the marginal $p(\mathbf{\Psi} \mid \mathcal{D}) = \int p(\mathbf{\Psi},\mathbf{\Omega} \mid \mathcal{D})\,\mathrm{d}\mathbf{\Omega}$, i.e., for the approximate posterior we can compute

$$q(\mathbf{\Psi}) = \int q(\mathbf{\Psi},\mathbf{\Omega})\,\mathrm{d}\mathbf{\Omega} \approx p(\mathbf{\Psi} \mid \mathcal{D}).$$

First, for solving Eq. (5.2.4), we use an equivalent formulation by maximizing the ELBO $\mathsf{L}[q] \leq \log p(\mathcal{D})$; i.e.,

$$\underset{q(\mathbf{\Psi},\mathbf{\Omega})\in\mathcal{Q}_{\mathbf{\Psi},\mathbf{\Omega}}}{\text{maximize}} \quad \mathsf{L}[q] = \mathsf{E}[\log p(\mathbf{\Psi},\mathbf{\Omega},\mathcal{D})] - \mathsf{E}[\log q(\mathbf{\Psi},\mathbf{\Omega})], \tag{5.2.5}$$

where above expectations are calculated w.r.t. the variational distribution $q(\mathbf{\Psi},\mathbf{\Omega})$. Hence, the formulation in Eq. (5.2.5) compared to Eq. (5.2.4) avoids computing the log evidence $\log p(\mathcal{D})$, which corresponds to the intractable computation

$$p(\mathcal{D}) = \int p(\mathbf{\Psi},\mathbf{\Omega},\mathcal{D})\,\mathrm{d}\mathbf{\Psi}\mathrm{d}\mathbf{\Omega}.$$

Second, in order to arrive at a tractable expression for the ELBO, we recapitulate the following data augmentation scheme derived in [89],

$$\begin{aligned} p(\mathbf{\Psi},\mathcal{D}) &= p(\mathbf{\Psi})p(\mathcal{D} \mid \mathbf{\Psi}) \\ &= \left(\prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma})\right)\left(\prod_{c\in\mathcal{C}} \mathrm{Mult}(\mathbf{s}_c \mid N_c, \mathbf{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{c\cdot}))\right) \\ &= \left(\prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma})\right)\left(\prod_{c=1}^{C}\prod_{k=1}^{K-1} \mathrm{Bin}(s_{ck} \mid b_{ck}, \sigma(\psi_{ck}))\right), \end{aligned} \tag{5.2.6}$$

where the stick-breaking representation of the multinomial distribution has been expanded using $b_{ck} = N_c - \sum_{j=1}^{k-1} s_{cj}$. From Eq. (5.2.6), by using the binomial PMF we arrive at

$$\begin{aligned} p(\mathbf{\Psi},\mathcal{D}) &= \left(\prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma})\right)\left(\prod_{c=1}^{C}\prod_{k=1}^{K-1} \binom{b_{ck}}{s_{ck}} \sigma(\psi_{ck})^{s_{ck}}(1-\sigma(\psi_{ck}))^{b_{ck}-s_{ck}}\right) \\ &= \left(\prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma})\right)\left(\prod_{c=1}^{C}\prod_{k=1}^{K-1} \binom{b_{ck}}{s_{ck}} \frac{(\exp(\psi_{ck}))^{s_{ck}}}{(1+\exp(\psi_{ck}))^{b_{ck}}}\right). \end{aligned} \tag{5.2.7}$$

The PG augmentation, as introduced in [103], is obtained using the integral identity

$$\frac{(\exp(\psi))^s}{(1+\exp(\psi))^b} = 2^{-b}\exp(\kappa\psi)\int \exp(-\omega\psi^2/2)\,\mathrm{PG}(\omega \mid b, 0)\,\mathrm{d}\omega, \tag{5.2.8}$$

with $\kappa = s - \frac{b}{2}$ and $\mathrm{PG}(\omega \mid b, 0)$ is the density of the PG distribution. Note that the PG distribution can be defined as an infinite sum, see Section A.2.12. Additionally, for a PG distributed variable

$$\zeta \sim \mathrm{PG}(\zeta \mid u, v)$$

we have the exponential tilting property

$$\mathrm{PG}(\zeta \mid u, v) = \frac{\exp(-\frac{v^2}{2}\zeta)\,\mathrm{PG}(\zeta \mid u, 0)}{\cosh^{-u}(v/2)}$$

and the first moment can be computed as

$$\mathsf{E}[\zeta] = \frac{u}{2v}\tanh(v/2).$$

Using the integral identity from Eq. (5.2.8) yields for Eq. (5.2.7)

$$p(\boldsymbol{\Psi}, \mathcal{D}) = \int \underbrace{\prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) \prod_{c=1}^{C} \binom{b_{ck}}{s_{ck}} 2^{-b_{ck}} \exp(\kappa_{ck}\psi_{ck}) \cdot \exp(-\omega_{ck}\psi_{ck}^2/2)\,\mathrm{PG}(\omega_{ck} \mid b_{ck}, 0)}_{p(\boldsymbol{\Psi}, \boldsymbol{\Omega}, \mathcal{D})}\,\mathrm{d}\boldsymbol{\Omega}, \tag{5.2.9}$$

where, $\kappa_{ck} = s_{ck} - b_{ck}/2$. Hence, the integrand of Eq. (5.2.9) defines the augmented distribution $p(\boldsymbol{\Psi}, \boldsymbol{\Omega}, \mathcal{D})$. This augmented formulation is very appealing since the augmented distributions

$$p(\boldsymbol{\Psi}, \boldsymbol{\Omega}, \mathcal{D}) = p(\boldsymbol{\Psi})p(\boldsymbol{\Omega}, \mathcal{D} \mid \boldsymbol{\Psi}) \propto p(\boldsymbol{\Psi} \mid \boldsymbol{\Omega}, \mathcal{D})$$

all have a Gaussian form in $\boldsymbol{\Psi}$ and are hence conjugate.

5.2.1 *Computing the Optimal Variational Distributions*

With this augmented distribution at hand, we can now derive by exploiting calculus of variations [31] the optimal distributions in Eq. (5.2.5). In this setting, we select a mean-field family for the class of variational distributions $\mathcal{Q}_{\boldsymbol{\Psi},\boldsymbol{\Omega}}$ as

$$q(\boldsymbol{\Psi}, \boldsymbol{\Omega}) = \prod_{k=1}^{K-1} q(\boldsymbol{\psi}_{\cdot k}) \prod_{c=1}^{C} q(\omega_{ck}). \tag{5.2.10}$$

5.2.1.1 *Calculation for the Variational Distribution $q(\boldsymbol{\psi}_{\cdot k})$*

First, we calculate the optimal forms of the variational distributions $q(\boldsymbol{\psi}_{\cdot k})$ for all categories $k \in \{1, \ldots, K-1\}$. Collecting all terms for the objective in Eq. (5.2.5) that depend on $\boldsymbol{\psi}_{\cdot k}$ gives

$$\mathsf{L}[q] = \mathsf{L}[q(\boldsymbol{\psi}_{\cdot k})] + \mathsf{L}_{\text{const}} \,.$$

Due to the factorization in Eq. (5.2.10) together with the augmented distribution $p(\boldsymbol{\Psi}, \boldsymbol{\Omega}, \mathcal{D})$ from Eq. (5.2.9), we have

$$\mathsf{L}[q(\boldsymbol{\psi}_{\cdot k})] = \mathsf{E}[\log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})] + \sum_{c=1}^{C} \mathsf{E}[\psi_{ck}]\kappa_{ck} - \sum_{c=1}^{C} \mathsf{E}\left[\omega_{ck}\psi_{ck}^2/2\right] - \mathsf{E}[\log q(\boldsymbol{\psi}_{\cdot k})].$$

The optimal distribution can be calculated by introducing the Lagrangian

$$\begin{aligned}\mathcal{L}(q(\boldsymbol{\psi}_{\cdot k}), \nu) = \mathsf{E}[\log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})] + \sum_{c=1}^{C} \mathsf{E}[\psi_{ck}]\kappa_{ck} - \sum_{c=1}^{C} \mathsf{E}\left[\omega_{ck}\psi_{ck}^2/2\right] \\ - \mathsf{E}[\log q(\boldsymbol{\psi}_{\cdot k})] + \nu \left(\int q(\boldsymbol{\psi}_{\cdot k}) \, d\boldsymbol{\psi}_{\cdot k} - 1 \right),\end{aligned}$$

which ensures that $q(\boldsymbol{\psi}_{\cdot k})$ is a proper density, using ν as a Lagrange multiplier to enforce the normalization constraint. The Euler Lagrange equation and optimality condition are

$$\frac{\delta \mathcal{L}}{\delta q} = 0 \quad \text{and} \qquad \frac{\partial \mathcal{L}}{\partial \nu} = 0, \tag{5.2.11}$$

respectively. The functional derivative of the Lagrangian yields

$$\frac{\delta \mathcal{L}}{\delta q} = \log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \sum_{c=1}^{C} \psi_{ck}\kappa_{ck} - \sum_{c=1}^{C} \mathsf{E}[\omega_{ck}]\psi_{ck}^2/2 - \log q(\boldsymbol{\psi}_{\cdot k}) - 1 + \nu.$$

By solving the Euler Lagrange equation for $q(\boldsymbol{\psi}_{\cdot k})$, we obtain

$$q(\boldsymbol{\psi}_{\cdot k}) = \exp\left(\nu - 1 + \log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \sum_{c=1}^{C} \psi_{ck}\kappa_{ck} - \sum_{c=1}^{C} \mathsf{E}[\omega_{ck}]\psi_{ck}^2/2 \right).$$

The optimality condition (normalization constraint) in Eq. (5.2.11) yields

$$\begin{aligned} q(\boldsymbol{\psi}_{\cdot k}) &\propto \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) \exp\left(-\frac{1}{2} \sum_{c=1}^{C} \mathsf{E}[\omega_{ck}]\psi_{ck}^2 + \sum_{c=1}^{C} \psi_{ck}\kappa_{ck} \right) \\ &\propto \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) \prod_{c=1}^{C} \mathcal{N}(\kappa_{ck}/\,\mathsf{E}[\omega_{ck}] \mid \psi_{ck}, 1/\,\mathsf{E}[\omega_{ck}]) \\ &= \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) \, \mathcal{N}(\operatorname{diag}(\mathsf{E}[\boldsymbol{\omega}_k])^{-1} \boldsymbol{\kappa}_k \mid \boldsymbol{\psi}_{\cdot k}, \operatorname{diag}(\mathsf{E}[\boldsymbol{\omega}_k])^{-1}), \end{aligned}$$

with $\boldsymbol{\omega}_k = [\omega_{1k}, \ldots, \omega_{Ck}]^\top$ and $\boldsymbol{\kappa}_k = [\kappa_{1k}, \ldots, \kappa_{Ck}]^\top$. Therefore, the optimal distribution $q(\boldsymbol{\psi}_{\cdot k})$ can be identified as a Gaussian by completing the square

$$q(\boldsymbol{\psi}_{\cdot k}) = \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\lambda}_k, \mathbf{V}_k), \tag{5.2.12}$$

with the variational parameters

$$\mathbf{V}_k = (\boldsymbol{\Sigma}^{-1} + \operatorname{diag}(\mathsf{E}[\boldsymbol{\omega}_k]))^{-1} \quad \text{and} \qquad \boldsymbol{\lambda}_k = \mathbf{V}_k(\boldsymbol{\kappa}_k + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k).$$

5.2.1.2 *Calculation for the Variational Distribution $q(\omega_{ck})$*

The distribution for $q(\omega_{ck})$ is calculated in a similar fashion. The ELBO in Eq. (5.2.5) in terms dependent on ω_{ck} can be written as

$$\mathsf{L}[q(\omega_{ck})] = \mathsf{L}[q(\omega_{ck})] + \mathsf{L}_{\text{const}},$$

with

$$\mathsf{L}[q(\omega_{ck})] = -\mathsf{E}[\omega_{ck}]\,\mathsf{E}\left[\psi_{ck}^2\right]/2 + \mathsf{E}[\log \mathrm{PG}(\omega_{ck} \mid b_{ck}, 0)] - \mathsf{E}[\log q(\omega_{ck})].$$

The Lagrangian for the distribution $q(\omega_{ck})$ is

$$\mathcal{L}(q(\omega_{ck}), \nu) = -\mathsf{E}[\omega_{ck}]\,\mathsf{E}\left[\psi_{ck}^2\right]/2 + \mathsf{E}[\log \mathrm{PG}(\omega_{ck} \mid b_{ck}, 0)] - \mathsf{E}[\log q(\omega_{ck})] + \nu\left(\int q(\omega_{ck})\, d\omega_{ck} - 1\right).$$

The functional derivative of the Lagrangian yields

$$\frac{\delta \mathcal{L}}{\delta q} = -\omega_{ck}\,\mathsf{E}\left[\psi_{ck}^2\right]/2 + \log \mathrm{PG}(\omega_{ck} \mid b_{ck}, 0) - \log q(\omega_{ck}) - 1 + \nu.$$

Solving the Euler Lagrange equation $\frac{\delta \mathcal{L}}{\delta q} = 0$ for $q(\omega_{ck})$, we find

$$q(\omega_{ck}) = \exp\left(\nu - 1 + \log \mathrm{PG}(\omega_{ck} \mid b_{ck}, 0) - \omega_{ck}\,\mathsf{E}\left[\psi_{ck}^2\right]/2\right).$$

The normalization constraint is used to identify

$$q(\omega_{ck}) \propto \mathrm{PG}(\omega_{ck} \mid b_{ck}, 0) \exp\left(-\omega_{ck}\,\mathsf{E}\left[\psi_{ck}^2\right]/2\right).$$

By exploiting the exponential tilting property of the PG distribution, that is,

$$\mathrm{PG}(\zeta \mid u, v) \propto \exp(-\frac{v^2}{2}\zeta)\,\mathrm{PG}(\zeta \mid u, 0),$$

we obtain

$$q(\omega_{ck}) = \mathrm{PG}(\omega_{ck} \mid b_{ck}, w_{ck}), \tag{5.2.13}$$

with the variational parameter $w_{ck} = \sqrt{\mathsf{E}[\psi_{ck}^2]}$.

5.2.2 *The Evidence Lower Bound for the Optimal Distributions*

Given the optimal distributions in Eqs. (5.2.12) and (5.2.13), we are now ready to derive an expression for the ELBO in Eq. (5.2.5); i.e.,

$$\mathsf{L}[q] = \mathsf{E}[\log p(\mathbf{\Psi}, \mathbf{\Omega}, \mathcal{D})] - \mathsf{E}[\log q(\mathbf{\Psi}, \mathbf{\Omega})].$$

Above expectations w.r.t. the previously derived optimal distributions yield

$$\begin{aligned}\mathsf{E}[\log p(\boldsymbol{\Psi}, \boldsymbol{\Omega}, \mathcal{D})] &= \sum_{k=1}^{K-1} \mathsf{E}[\log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})] + \sum_{k=1}^{K-1}\sum_{c=1}^{C} \log \binom{b_{ck}}{s_{ck}} - \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log 2 \\ &\quad + \sum_{k=1}^{K-1} \boldsymbol{\lambda}_k^\top \boldsymbol{\kappa}_k + \sum_{k=1}^{K-1}\sum_{c=1}^{C} \mathsf{E}[\log \mathrm{PG}(\omega_{ck} \mid b_{ck}, w_{ck})] - \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log\left(\cosh \frac{w_{ck}}{2}\right), \\ \mathsf{E}[\log q(\boldsymbol{\Psi}, \boldsymbol{\Omega})] &= \sum_{k=1}^{K-1} \mathsf{E}[\log \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\lambda}_k, \mathbf{V}_k)] + \sum_{k=1}^{K-1}\sum_{c=1}^{C} \mathsf{E}[\log \mathrm{PG}(\omega_{ck} \mid b_{ck}, w_{ck})].\end{aligned}$$

Canceling out the terms $\mathsf{E}[\log \mathrm{PG}(\omega_{ck} \mid b_{ck}, w_{ck})]$ and rewriting the prior and variational terms as KL divergence, we obtain

$$\begin{aligned}\mathsf{L}[q] = &- \sum_{k=1}^{K-1} \mathsf{KL}(\mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\lambda}_k, \mathbf{V}_k) \parallel \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})) + \sum_{k=1}^{K-1}\sum_{c=1}^{C} \log \binom{b_{ck}}{s_{ck}} \\ &- \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log 2 + \sum_{k=1}^{K-1} \boldsymbol{\lambda}_k^\top \boldsymbol{\kappa}_k - \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log\left(\cosh \frac{w_{ck}}{2}\right).\end{aligned}$$

Finally, by computing the KL divergence, the ELBO can be expressed in terms of the variational parameters as

$$\begin{aligned}\mathsf{L}[q] = &- \frac{K-1}{2}|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{k=1}^{K-1} \log|\mathbf{V}_k| - \frac{1}{2}\sum_{k=1}^{K-1} \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{V}_k) \\ &- \frac{1}{2}\sum_{k=1}^{K-1} (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k) + C(K-1) + \sum_{k=1}^{K-1}\sum_{c=1}^{C} \log \binom{b_{ck}}{s_{ck}} \\ &- \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log 2 + \sum_{k=1}^{K-1} \boldsymbol{\lambda}_k^\top \boldsymbol{\kappa}_k - \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck} \log\left(\cosh \frac{w_{ck}}{2}\right).\end{aligned}$$

5.2.3 *Optimizing the Evidence Lower Bound*

Here, we again summarize the results of our variational procedure and give an optimization algorithm. A PGM for the optimal variational mean-field approximation to the true posterior is depicted in Fig. 5.2

For the optimal distributions, we have

$$q(\boldsymbol{\psi}_{\cdot k}) = \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\lambda}_k, \boldsymbol{V}_k) \quad \text{and} \qquad q(\omega_{ck}) = \mathrm{PG}(\omega_{ck} \mid b_{ck}, w_{ck}).$$

The optimal parameters and first moments of the variational distributions are

$$\begin{aligned} w_{ck} = \sqrt{\mathrm{E}[\psi_{ck}^2]}, \quad \mathbf{V}_k = (\boldsymbol{\Sigma}^{-1} + \mathrm{diag}(\mathrm{E}[\boldsymbol{\omega}_k]))^{-1}, \quad \boldsymbol{\lambda}_k = \mathbf{V}_k(\boldsymbol{\kappa}_k + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\ \mathrm{E}[\psi_{ck}^2] = \left((\mathbf{V}_k)_{cc} + \lambda_{ck}^2\right), \quad \mathrm{E}[\omega_{ck}] = \frac{b_{ck}}{2w_{ck}} \tanh(w_{ck}/2), \end{aligned} \tag{5.2.14}$$

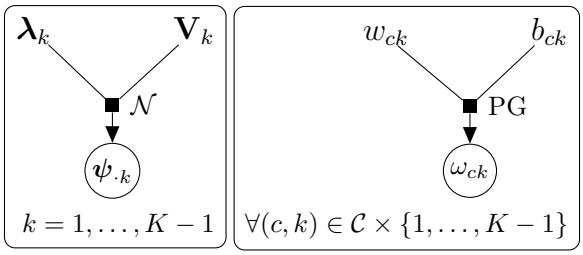


FIGURE 5.2: PGM for the approximate posterior distribution under the mean-field assumption. By introducing the PG distributed auxiliary variables $\{\omega_{ck} \mid (c,k) \in \mathcal{C} \times \{1,\ldots,K\}\}$, the optimal variational distribution for the latent variables $\boldsymbol{\Psi}$ become Gaussian under the augmented mean-field assumption. Note that the conditioned data is included by fitting the variational parameters.

with $\boldsymbol{\omega}_k = [\omega_{1k}, \ldots, \omega_{Ck}]^\top$ and $\boldsymbol{\kappa}_k = [\kappa_{1k}, \ldots, \kappa_{Ck}]^\top$. Additionally, the ELBO can be computed as

$$\begin{aligned}\mathsf{L}[q] = &-\frac{K-1}{2}|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{k=1}^{K-1}\log|\mathbf{V}_k| - \frac{1}{2}\sum_{k=1}^{K-1}\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{V}_k) \\ &-\frac{1}{2}\sum_{k=1}^{K-1}(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k) + C(K-1) + \sum_{k=1}^{K-1}\sum_{c=1}^{C}\log\binom{b_{ck}}{s_{ck}} \\ &-\sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck}\log 2 + \sum_{k=1}^{K-1}\boldsymbol{\lambda}_k^\top\boldsymbol{\kappa}_k - \sum_{k=1}^{K-1}\sum_{c=1}^{C} b_{ck}\log\left(\cosh\frac{w_{ck}}{2}\right).\end{aligned} \tag{5.2.15}$$

The variational approximation can then be optimized through coordinate-wise ascent by cycling through the parameters and their moments in Eq. (5.2.14). Finally, the corresponding distribution over probability vectors $\{\mathbf{p}_c \mid c \in \mathcal{C}\}$ is defined implicitly through the deterministic relationship in Eq. (5.1.2).

5.2.4 *Hyper-Parameter Optimization*

For hyper-parameter learning, we employ A VEM approach [104] to optimize the ELBO after each update of the variational parameters. Assuming a set of hyper-parameters namely a covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}$ parametrized by a vector $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_J]^\top$ and the set of mean vectors $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k \mid k = 1, \ldots, K-1\}$ the ELBO as function of the hyper parameters can be written as

$$\begin{aligned}\mathsf{L}(\boldsymbol{\phi}, \boldsymbol{\mu}) = &-\frac{K-1}{2}|\boldsymbol{\Sigma}_{\boldsymbol{\phi}}| - \frac{1}{2}\sum_{k=1}^{K-1}\operatorname{tr}(\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1}\mathbf{V}_k) \\ &-\frac{1}{2}\sum_{k=1}^{K-1}(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k) + \mathsf{L}_{\text{const}}\,.\end{aligned} \tag{5.2.16}$$

5.2.4.1 *Derivation of the Optimal Value for the Mean $\boldsymbol{\mu}_k$*

For the optimal mean parameters $\{\boldsymbol{\mu}_k \mid k = 1, \ldots, K-1\}$, we calculate the gradient of Eq. (5.2.16) as

$$\frac{\partial \mathsf{L}}{\partial \boldsymbol{\mu}_k} = \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} \left(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k\right).$$

Setting the gradient to zero, we obtain

$$\boldsymbol{\mu}_k = \boldsymbol{\lambda}_k. \tag{5.2.17}$$

5.2.4.2 *Derivation of the Optimal Hyper-Parameters of $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}$*

Analogously, for the optimal hyper-parameters $\boldsymbol{\phi}$ we calculate the gradient of Eq. (5.2.16) as

$$\begin{aligned}\frac{\partial \mathsf{L}}{\partial \phi_j} = -\frac{K-1}{2} \operatorname{tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} \frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\phi}}}{\partial \phi_j}\right) + \frac{1}{2} \sum_{k=1}^{K-1} \operatorname{tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} \frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\phi}}}{\partial \phi_j} \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} \boldsymbol{V}_k\right) \\ + \frac{1}{2} \sum_{k=1}^{K-1} (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} \frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\phi}}}{\partial \phi_j} \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k).\end{aligned} \tag{5.2.18}$$

When considering the special case of a scaled covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\phi}} = \phi \tilde{\boldsymbol{\Sigma}}$, we can find the optimizing hyper-parameter ϕ in closed form. Note that $\frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\phi}}}{\partial \phi} = \tilde{\boldsymbol{\Sigma}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} = \frac{1}{\phi} \tilde{\boldsymbol{\Sigma}}^{-1}$, therefore, the gradient computes to

$$\frac{\partial \mathsf{L}}{\partial \phi} = -\frac{K}{2\phi} \operatorname{tr}(\mathbf{I}) + \frac{1}{2\phi^2} \sum_{k=1}^{K-1} \operatorname{tr}(\tilde{\boldsymbol{\Sigma}}^{-1} \mathbf{V}_k) + \frac{1}{2\phi^2} \sum_{k=1}^{K-1} (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top \tilde{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k).$$

Setting the derivative to zero and solving for ϕ, we obtain the closed-form expression

$$\phi = \frac{1}{KS} \sum_{k=1}^{K-1} \operatorname{tr}\left(\tilde{\boldsymbol{\Sigma}}^{-1} \left(\boldsymbol{V}_k + (\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)(\boldsymbol{\mu}_k - \boldsymbol{\lambda}_k)^\top\right)\right). \tag{5.2.19}$$

5.2.4.3 *The Maximization Step*

The ELBO can now be optimized w.r.t. the hyper-parameters. For this we can use for the mean parameters $\{\boldsymbol{\mu}_k \mid k = 1, \ldots, K-1\}$ the closed form solution in Eq. (5.2.17). For the optimization of the covariance parameters $\boldsymbol{\phi}$, we can resort to a numerical scheme using the gradient expression in Eq. (5.2.18); however, this requires a full inversion of the covariance matrix in each update step. This can be avoided using the provided closed-form expression in Eq. (5.2.19) for the special case where $\boldsymbol{\phi}$ is a scale parameter, i.e., $\boldsymbol{\Sigma}_{\boldsymbol{\phi}} = \phi \tilde{\boldsymbol{\Sigma}}$, for some fixed $\tilde{\boldsymbol{\Sigma}}$. Here, the closed-form solution does not have to perform repeated matrix inversions since $\tilde{\boldsymbol{\Sigma}}^{-1}$, being independent of all hyper-parameters and variational parameters, can be evaluated at the start of the optimization procedure.

For the experiments, we consider a squared exponential covariance function of the form $(\boldsymbol{\Sigma}_{\boldsymbol{\phi}})_{cc'} = \phi \exp\left(-\frac{d(c,c')^2}{l^2}\right)$, with a covariate distance measure $d : \mathcal{C} \times \mathcal{C} \to \mathbb{R}_{\geq 0}$ and a length scale $l \in \mathbb{R}_{\geq 0}$ adapted to the specific modeling scenario. Yet, we note that multiple covariance functions can be easily compared against each other for model selection purposes based on the resulting values of the ELBO [104]. Also, a combination of functions can be employed, provided that the resulting covariance matrix is positive semi-definite, see covariance kernels of GPs [73].

Finally, the entire algorithm for finding the optimal variational distribution with optimal hyper-parameters is shown in Fig. 5.3

input : $\mathcal{D} = \{\mathbf{s}_c \mid c \in \mathcal{C}\}$: data set
output : $\{\boldsymbol{\lambda}_k, \mathbf{V}_k \mid k = 1, \dots, K-1\}$: optimal Gaussian variational parameters
$\{w_{ck} \mid (c,k) \in \mathcal{C} \times \{1, \dots, K-1\}\}$: optimal PG variational parameters
$\boldsymbol{\phi}, \{\boldsymbol{\mu}_k \mid k = 1, \dots, K-1\}$: optimal hyper-parameters

Initialize variational moments in Eq. (5.2.14)
while *ELBO in Eq.* (5.2.15) *not converged* **do**
  Update variational parameters $\{\boldsymbol{\lambda}_k, \mathbf{V}_k \mid k = 1, \dots, K-1\}$ and
  $\{w_{ck} \mid (c,k) \in \mathcal{C} \times \{1, \dots, K-1\}\}$ by cycling through moments in Eq. (5.2.14)
  Update mean hyper-parameters $\{\boldsymbol{\mu}_k \mid k = 1, \dots, K-1\}$ using Eq. (5.2.17)
  Update covariance hyper-parameters $\boldsymbol{\phi}$ using a gradient ascent step with
  Eq. (5.2.18) or using closed form update Eq. (5.2.19)
end
return $\{\boldsymbol{\lambda}_k, \mathbf{V}_k \mid k = 1, \dots, K-1\}$
$\{w_{ck} \mid (c,k) \in \mathcal{C} \times \{1, \dots, K-1\}\}$
$\boldsymbol{\phi}, \{\boldsymbol{\mu}_k \mid k = 1, \dots, K-1\}$

FIGURE 5.3: Pseudo-Code for VI based on PG augmentation.

5.3 POSTERIOR DISTRIBUTIONS IN DECISION-MAKING

The presented inference framework lends itself nicely to dealing with problems in decision-making under uncertainty. Here, we restrict ourselves to two significant problems in the context of decision-making in MDPs: (i) reconstructing a policy and (ii) estimating the transition model. Using respective estimates, many problems of interest can be solved, such as imitation learning, subgoal modeling, system identification, and BRL. However, we would like to point out that the same modeling principles can be applied in many other situations, e.g., for behavior coordination among agents [86] or knowledge transfer between related tasks [105], to name just two examples, though a more comprehensive evaluation study is left for future work.

5.3.1 *Policy Reconstruction*

The first type of problem we consider here is to reconstruct the policy of the agent (in this context called the *expert*), as described in Section 5.1, Example 2. For the reconstruction, we suppose to have access to a demonstration data set $\mathcal{D} = \{x(t), u(t) \mid t = 1, \dots, T\}$ containing T state-action pairs, where each action has been generated through the expert policy; i.e.,

$$u(t) \sim p(u(t) \mid x(t)) \equiv \mu(u(t), x(t)).$$

Assuming a discrete state and action space, with $n = |\mathcal{X}|$ and $m = |\mathcal{U}|$, respectively, the policy can be represented as a stochastic matrix $\boldsymbol{\Pi} = [\mathbf{p}_1, \dots, \mathbf{p}_n]$, whose xth column $\mathbf{p}_x \in \Delta^m$ represents the local action distribution of the expert at state x in form of a vector. Our goal is to estimate this matrix from the demonstrations $\mathcal{D}$. In this setting, we construct the count vector $\mathbf{s}_x = [s_{x1}, \dots, s_{xm}]^\top$ using

$$s_{xu} \coloneqq \sum_{t=1}^{T} \mathbb{1}(x(t) = x \wedge u(t) = u). \tag{5.3.20}$$

Therefore, the inference problem can be directly mapped to our PG model, with covariate space $\mathcal{C} = \mathcal{X}$ and $K = m$ categories. This allows to jointly estimate the coupled quantities $\{\mathbf{p}_x \mid x \in \mathcal{X}\}$ through their latent representation $\boldsymbol{\Psi}$ by approximating the posterior distribution $p(\boldsymbol{\Psi} \mid \mathcal{D})$ in Eq. (5.2.3). Given the optimal variational distribution

$$q(\boldsymbol{\Psi}) = \prod_{k=1}^{K-1} \mathcal{N}(\boldsymbol{\psi}_{\cdot k} \mid \boldsymbol{\lambda}_k, \mathbf{V}_k),$$

statistics can be easily obtained such as the posterior means

$$\mathsf{E}[\boldsymbol{\psi}_{\cdot k}] = \boldsymbol{\lambda}_k \quad \forall k \in \{1, \dots, K-1\}.$$

Note that even though the variational distribution over the latent variables $\boldsymbol{\Psi}$ has a desirable factorized Gaussian form, the corresponding distributions $q(\mathbf{p}_x)$ for the probability vectors $\{\mathbf{p}_x \mid x \in \mathcal{X}\}$, with $\mathbf{p}_x = \boldsymbol{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{x\cdot})$ are given by the respective Jacobian transforms; i.e.,

$$q(\mathbf{p}_x) = \left| \frac{\partial \mathbf{p}_x}{\partial \boldsymbol{\psi}_{x\cdot}} \right|^{-1} q(\boldsymbol{\psi}_{x\cdot}), \quad \forall x \in \mathcal{X}. \tag{5.3.21}$$

This leads to problems when computing posterior statistics, such as the posterior mean $\mathsf{E}[\mathbf{p}_x]$ under the probability measure in Eq. (5.3.21), since the posterior mean does not have an analytic form. However, a tractable alternative is to use the mean of the posterior of the latent variable and then to use the transform as an estimate for the probability vectors; i.e.,

$$\hat{\mathbf{p}}_x = \boldsymbol{\Pi}_{\mathrm{SB}}(\mathsf{E}[\boldsymbol{\psi}_{x\cdot}]) = \boldsymbol{\Pi}_{\mathbf{SB}}([\lambda_{x1}, \dots, \lambda_{xm}]^\top),$$

where λ_{xk} is the xth component of the vector $\boldsymbol{\lambda}_k$.

5.3.2 *Transition Model Estimation*

Another type of problem, which can be solved using the presented approximate inference scheme, is faced when we want to estimate the transition probabilities of an MDP as in Section 5.1, Example 1. In this setting, we consider the state-action trajectory

$$\mathcal{D} = \{x(1), u(1), \ldots, x(T-1), u(T-1), x(T)\}, \tag{5.3.22}$$

with $T-1$ state transitions. Building the model with covariate space $\mathcal{C} = \mathcal{X} \times \mathcal{U}$ and $K = n \equiv |\mathcal{X}|$ categories. We construct the count vector $\mathbf{s}_{xu} = [s_{xu1}, \ldots, s_{xun}]^\top$ as

$$s_{xux'} \coloneqq \sum_{t=1}^{T-1} \mathbb{1}(x(t) = x \wedge u(t) = u \wedge x(t+1) = x'), \quad \forall (x, u, x') \in \mathcal{X} \times \mathcal{U} \times \mathcal{X}, \tag{5.3.23}$$

where the element $s_{xux'}$ represents the number of observed transitions from state x to x' for action u. Here, our goal is to estimate the transition probability vectors $\{\mathbf{p}_{xu} \mid (x, u) \in \mathcal{X} \times \mathcal{U}\}$. Hence, this can be achieved by fitting the variational posterior distribution of the latent variables $\boldsymbol{\Psi}$. Again, we can easily compute statistics like the posterior mean, or we can calculate the density of the probability vectors using the Jacobian transformation in Eq. (5.3.21).

When making decisions in an MDP the posterior distribution can be directly used by, e.g., following a greedy strategy w.r.t. to the posterior mean.

GREEDY STRATEGY. For a *greedy strategy*, we apply the deterministic policy $\mu : \mathcal{X} \to \mathcal{U}$ which maximizes the state-action values $\{Q(x, u) \mid (x, u) \in \mathcal{X} \times \mathcal{U}\}$ under the posterior mean model, hence,

$$\begin{aligned}
&\hat{\mathbf{p}}_{xu} = \boldsymbol{\Pi}_{\mathrm{SB}}(\mathsf{E}[\boldsymbol{\psi}_{xu\cdot}]) \\
&Q(x, u) = R(x, u) + \gamma \sum_{x' \in \mathcal{X}} \hat{p}_{xux'} \max_{u' \in \mathcal{U}} Q(x', u') \\
&u = \mu(x) = \arg\max_{u' \in \mathcal{U}} Q(x, u').
\end{aligned}$$

Other strategies can be found by applying the principle of BRL. BRL offers a natural playground for the task of combined model-learning and decision-making as it intrinsically balances the importance of information gathering and instantaneous reward maximization, avoiding the exploration-exploitation dilemma encountered in classical RL schemes [14]. There are numerous BRL algorithms such as BEETLE [99] or BAMCP [100], which approximately plan using the posterior belief as a state in an augmented Bayes-adaptive Markov decision process (BAMDP) [106]. However, as these methods tend to be very time-consuming, we follow a different route here and apply a simple bandit algorithm [30]. We can easily apply posterior sampling for reinforcement learning (PSRL) [107] for the presented model, where future actions are planned using a probabilistic model of the environment's transition dynamics. For PSRL, we consider two variants.

PSRL SAMPLING VARIANT. In the first variant we compute the optimal state-action-values $\{Q^{(l)}(x,u) \mid (x,u,l) \in \mathcal{X} \times \mathcal{U} \times \{1,\dots,L\}\}$ for a fixed number L of posterior samples representing instantiations of the transition model by solving the corresponding Bellman equations for $Q^{(l)}(x,u)$ as

$$\begin{aligned} &\boldsymbol{\Psi}^{(l)} \sim q(\boldsymbol{\Psi}^{(l)}) \\ &\mathbf{p}_{xu}^{(l)} = \boldsymbol{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{xu\cdot}^{(l)}) \\ &Q^{(l)}(x,u) = R(x,u) + \gamma \sum_{x' \in \mathcal{X}} p_{xux'} \max_{u' \in \mathcal{U}} Q^{(l)}(x',u'). \end{aligned} \qquad l = 1,\dots,L \tag{5.3.24}$$

After calculating the state-action-values for all samples, by, e.g., value iteration, we choose the deterministic policy $\mu : \mathcal{X} \to \mathcal{U}$ that yields the highest expected return on average; i.e.,

$$u = \mu(x) = \arg\max_{u' \in \mathcal{U}} \frac{1}{L} \sum_{l=1}^{L} Q^{(l)}(x,u')$$

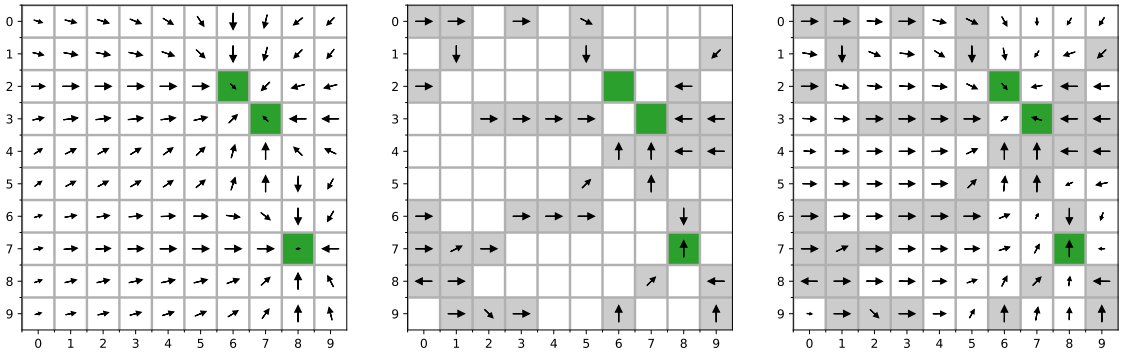
PSRL MEAN VARIANT. In the second variant, we select the greedy policy dictated by the empirical posterior mean of the transition dynamics; i.e.,

$$\begin{aligned} &\boldsymbol{\Psi}^{(l)} \sim q(\boldsymbol{\Psi}^{(l)}) && l = 1,\dots,L \\ &\hat{\mathbf{p}}_{xu} = \frac{1}{L} \sum_{l=1}^{L} \boldsymbol{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{xu\cdot}^{(l)}) \\ &Q(x,u) = R(x,u) + \gamma \sum_{x' \in \mathcal{X}} \hat{p}_{xux'} \max_{u' \in \mathcal{U}} Q(x',u') \\ &u = \mu(x) = \arg\max_{u' \in \mathcal{U}} Q(x,u'). \end{aligned} \tag{5.3.25}$$

In both variants, the obtained policy is followed for a fixed number of transitions before new observations are taken into account for updating the posterior distribution. These variants of the PSRL principle are a form of Thompson sampling [108]. Thompson sampling, which performs well in practice, trades-off exploration and exploitation naturally, since for uncertain posterior distributions, samples tend to be very random and lead to exploration. Additionally, for peaked, meaning certain, posterior distributions, a more greedy strategy is often chosen as the samples become more deterministic. Another appealing property of the sampling-based approach in, e.g., Eqs. (5.3.24) and (5.3.25) is that, we do not have to compute the Jacobian transforms for the probability vectors as in Eq. (5.3.21). However, note that applying other bandit strategies with the presented inference scheme are also possible, such as UCB as presented in Chapter 4.

5.4 EVALUATION

To demonstrate the versatility of our inference framework, we test it on several modeling scenarios that commonly occur in decision-making contexts. For this, we concentrate on imitation learning, subgoal modeling, system identification, and BRL.



(a) Expert policy. (b) Dirichlet estimate. (c) PG estimate.

FIGURE 5.4: Imitation learning example. The expert policy in (a) is reconstructed using the posterior mean estimates of (b) an independent Dirichlet policy model and (c) a correlated PG model, based on action data observed at the states marked in gray. The PG joint estimate of the local policies yields a significantly improved reconstruction.

5.4.1 *Policy Reconstruction in a Grid World Example*

5.4.1.1 *Imitation Learning*

First, we illustrate our framework on an imitation learning example and try to reconstruct the policy from an expert, as described in Section 5.3.1. To demonstrate the advantages of the presented joint inference approach over a correlation-agnostic estimation method, we compare our framework to the independent Dirichlet model described in Section 5.1. Both reconstruction methods are evaluated on a classical grid world scenario comprising $n = 100$ states and $m = 4$ actions. Each action triggers a noisy transition in one of the four cardinal directions such that the pattern of the resulting next-state distribution resembles a discretized Gaussian distribution centered around the targeted adjacent state. Rewards are distributed randomly in the environment. The expert follows a near-optimal stochastic policy, choosing actions from a softmax distribution obtained from the state-action-values (Q-values) of the current state. An example scenario is shown in Fig. 5.4a, where the displayed arrows are obtained by weighting the four unit-length vectors associated with the action set $\mathcal{U} = \{left, down, right, up\}$ according to their local action probabilities. The reward locations are highlighted in green.

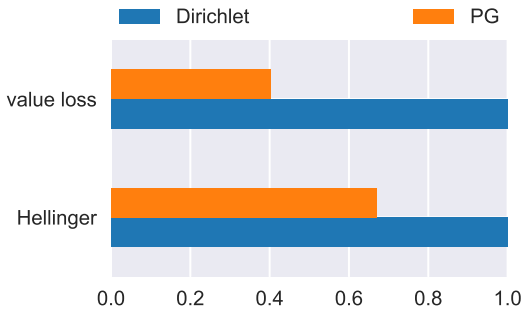


FIGURE 5.5: Normalized evaluation metrics for the imitation learning example. The Hellinger distance to the expert policy and the corresponding value loss [93] is shown for both an independent Dirichlet policy model and a correlated PG model.

The results are depicted in Figs. 5.4 and 5.5. Figure 5.4b shows the reconstruction of the policy obtained through the independent Dirichlet model. Since no dependencies between the local action distributions are considered in this approach, a posterior estimate can only be obtained for states where demonstration data is available, highlighted by the gray coloring of the background. For all remaining states, the mean estimate predicts a uniform action choice for the expert behavior since no action is preferred by the symmetry of the Dirichlet prior, resulting in an effective arrow length of zero. By contrast, the PG model (Fig. 5.4c) is able to generalize the expert behavior to unobserved regions of the state space, resulting in significantly improved reconstruction of the policy (Fig. 5.5). To capture the underlying correlations, we used the Euclidean distance between the grid positions as covariate distance measure d and set l to the maximum occurring distance value.

5.4.1.2 *Subgoal Modeling*

In many situations, modeling the actions of an agent is not of primary interest or proves to be difficult, e.g., because a more comprehensive understanding of the agent's behavior is desired (see inverse reinforcement learning (IRL) [109] and preference elicitation [110]) or because the policy is of complex form due to intricate system dynamics. A typical example is robot object manipulation, where contact-rich dynamics make it difficult for a controller trained from few demonstrations to generalize the expert behavior appropriately[111]. A simplistic example illustrating this problem is depicted in Fig. 5.7a, where the agent behavior is heavily affected by the geometry of the environment, and the action profiles at two wall-separated states differ drastically.

Similarly to the setup from Sections 5.3.1 and 5.4.1.1, we aspire to reconstruct the shown behavior from a demonstration data set of the form

$$\mathcal{D} = \{x(t), u(t) \mid t = 1, \dots, T\},$$

depicted in Fig. 5.7b. However, this time, we follow a conceptually different line of reasoning and assume that each state $x \in \mathcal{X}$ has an associated subgoal g_x that the agent is targeting at that state. Thus, action is considered as being drawn from some goal-dependent action distribution as

$$u(t) \sim p(u(t) \mid x(t), g_{x(t)}).$$

For our example, we adopt the normalized softmax action model described in [93]. Spatial relationships between the agent's decisions are taken into account with the help of our PG framework by coupling the probability vectors that govern the underlying subgoal selection process; i.e.,

$$g_x \sim \mathrm{Cat}(g_x \mid \mathbf{p}_x),$$

where $\mathbf{p}_x$ is described through the stick-breaking construction in Eq. (5.1.2). Accordingly, the underlying covariate space of the PG model is $\mathcal{C} = \mathcal{X}$ and we have $K = G$ categories, with a total number G distinct subgoals.

With the additional level of hierarchy introduced, the count data $\{\mathbf{s}_x \mid x \in \mathcal{X}\}$ to train our model (see Eq. (5.3.20)) is not directly available since the subgoals $\{g_x \mid x \in \mathcal{X}\}$ are not observable, for a PGM see Fig. 5.6. For demonstration purposes, instead of deriving the full

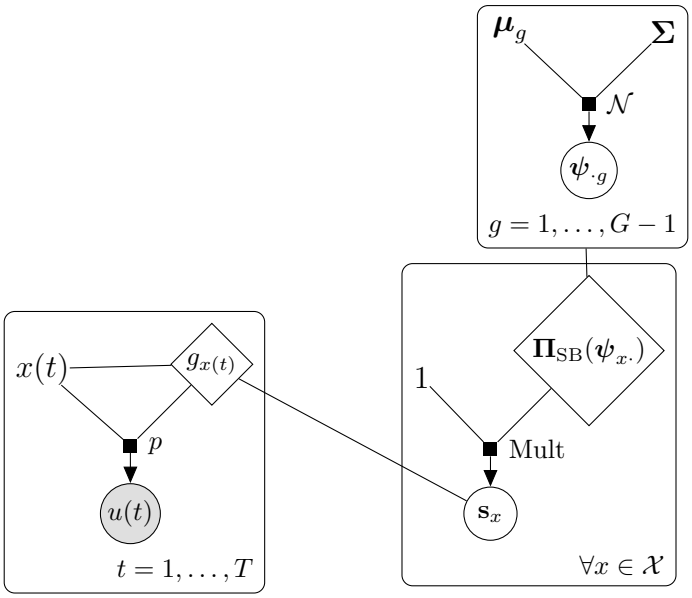


FIGURE 5.6: PGM of the generative subgoal model using the suggested correlation prior with a dependent multinomial model for the various goals.

variational update for the extended model, we follow a more straightforward strategy that leverages the existing inference framework within a Gibbs sampling procedure, switching between variational updates and drawing posterior samples of the latent subgoal variables. More precisely, we iterate between

- computing the variational approximation in Eq. (5.2.4) for a given set of subgoals $\{g_x \mid x \in \mathcal{X}\}$, treating each subgoal as single observation count, i.e.,

$$q(\boldsymbol{\Psi}, \boldsymbol{\Omega}) \approx p(\boldsymbol{\Psi}, \boldsymbol{\Omega} \mid \mathcal{D}')$$
$$\mathcal{D}' = \{\mathbf{s}_x = \mathrm{OneHot}(g_x) \mid x \in \mathcal{X}\}$$

  and

- updating the latent assignments based on the induced goal distributions, i.e.,

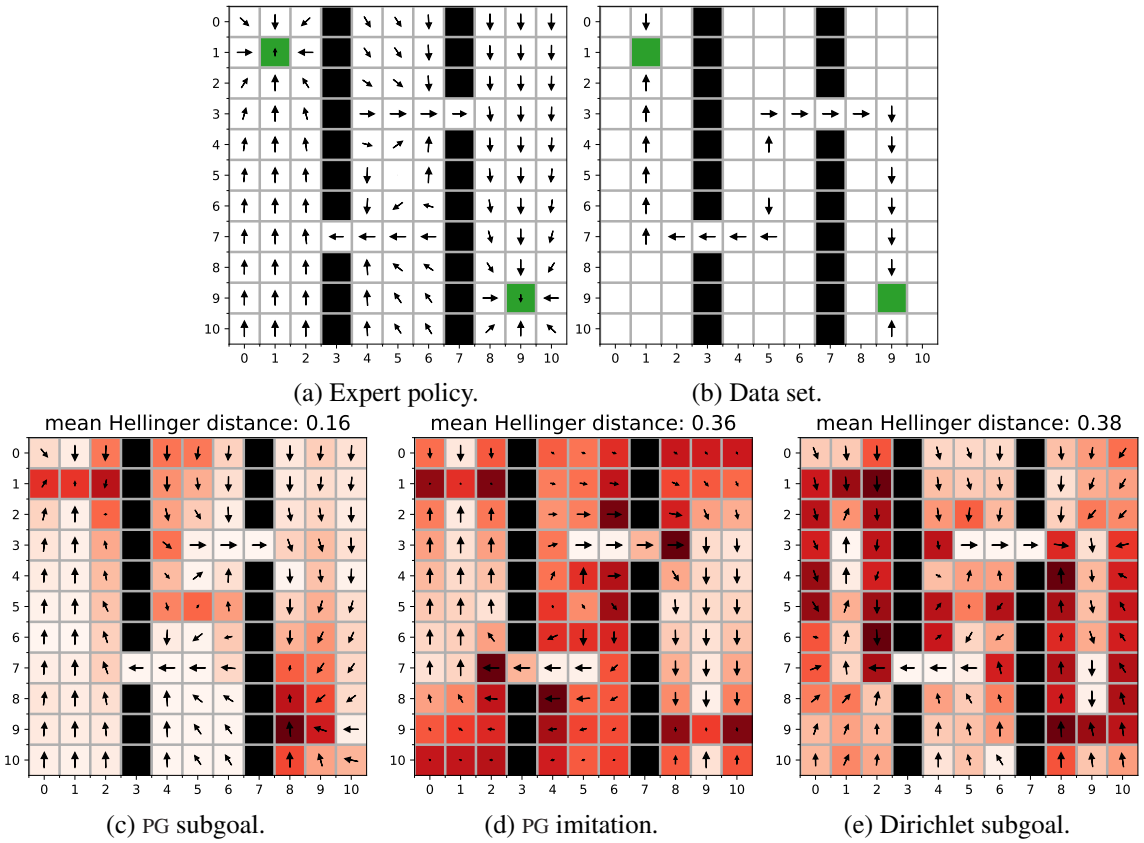
$$\boldsymbol{\Psi} \sim q(\boldsymbol{\Psi})$$
$$g_x \mid \mathcal{D}, \boldsymbol{\psi}_{x\cdot} \sim p(g_x \mid \mathcal{D}, \boldsymbol{\psi}_{x\cdot}) \quad \forall x \in \mathcal{X},$$

  where

$$p(g_x \mid \mathcal{D}, \boldsymbol{\psi}_{x\cdot}) \propto p(\mathcal{D} \mid g_x) p(g_x \mid \boldsymbol{\psi}_{x\cdot})$$
$$= \left\{ \prod_{t=1}^{T} [p(u(t) \mid x(t), g_x)]^{\mathbb{1}(x(t)=g_x)} \right\} \mathrm{Cat}(g_x \mid \boldsymbol{\Pi}_{\mathrm{SB}}(\boldsymbol{\psi}_{x\cdot})).$$

The results are depicted in Fig. 5.7. Figure 5.7c shows the policy model obtained by averaging the predictive action distributions of $L = 100$ drawn subgoal configurations; i.e.,

$$\hat{\mu}(u, x) = \frac{1}{L} \sum_{l=1}^{L} p(u \mid x, g_x^{(l)}),$$



(a) Expert policy. (b) Data set.

(c) PG subgoal. (d) PG imitation. (e) Dirichlet subgoal.

FIGURE 5.7: Subgoal modeling example. The expert policy in (a) targeting the green reward states is reconstructed from the demonstration data set in (b). By generalizing the demonstrations on the intentional level while taking into account the geometry of the problem, the PG subgoal model in (c) yields a significantly improved reconstruction compared to the corresponding action-based model in (d) and the uncorrelated subgoal model in (e). Red color encodes the Hellinger distance to the expert policy.

where $g_x^{(l)}$ denotes the lth Gibbs sample of the subgoal assignment at state x. The obtained reconstruction is visibly better than the one produced by the corresponding imitation learning model in Fig. 5.7d, which interpolates the behavior on the action level and thus fails to navigate the agent around the walls. While the Dirichlet-based subgoal model (Fig. 5.7e) can generally account for the walls through the use of the underlying softmax action model, it cannot propagate the goal information to unvisited states. For the considered uninformative prior distribution over subgoal locations, this has the consequence that actions assigned to such states tend to transport the agent to the center of the environment, as this is the dominating move obtained when blindly averaging over all possible goal locations.

5.4.2 *Transition Model Estimation and Decision-Making in a Grid World Example*

Having focused our attention on learning a model of an observed policy, we now enter the realm of BRL and optimize a behavioral model to the particular dynamics of a given environment and focus on the setting described in Section 5.3.2. For this purpose, we

slightly modify our grid world from Section 5.4.1.1 by placing a target reward of $+1$ in one corner and repositioning the agent to the opposite corner whenever the target state is reached (compare "Grid10" domain in [100]). For the experiment, we assume that the agent is aware of the target reward but does not know the transition dynamics of the environment.

5.4.2.1 *System Identification*

For the beginning, we ignore the reward mechanism altogether and focus on learning the transition dynamics of the environment. To this end, we let the agent perform a random walk on the grid, choosing actions uniformly at random and observing the resulting state transitions as in Eq. (5.3.22). Analogously to the previous two experiments, we estimate the transition dynamics of the environment, with $n = 100$ states and $m = 4$ actions, from the count data as in Eq. (5.3.23) using an independent Dirichlet prior model and our PG framework, where we employ a separate model for each action. Hence, we model the correlation for the next state distribution for a given action $u \in \mathcal{U}$ using a prior covariance over last states $\mathbf{\Sigma}(u) \in \mathbb{R}^{n \times n}$ and prior mean vectors $\mu_k(u) \in \mathbb{R}^n$, for each category $k \in \{1, \dots, n-1\}$ corresponding to the next states.

The graphs describe the resulting estimation accuracy in Fig. 5.8a, and show the distance between the ground truth dynamics of the environment and those predicted by the models, averaged over all states and actions. As expected, our PG model significantly outperforms the naive Dirichlet approach.

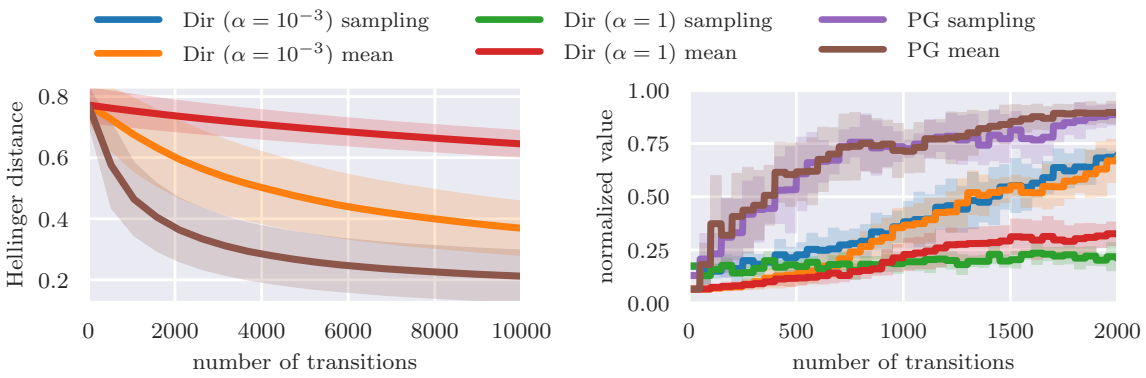
5.4.2.2 *Bayesian Reinforcement Learning*

Next, we consider the problem of combined model-learning and decision-making by exploiting the experience gathered from previous system interactions to optimize future behavior. For this, we leverage BRL as described in Section 5.3.2 and evaluate the grid world experiments for the two PSRL variants as described in Section 5.3.2, PSRL Sampling Variant and Section 5.3.2, PSRL Mean Variant. In both cases, the obtained policy is followed for a fixed number of transitions before new observations are taken into account for updating the posterior distribution.

Figure 5.8b shows the expected returns of the so-obtained policies over the entire execution period for the three prior models evaluated in Fig. 5.8a and both PSRL variants. The graphs reveal that the PG approach requires significantly fewer transitions to learn an effective decision-making strategy.

5.4.2.3 *Bayesian Reinforcement Learning for Queueing Networks*

As a final experiment, we evaluate our model on a network scheduling problem, depicted in Fig. 5.9a. The considered two-server network consists of two queues with buffer lengths $B_1 = B_2 = 10$. The state of the system is determined by the number of packets in each queue, summarized by the queueing vector $\mathbf{b} = [b_1, b_2]^\top$, where b_i denotes the number of



(a) Transition model error.

(b) Model-based BRL.

FIGURE 5.8: BRL results. (a) Estimation error of the transition dynamics over the number of observed transitions. Shown are the Hellinger distances to the true next-state distribution and the standard deviation of the estimation error, both averaged over all states and actions of the MDP. (b) Expected returns of the learned policies (normalized by the optimal return) when replanning with the estimated transition dynamics after every fiftieth state transition.

packets in queue i. The underlying system state space is $\mathcal{X} = \{0, \dots, B_1\} \times \{0, \dots, B_2\}$ with size $n = (B_1 + 1)(B_2 + 1)$.

For our experiment, we consider a system with batch arrivals and batch servicing. The task for the agent is to schedule the network's traffic flow under the condition that only one of the queues can be processed at a time. Accordingly, the actions are encoded as $u = 1$ for serving queue 1 and $u = 2$ for serving queue 2. The number of packets arriving at queue 1 is modeled as

$$q_1 \sim \text{Pois}(q_1 \mid \vartheta_1),$$

with mean rate $\vartheta_1 = 1$. The packets are transferred to buffer 1 and subsequently processed in batches of random size

$$q_2 \sim \text{Pois}(q_2 \mid \vartheta_2),$$

provided that the agent selects queue 1. Therefore, $\vartheta_2 = \beta_1 \, \mathbb{1}(u = 1)$, where we consider an average batch size of $\beta_1 = 3$. Processed packets are transferred to the second queue, where they wait to be processed further in batches of size

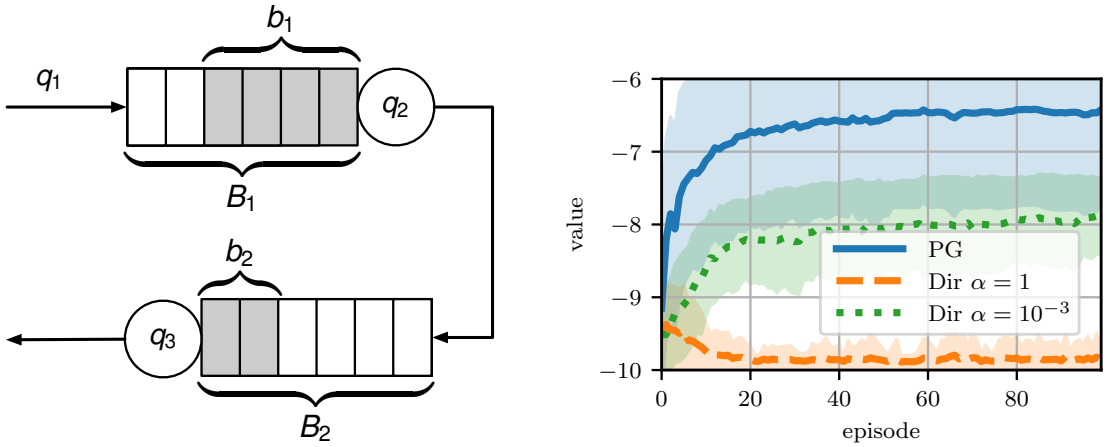
$$q_3 \sim \text{Pois}(q_3 \mid \vartheta_3),$$

with $\vartheta_3 = \beta_2 \, \mathbb{1}(u = 2)$ and an average batch size of $\beta_2 = 2$. The resulting transition to the new queueing state $\mathbf{b}'$ after one processing step can be compactly written as

$$\mathbf{b}' = \begin{bmatrix} (b_1 + q_1 - q_2)_0^{B_1} \\ (b_2 + q_2 - q_3)_0^{B_2} \end{bmatrix},$$

where the truncation operation $(\cdot)_0^B = \max(0, \min(B, \cdot))$ accounts for the nonnegativity and finiteness of the buffers. The reward function, which is known to the agent, computes the negative sum of the queue lengths

$$R(\mathbf{b}) = -(b_1 + b_2).$$



(a) Queueing network.

(b) Comparison of learned policies.

FIGURE 5.9: BRL for batch queueing. (a) Considered two-server queueing network. (b) Expected returns over the number of learning episodes, each consisting of twenty state transitions.

Despite the simplistic architecture of the network, finding an optimal policy for this problem is challenging since determining the state transition matrices requires nontrivial calculations involving concatenations of Poisson distributions. More importantly, when applied in a real-world context, the arrival and processing rates of the network are typically unknown, so that planning-based methods cannot be used.

Figure 5.9b shows the evaluation of PSRL on the network. As in the previous experiment, we use a separate PG model for each action and compute the covariance matrix Σ_ϕ based on the normalized Euclidean distances between the queueing states of the system. This encodes our prior knowledge that the queue lengths obtained after servicing two independent copies of the network tend to be similar if their previous buffer states were similar. Our agent follows a greedy strategy w.r.t. the posterior mean of the estimated model, see Section 5.3.2, Greedy Strategy. After each policy update, the policy is evaluated by performing one thousand steps from all possible queueing states of the system. As the graphs reveal, the PG approach significantly outperforms its correlation agnostic counterpart, requiring fewer interactions with the system while yielding better scheduling strategies by generalizing the networks' dynamics over queueing states.

5.5 SUMMARY

In this chapter, we presented a correlation model for decision-making under model uncertainty. Here, compared to situations with continuous state spaces, as in Chapter 4, we specifically considered discrete state-action spaces. Since a tractable framework for modeling and inferring a correlated model in decision-making was missing, we introduced a new method. With the proposed variational PG model, we have presented a self-contained learning framework for flexible use in many familiar decision-making contexts. The framework allows an intuitive consideration of prior knowledge about the behavior of an agent and the structures of its environment, which can significantly boost the predictive performance of the resulting models by leveraging correlations and

reoccurring patterns in the decision-making process. A key feature is the adjustment of the model regularization through automatic calibration of its hyper-parameters to the specific decision-making scenario at hand, which provides a built-in solution to infer the effective range of correlations from the data. We have evaluated the framework on various benchmark tasks, including an example from communication systems. The framework lends itself nicely to, e.g., realistic queueing problems, which in a real-world situation admit no planning-based solution due to unknown system parameters. In all presented scenarios, our framework consistently outperformed the naive baseline methods, which neglect the rich statistical relationships to be unraveled in the estimation problems.

6

OPTIMAL DECISION-MAKING IN CONTINUOUS TIME AND DISCRETE SPACES WITH BAYESIAN STATE INFERENCE

In this chapter, we introduce compared to Chapters 4 and 5 another aspect of complexity to the up until now discussed decision-making problems. We consider problems under partial observability and therefore close the circle to the inference problem discussed in Chapter 3. A famous model for making decisions under partial observability is the partially observable Markov decision process (POMDP) framework [112]. Here, we extend an MDP by an additional noisy observation process, on which decisions have to be based. We will discuss in this chapter specifically a problem in continuous time, similar to Chapter 3. For this, we take a look at processes that evolve in discrete space and continuous time. Such processes are omnipresent in many areas, as vast as, e.g., discrete event systems in engineering or population dynamics in biology. To be precise, we consider the problem of optimal decision-making in such discrete state and action space systems under partial observability. This places our work at the intersection of optimal filtering and OC. At the current state of research, a mathematical description for simultaneous decision-making and filtering in continuous time with finite state and action spaces is still missing. In this chapter, we give a mathematical description of a continuous-time POMDP. By leveraging optimal filtering theory, we derive a HJB type equation that characterizes the optimal solution. Using techniques from deep learning [113] we approximately solve the resulting partial integrodifferential equation. We present (i) an approach solving the decision problem offline by learning an approximation of the value function and (ii) an online algorithm that provides a solution in belief space using deep RL. We show the applicability on a set of toy examples which pave the way for future methods providing solutions for high dimensional problems. This chapter extends and contains parts and material from the published work [1].

Background

Continuous-time models have extensively been studied in machine learning and control. They are especially beneficial to reason about latent variables at time points that are not included in the data. In a broad range of topics such as natural language processing [114], social media dynamics [115] or biology [116] to name just a few, the underlying process naturally evolves continuously in time. In many applications, the control of such time-continuous models is of interest. There are already numerous approaches that tackle the control problem of continuous state space systems; however, for many processes, a discrete state-space formulation is more suited. This class of systems is discussed in the area of discrete event systems [117]. Decision-making in these systems has a long history, yet, if the state is not fully observed, acting optimally in such scenarios is notoriously hard. Many approaches resort to heuristics, such as applying a separation principle between inference and control. Unfortunately, this can lead to weak performance as the agent does not incorporate the effects of its decisions for future inference.

In the past, this problem was also approached by using a discrete-time formulation such as a POMDP model [112]. Nevertheless, it is not always straightforward to discretize the problem as it requires adding pseudo observations for time points without observations. Additionally, time discretization can lead to problems when learning optimal controllers in the continuous-time setting [118].

A more principled way to approach this problem is to define the model in continuous time with a proper observation model and to solve the resulting model formulation. Still, it is not clear a priori how to design such a model and even less how to control it optimally. In this chapter, we formulate this problem by introducing a continuous-time analog to the POMDP framework. We also show how deep learning methods can be used to find approximate solutions for control under the continuous-time assumption. Our work can be seen as providing the first step towards scalable control laws for high-dimensional problems by using further approximation methods. The accompanying code is publicly available via Git.[1]

Related Work

Historically, OC in continuous time and space is a classical problem and has been addressed ever since the early works of Pontryagin [32] and Kalman [119]. Continuous-time RL formulations have been studied [120–122] and resulted in works such as the advantage updating and advantage learning algorithms [123, 124] and more recently in function approximation methods [118]. Continuous-time formulations in the case of full observability and discrete spaces are regarded in the context of semi-Markov decision processes (SMDPs) [27, 125], with applications, e.g., in E-commerce [126] or as part of the options framework [127, 128].

1 https://github.com/bastianalt/pomdps_continuous_time_discrete_spaces

A critical area for applying time-continuous discrete-state space models is discussed in the queueing theory literature [36]. Here, the state space describes the dynamics of the discrete number of customers in the queue; see also Section 5.4.2.3 in the previous chapter for an analog discrete-time problem. These models are used, for example, for internet communication protocols, such as in TCP congestion control. More generally, the topic is also considered within the area of stochastic hybrid systems [129], where mixtures of continuous and discrete state spaces are discussed. Though, the control under partial observability is often only considered under very specific assumptions on the dynamics.

A classic example for simultaneous optimal filtering and control in continuous space is the LQG controller [13]. In the case of partial observability and discrete spaces, the widely used POMDP framework [112] builds a sound foundation for optimal decision-making in discrete-time and is the basis of many modern methods, e.g., [130–132]. By applying discretizations, it was also used to solve continuous state space problems as discussed in [133]. Another existing framework close to our work is the partially observable semi-Markov decision process (POSMDP) [134], which has applications in fields such as robotics [135] or maintenance [136]. One major drawback of this framework is that observations are only allowed to occur at time points where the latent CTMC jumps. This assumption is very limiting, as in many applications, observations are received at arbitrary time points, or knowledge about jump times is not available.

The development of numerical solution methods for high dimensional PDEs, such as the HJB equation, is an ongoing research topic. Popular approaches include techniques based on linearization such as differential dynamic programming [137, 138], stochastic simulation techniques as in the path integral formalism ,[139, 140] and collocation-based approaches as in [141]. Latter has been extensively discussed due to the recent advances of function approximation by neural networks, which have achieved substantial empirical success [142]. Approaches to solving HJB equations among other PDEs using deep learning can be found in [143–146].

6.1 A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS MODEL IN CONTINUOUS TIME

In this chapter, we now present a model formulation for a continuous-time equivalent of a partially observable Markov decision process (POMDP) model with time index $t \in \mathbb{R}_{\geq 0}$ defined by the tuple

$$\left(\mathcal{X}, \mathcal{U}, \mathcal{Y}, \Lambda, \mathsf{P}_{Y|X,u}, R, \tau\right).$$

We assume a finite state space

$$\mathcal{X} = \{x^{(i)} \mid i = 1, \ldots, n\}$$

and a finite action space

$$\mathcal{U} = \{u^{(i)} \mid i = 1, \ldots, m\},$$

with n distinct states and m distinct actions, respectively. The observation space $\mathcal{Y}$ can be either a discrete space or an uncountable space. For the model we assume that a latent

controlled Markov process $\{x(t) \mid t \in \mathbb{R}_{\geq 0}\}$, with $x(t) \in \mathcal{X}$, follows a CTMC with rate function $\Lambda : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$; i.e.,

$$\mathsf{P}(X(t+h) = x \mid X(t) = x', u(t) = u) = \Lambda(x', x \mid u)h + o(h), \tag{6.1.1}$$

for all $x \neq x'$ and the exit rate is defined as

$$\Lambda(x \mid u) \coloneqq \sum_{x' \in \mathcal{X} \setminus \{x\}} \Lambda(x, x' \mid u).$$

Note that the rate function Λ is a function of the action variable, therefore, the system dynamics are described by a time inhomogeneous CTMC, which is modulated by an action signal $\{u(t) \mid t \in \mathbb{R}_{\geq 0}\}$, with $u(t) \in \mathcal{U}$. We consider a partial observable setting, where the underlying process $\{x(t) \mid t \in \mathbb{R}_{\geq 0}\}$ cannot be directly observed, but an observation process $\{\mathbf{y}(t) \mid t \in \mathbb{R}_{\geq 0}\}$ is available providing information about $\{x(t) \mid t \in \mathbb{R}_{\geq 0}\}$. The observation model is specified by the conditional probability measure $\mathsf{P}_{Y|X,u}$. This observation model specifies either a discrete-time measurement model for a set of N observations $\{\mathbf{y}_i \mid i = 1, \dots, N\}$ as

$$\mathbf{y}_i \sim p(\mathbf{y}_i \mid x(t_i), u(t_i)) \quad \forall i \in \{1, \dots, N\}, \tag{6.1.2}$$

where $\mathbf{y}_i \in \mathcal{Y}$ is the ith observation at time point t_i; i.e., $\mathbf{y}_i \coloneqq \mathbf{y}(t_i)$. Alternatively, $\mathsf{P}_{Y|X,u}$ is the underlying probability measure corresponding to a continuous-time process $\{\mathbf{y}(t) \mid t \in \mathbb{R}_{\geq 0}\}$, with $\mathbf{y}(t) \in \mathcal{Y}$, which follows for example the l-dimensional SDE

$$\mathrm{d}\mathbf{y}(t) = \mathbf{g}(x(t), u(t))\,\mathrm{d}t + \mathbf{B}\,\mathrm{d}\mathbf{w}(t), \tag{6.1.3}$$

with $\mathbf{y}(t) \in \mathcal{Y} \subseteq \mathbb{R}^l$, drift function $\mathbf{g} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^l$, dispersion matrix $\mathbf{B} \in \mathbb{R}^{l \times K}$ and standard Brownian motion $\mathbf{w}(t) \in \mathbb{R}^K$. The reward function is given by $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. Throughout, we consider an infinite horizon problem with discount τ. The goal of the agent is to solve the problem

$$\underset{u_{[0,\infty]}}{\text{maximize}} \quad J[u_{[0,\infty]}] = \mathsf{E}\left[\int_0^\infty \frac{1}{\tau} e^{-\frac{s}{\tau}} R(x(s), u(s))\,\mathrm{d}s\right], \tag{6.1.4}$$

over the action trajectory $u_{[0,\infty]} \coloneqq \{u(t) \mid t \in \mathbb{R}_{\geq 0}\}$. Note that, in this chapter, we use a normalization by $1/\tau$ for the value function as it was found to stabilize its learning process when function approximations are used [143].

Since, the agent does not have access to the latent process $x(t)$, the decision $u(t)$ at time point t has to be based on the history of the observation process $\mathbf{y}_{[0,t)} \coloneqq \{\mathbf{y}(s) \mid s \in [0, t)\}$ and the action process $u_{[0,t)} \coloneqq \{u(s) \mid s \in [0, t)\}$. Hence, a sufficient statistic for the latent state $x(t)$ at time t is provided by the filtering distribution $\pi(x, t) \coloneqq \mathsf{P}(X(t) = x \mid \mathbf{y}_{[0,t)}, u_{[0,t)})$. This filtering distribution is used as a state of the partially observable system in form of a vector valued belief state $\boldsymbol{\pi}(t) \in \Delta^n$, with components $\{\pi(x, t) \mid x \in \mathcal{X}\}$. Therefore, we define $\mu : \Delta^n \to \mathcal{U}$ as a deterministic policy, which maps a belief state to an action. The performance of such a policy μ with action $u(s) = \mu(\boldsymbol{\pi}(s))$, $s \in [0, \infty)$ in the infinite horizon case is then given by

$$J[\mu] = \mathsf{E}\left[\int_0^\infty \frac{1}{\tau} e^{-\frac{s}{\tau}} R(x(s), \mu(\boldsymbol{\pi}(s)))\,\mathrm{d}s \;\middle|\; \boldsymbol{\pi}(0) = \boldsymbol{\pi}\right],$$

with initial belief $\boldsymbol{\pi} \in \Delta^n$ and above expectation is carried out w.r.t. both the latent state and observation processes.

6.2 POSTERIOR INFERENCE: A BAYESIAN ESTIMATE OF THE BELIEF STATE

Next, we derive the equations which describe the evolution of the belief state $\{\boldsymbol{\pi}(t) \mid t \in \mathbb{R}_{\geq 0}\}$. For this, we specifically consider (i) a continuous-discrete measurement model as in Eq. (6.1.2) and (ii) a continuous-time measurement model as in Eq. (6.1.3).

THE LIKELIHOOD. For the continuous-discrete case with observations at time points $\{t_i \mid i = 1, \dots, N\}$, the likelihood of the data $\mathcal{D}(t) = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$, with $N = \max(N' : t_{N'} \leq t)$, for a given latent CTMC trajectory $x_{[0,t]} \coloneqq \{x(s) \mid s \in [0,t]\}$ and action trajectory $u_{[0,t]}$ can be expressed as

$$p(\mathcal{D}(t) \mid x_{[0,t]}, u_{[0,t]}) = \prod_{i=1}^{N} p(\mathbf{y}_i \mid x(t_i), u(t_i)). \tag{6.2.5}$$

In the case of a continuous observation trajectory $\mathbf{y}_{[0,t]} \coloneqq \{\mathbf{y}(s) \mid s \in [0,t]\}$ we can not express the likelihood in form of a probability density, as $\mathbf{y}_{[0,t]}$ is a collection of uncountably infinite many RVs. However, for the SDE observation model described in Eq. (6.1.3) we can express the likelihood of the data $\mathcal{D}(t) = \mathbf{y}_{[0,t]}$ as a path measure using Girsanov's theorem (see, e.g., [17, 20]), as

$$\begin{aligned} &\mathsf{P}(\mathcal{D}(t) \in \mathrm{d}\mathbf{y}_{[0,t]} \mid x_{[0,t]}, u_{[0,t]}) \equiv \mathsf{P}(\mathbf{Y}_{[0,t]} \in \mathrm{d}\mathbf{y}_{[0,t]} \mid x_{[0,t]}, u_{[0,t]}) \\ &= \exp\left(\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) - \frac{1}{2} \int_0^t \mathbf{g}^\top(x(s), u(s)) \right. \\ &\qquad \left. \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathbf{g}(x(s), u(s)) \, \mathrm{d}s \right) \mathsf{P}(\mathbf{W}_{[0,t]} \in \mathrm{d}\mathbf{y}_{[0,t]}), \end{aligned} \tag{6.2.6}$$

where $\mathsf{P}(\mathbf{W}_{[0,t]} \in \cdot)$ denotes the Wiener measure w.r.t. the Brownian motion $\mathbf{w}_{[0,t]} \coloneqq \{\mathbf{w}(s) \mid s \in [0,t]\}$ in Eq. (6.1.3).

THE PRIOR DISTRIBUTION. The prior distribution is given by the controlled CTMC as defined by the infinitesimal definition in Eq. (6.1.1). Given this, we give now a derivation for the *master equation* [19] of the controlled process $\{x(t) \mid u_{[0,t]}\}$.

For the derivation of the master equation, we start with the Chapman-Kolmogorov equation (see, e.g., [19, 20]) by exploiting the law of total probability

$$\begin{aligned} \mathsf{P}(X(t+h) = x \mid u_{[0,t+h]}) = \sum_{x' \in \mathcal{X}} &\mathsf{P}(X(t+h) = x \mid X(t) = x', u_{[t,t+h]}) \\ &\cdot \mathsf{P}(X(t) = x' \mid u_{[0,t]}), \end{aligned}$$

for some $h > 0$. Next, we compute the relative change in probability as

$$\begin{aligned} &\frac{\mathsf{P}(X(t+h) = x \mid u_{[0,t+h]}) - \mathsf{P}(X(t) = x \mid u_{[0,t]})}{h} \\ &= \frac{1}{h} \left\{ \sum_{x' \in \mathcal{X} \setminus \{x\}} \mathsf{P}(X(t+h) = x \mid X(t) = x', u_{[t,t+h]}) \, \mathsf{P}(X(t) = x' \mid u_{[0,t]}) \right. \\ &\quad \left. + \mathsf{P}(X(t) = x \mid u_{[0,t]}) \left(\mathsf{P}(X(t+h) = x \mid X(t) = x, u_{[t,t+h]}) - 1 \right) \right\}. \end{aligned} \tag{6.2.7}$$

Using the complement rule $\mathsf{P}(X(t+h) = x \mid X(t) = x, u_{[t,t+h]}) = 1 - \sum_{x' \in \mathcal{X} \setminus \{x\}} \mathsf{P}(X(t+h) = x' \mid X(t) = x, u_{[t,t+h]})$ for the transition probabilities yields for Eq. (6.2.7)

$$\begin{aligned}
&\frac{\mathsf{P}(X(t+h) = x \mid u_{[0,t+h]}) - \mathsf{P}(X(t) = x \mid u_{[0,t]})}{h} \\
&= \frac{1}{h} \left\{ \sum_{x' \in \mathcal{X} \setminus \{x\}} \mathsf{P}(X(t+h) = x \mid X(t) = x', u_{[t,t+h]}) \, \mathsf{P}(X(t) = x' \mid u_{[0,t]}) \right. \\
&\quad \left. - \sum_{x' \in \mathcal{X} \setminus \{x\}} \mathsf{P}(X(t+h) = x' \mid X(t) = x, u_{[t,t+h]}) \, \mathsf{P}(X(t) = x \mid u_{[0,t]}) \right\}.
\end{aligned}$$

By exploiting the infinitesimal definition in Eq. (6.1.1) and taking the limit $\lim_{h \to 0}$ from both sides, we arrive at

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \mathsf{P}(X(t) = x \mid u_{[0,t]}) = &\sum_{x' \in \mathcal{X} \setminus \{x\}} \Lambda(x', x \mid u(t)) \, \mathsf{P}(X(t) = x' \mid u_{[0,t]}) \\
&- \sum_{x' \in \mathcal{X} \setminus \{x\}} \Lambda(x, x' \mid u(t)) \, \mathsf{P}(X(t) = x \mid u_{[0,t]}).
\end{aligned}$$

Finally, let us define $\Lambda(x, x \mid u) \coloneqq -\Lambda(x \mid u) \equiv -\sum_{x' \in \mathcal{X} \setminus \{x\}} \Lambda(x, x' \mid u)$ and $p(x,t) \coloneqq \mathsf{P}(X(t) = x \mid u_{[0,t]})$. Therefore, we have the *master equation*

$$\frac{\mathrm{d}p(x,t)}{\mathrm{d}t} = \sum_{x' \in \mathcal{X}} \Lambda(x', x \mid u(t)) p(x', t) \quad \forall x \in \mathcal{X}, \tag{6.2.8}$$

with initial condition $p(x, 0) = \mathsf{P}(X(0) = x)$ for every $x \in \mathcal{X}$.

Given the prior distribution and the likelihoods, we are now ready to compute the posterior probabilities for the filtering distribution.

6.2.1 *Continuous-Discrete Filtering*

We start with the case of continuous-discrete filtering, as in [116]. In continuous-discrete filtering, we consider the observation model from Eq. (6.2.5) and prior model from Eq. (6.2.8). Here, we can find an evolution equation for the filtering distribution $\{\pi(x,t) \mid (x,t) \in \mathcal{X} \times \mathbb{R}_{\geq 0}\}$, with

$$\pi(x,t) = \mathsf{P}(X(t) = x \mid \mathbf{y}_{[0,t]}, u_{[0,t]}) = \mathsf{P}(X(t) = x \mid \mathbf{y}_1, \ldots, \mathbf{y}_N, u_{[0,t]}).$$

For the calculation, we consider two cases. First, we compute the filtering distribution in between observations. In this case, we consider an arbitrary interval $[t, t+h]$, where no observations occur. Here, we have

$$\begin{aligned}\pi(x, t+h) &= \mathsf{P}(X(t+h) = x \mid \mathbf{y}_1, \dots, \mathbf{y}_N, u_{[0,t+h]}) \\ &= \sum_{x' \in \mathcal{X}} \mathsf{P}(X(t+h) = x, X(t) = x' \mid \mathbf{y}_1, \dots, \mathbf{y}_N, u_{[0,t+h]}) \\ &= \sum_{x' \in \mathcal{X}} \mathsf{P}(X(t+h) = x \mid X(t) = x', \mathbf{y}_1, \dots, \mathbf{y}_N, u_{[0,t+h]}) \\ &\qquad \cdot \mathsf{P}(X(t) = x' \mid \mathbf{y}_1, \dots, \mathbf{y}_N, u_{[0,t+h]}) \\ &= \sum_{x' \in \mathcal{X}} \mathsf{P}(X(t+h) = x \mid X(t) = x', u_{[t,t+h]}) \pi(x', t).\end{aligned}$$

Therefore, $\pi(x, t+h)$ in between observations is given by the Chapman-Kolmogorov equation with the transition probabilities $\mathsf{P}(X(t+h) = x \mid X(t) = x', u_{[t,t+h]})$, which correspond to those of the prior dynamics, as given in the definition in Eq. (6.1.1). Hence, analogous to the derivation in Section 6.2, The Prior Distribution, we can compute the filtering distribution in between observations, which follows the *master equation*

$$\frac{\mathrm{d}\pi(x,t)}{\mathrm{d}t} = \sum_{x' \in \mathcal{X}} \Lambda(x', x \mid u(t)) \pi(x', t) \quad \forall x \in \mathcal{X}, \tag{6.2.9}$$

with initial condition $\pi(x, 0) = \mathsf{P}(X(0) = x)$ for every $x \in \mathcal{X}$.

Second, for the case at observation time points $\{t_i \mid i = 1, \dots, N\}$ the filtering distribution computes to

$$\begin{aligned}\pi(x, t_i) &= \mathsf{P}(X(t_i) = x \mid \mathbf{y}_1, \dots, \mathbf{y}_i, u_{[0,t_i]}) \\ &\equiv p(x, t_i \mid \mathbf{y}_1, \dots, \mathbf{y}_i, u_{[0,t_i]}) \\ &= \frac{p(x, t_i, \mathbf{y}_1, \dots, \mathbf{y}_i \mid u_{[0,t_i]})}{p(\mathbf{y}_1, \dots, \mathbf{y}_i \mid u_{[0,t_i]})} \\ &= \frac{p(\mathbf{y}_i \mid x, t_i, \mathbf{y}_1, \dots, \mathbf{y}_{i-1}, u_{[0,t_i]}) p(x, t_i, \mathbf{y}_1, \dots, \mathbf{y}_{i-1} \mid u_{[0,t_i]})}{p(\mathbf{y}_1, \dots, \mathbf{y}_i \mid u_{[0,t_i]})} \\ &= \frac{p(\mathbf{y}_i \mid x, t_i, \mathbf{y}_1, \dots, \mathbf{y}_{i-1}, u_{[0,t_i]}) p(x, t_i \mid \mathbf{y}_1, \dots, \mathbf{y}_{i-1}, u_{[0,t_i]})}{p(\mathbf{y}_1, \dots, \mathbf{y}_i \mid u_{[0,t_i]})} \\ &\qquad \cdot p(\mathbf{y}_1, \dots, \mathbf{y}_{i-1} \mid u_{[0,t_i]}) \\ &= \frac{p(\mathbf{y}_i \mid x, u(t_i)) \pi(x, t_i^-)}{\sum_{x' \in \mathcal{X}} p(\mathbf{y}_i \mid x', u(t_i)) \pi(x', t_i^-)},\end{aligned} \tag{6.2.10}$$

where $\pi(x, t_i^-) \coloneqq p(x, t_i \mid \mathbf{y}_1, \dots, \mathbf{y}_{i-1}, u_{[0,t_i]})$ denotes the belief just before the observation. The reset conditions at the observation time points represent the posterior distribution of $x(t)$ which combines the likelihood model $p(y \mid x, u)$ with the previous filtering distribution $\pi(x, t_i^-)$ as prior. Note that the filter equations in Eqs. (6.2.9) and (6.2.10) can be written in the differential form as

$$\mathrm{d}\pi(x,t) = \sum_{x' \in \mathcal{X}} \Lambda(x', x \mid u(t)) \pi(x', t) \, \mathrm{d}t + \big(\pi(x, t_{N(t)}) - \pi(x,t)\big) \, \mathrm{d}N(t). \tag{6.2.11}$$

Here, the observations enter the filtering equation by the counting process $N(t) = \sum_{i=1}^{N} \mathbb{1}(t_i \leq t)$ and the jump amplitude $\pi(x, t_{N(t)}) - \pi(x, t)$, which sets the filter to the new posterior distribution $\pi(x, t_{N(t)})$, as in Eq. (6.2.10).

6.2.2 *The Wonham Filter*

Next, we derive an evolution equation for the so-called Wonham filter [147]. For the Wonham filter, we consider the continuous-time observation model from Eq. (6.1.3) and the CTMC prior model from Eq. (6.2.8). Here, the Wonham filter describes the time evolution of filtering distribution $\{\pi(x,t) \mid (x,t) \in \mathcal{X} \times \mathbb{R}_{\geq 0}\}$, with

$$\pi(x,t) = \mathsf{P}(X(t) = x \mid \mathbf{y}_{[0,t]}, u_{[0,t]}).$$

Further, we give an outline of the derivation, which largely follows the one discussed in [20, Chapter 18.5], for additional details, see [147, 148].

We start by computing the Bayes rule for the posterior path measure using the likelihood in Eq. (6.2.6), this yields

$$\begin{aligned}
&\mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid \mathbf{y}_{[0,t]}, u_{[0,t]}) \\
&\propto \mathsf{P}\big(\big(X_{[0,t]}, \mathbf{Y}_{[0,t]}\big) \in \mathrm{d}\left(x_{[0,t]}, \mathbf{y}_{[0,t]}\right) \mid u_{[0,t]}\big) \\
&= \mathsf{P}(\mathbf{Y}_{[0,t]} \in \mathrm{d}\mathbf{y}_{[0,t]} \mid x_{[0,t]}, u_{[0,t]})\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]}) \\
&= \exp\left(\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) - \frac{1}{2}\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \right. \\
&\qquad \left. \mathbf{g}(x(s), u(s))\, \mathrm{d}s\right) \mathsf{P}(\mathbf{W}_{[0,t]} \in \mathrm{d}\mathbf{y}_{[0,t]})\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]}) \\
&\propto \exp\left(\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) - \frac{1}{2}\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \right. \\
&\qquad \left. \mathbf{g}(x(s), u(s))\, \mathrm{d}s\right) \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]}).
\end{aligned}$$

Therefore, the posterior path measure can be written as

$$\mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid \mathbf{y}_{[0,t]}, u_{[0,t]}) = \frac{Z(x_{[0,t]}, y_{[0,t]})\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]})}{\int Z(x'_{[0,t]}, y_{[0,t]})\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x'_{[0,t]} \mid u_{[0,t]})},$$

which is called Kallianpur-Striebel formula. Here, $Z(x_{[0,t]}, y_{[0,t]})$ is the Radon-Nikodym derivative between the posterior and prior path measure; i.e.,

$$\begin{aligned}
Z(x_{[0,t]}, y_{[0,t]}) \equiv \frac{\mathrm{d}\,\mathsf{P}^{X|Y}}{\mathrm{d}\,\mathsf{P}^{X}} = \exp\Bigg(&\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) \\
&-\frac{1}{2}\int_0^t \mathbf{g}^\top(x(s), u(s)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathbf{g}(x(s), u(s))\, \mathrm{d}s\Bigg).
\end{aligned}$$

Further, the filtering distribution is given as the t-point marginal of this path measure; i.e.,

$$\begin{aligned}
\pi(x,t) &= \mathsf{P}(X(t) = x \mid \mathbf{y}_{[0,t]}, u_{[0,t]}) \\
&= \mathsf{E}\big[\mathbb{1}(x(t) = x) \;\big|\; \mathbf{y}_{[0,t]}, u_{[0,t]}\big] \\
&= \int \mathbb{1}(x(t) = x)\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid \mathbf{y}_{[0,t]}, u_{[0,t]})
\end{aligned}$$

Next, for further derivation, we define an analog equation for the unnormalized filter as

$$\tilde{\pi}(x,t) = \int \mathbb{1}(x(t)=x) Z(x_{[0,t]}, y_{[0,t]})\, \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]}), \tag{6.2.12}$$

where $\pi(x,t) = \frac{\tilde{\pi}(x,t)}{\sum_{x' \in \mathcal{X}} \tilde{\pi}(x',t)}$.

Computing $\mathrm{d}\tilde{\pi}(x,t)$ from Eq. (6.2.12) by the use of Itô calculus gives the Zakai equation

$$\mathrm{d}\tilde{\pi}(x,t) = \sum_{x' \in \mathcal{X}} \Lambda(x', x \mid u(t)) \tilde{\pi}(x',t)\, \mathrm{d}t + \tilde{\pi}(x,t) \mathbf{g}^\top(x, u(t)) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(t),$$

with initial condition $\tilde{\pi}(x,0) = p(x,0) = \mathsf{P}(X(0)=x)$ for every $x \in \mathcal{X}$..

Further, by computing $\mathrm{d}\log \sum_{x' \in \mathcal{X}} \tilde{\pi}(x',t)$ with Itô calculus, we have

$$\mathrm{d}\log \sum_{x' \in \mathcal{X}} \tilde{\pi}(x',t) = \int_0^t \bar{\mathbf{g}}^\top(u(s),s) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) - \frac{1}{2} \int_0^t \bar{\mathbf{g}}^\top(u(s),s) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \bar{\mathbf{g}}(u(s),s)\, \mathrm{d}s,$$

with $\bar{\mathbf{g}}(u(t),t) = \sum_{x' \in \mathcal{X}} \mathbf{g}(x', u(t)) \pi(x',t)$. This gives an explicit expression for the normalizer as

$$\sum_{x' \in \mathcal{X}} \tilde{\pi}(x',t) = \exp\left(\int_0^t \bar{\mathbf{g}}^\top(u(s),s) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \mathrm{d}\mathbf{y}(s) - \frac{1}{2} \int_0^t \bar{\mathbf{g}}^\top(u(s),s) \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \bar{\mathbf{g}}(u(s),s)\, \mathrm{d}s \right). \tag{6.2.13}$$

Hence, Eqs. (6.2.12) and (6.2.13) yield for the normalized probabilities

$$\begin{aligned}
\pi(x,t) &= \frac{\tilde{\pi}(x,t)}{\sum_{x' \in \mathcal{X}} \tilde{\pi}(x',t)} \\
&= \int \mathbb{1}(x(t)=x) \exp\left(\int_0^t \left(\mathbf{g}(x(s),u(s)) - \bar{\mathbf{g}}(u(s),s)\right)^\top \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \right. \\
&\qquad [\mathrm{d}\mathbf{y}(s) - \bar{\mathbf{g}}(u(s),s)\, \mathrm{d}s] - \frac{1}{2} \int_0^t \left(\mathbf{g}(x(s),u(s)) - \bar{\mathbf{g}}(u(s),s)\right)^\top \\
&\left. \left(\mathbf{B}\mathbf{B}^\top\right)^{-1} \left(\mathbf{g}(x(s),u(s)) - \bar{\mathbf{g}}(u(s),s)\right)\, \mathrm{d}s \right) \mathsf{P}(X_{[0,t]} \in \mathrm{d}x_{[0,t]} \mid u_{[0,t]}).
\end{aligned}$$

Finally, by computing $\mathrm{d}\pi(x,t)$ with Itô calculus, we have the controlled Wonham filter or Kushner-Stratonovich equation [17, 20, 149]

$$\begin{aligned}
\mathrm{d}\pi(x,t) &= \sum_{x' \in \mathcal{X}} \Lambda(x', x \mid u(t)) \pi(x',t)\, \mathrm{d}t \\
&\quad + \pi(x,t) \left(\mathbf{g}(x,u(t)) - \bar{\mathbf{g}}(u(t),t)\right)^\top (\mathbf{B}\mathbf{B}^\top)^{-1} (\mathrm{d}\mathbf{y}(t) - \bar{\mathbf{g}}(u(t),t)\, \mathrm{d}t).
\end{aligned} \tag{6.2.14}$$

Note that we denote the filtering equation as the controlled Wonham filter as the observation and latent processes are modulated by the control signal $u(t)$.

Analog to the presented filters, other optimal filters can be derived. Note that the dynamical laws of these filters are often in the class of jump-diffusion processes, see [21] for an introduction and the control thereof. This includes also the presented filters in Eqs. (6.2.11) and (6.2.14), where for the Wonham filter case in Eq. (6.2.14) we have a jump diffusion process without a jump part and for the continuous-discrete case in Eq. (6.2.11) we have a filter which follows a jump diffusion process without a diffusion part. Another important observation is that for the case of finite-state spaces, the filtering distribution is described by a finite-dimensional object, opposed to the case of, e.g., uncountable state spaces, where the filtering distribution is characterized by a probability density, which is infinite-dimensional. This is helpful as the filtering distribution can be used as a finite-dimensional state in the POMDP setting.

6.3 OPTIMAL DECISION-MAKING IN CONTINUOUS TIME

Given the description of how to compute the filtering distribution, we are now ready to discuss decision-making. For this, we start by describing a generative process for drawing trajectories from a continuous-time POMDP.

6.3.1 *Simulating the Model*

For drawing trajectories, we consider the stochastic simulation with given policy μ, which maps a belief $\boldsymbol{\pi}$ to action u. We start at time t with state $x(t) = x'$, and belief $\boldsymbol{\pi}(t) \in \Delta^n$. The first step is to draw a waiting time ξ of the latent CTMC after which the state switches. This CTMC is time inhomogeneous with the exit rate $\Lambda(x' \mid \mu(\boldsymbol{\pi}(t)))$ since the action modulates the transition function. Therefore, we sample ξ from the time-dependent exponential distribution with CDF

$$P(\xi) = 1 - \exp\left(\int_t^{t+\xi} \Lambda(x' \mid \mu(\boldsymbol{\pi}(s)))\,\mathrm{d}s\right)$$

for which one needs to solve the filtering trajectory $\{\boldsymbol{\pi}(s) \mid s \in [t, t+\xi)\}$ using a numeric (stochastic) differential equation solver beforehand. There are multiple ways to sample from time-dependent exponential distributions. A convenient method to jointly calculate the filtering distribution and the latent state trajectory is provided by the thinning algorithm [150]. For an adaptation for the continuous-time POMDP see Fig. 6.1.

As the second step, after having sampled the waiting time ξ, we can draw the next state $x(t+\xi) = x$ given ξ from the categorical distribution

$$\mathsf{P}(X(t+\xi) = x \mid x', \xi) = \begin{cases} \frac{\Lambda(x', x \mid \mu(\boldsymbol{\pi}(t+\xi)))}{\Lambda(x' \mid \mu(\boldsymbol{\pi}(t+\xi)))} & \text{if } x' \neq x, \\ 0 & \text{otherwise} \end{cases}.$$

The described process is executed for the initial belief and state at $t = 0$ and repeated until the total simulation time has been reached.

input : t: time point
x: state of the latent CTMC; i.e., $x(t) = x$
$q_{\max} = \max_{u \in \mathcal{U}} \Lambda(x \mid u)$: maximum exit rate
output : ξ: Waiting time between the start time t and the next jump of $x(t)$

Set current time to start time $T = t$
while *True* **do**
  Sample uniform variable for inverse CDF method $\zeta \sim \mathrm{Uniform}(\zeta \mid 0, 1)$
  Calculate minimum waiting time sample $\xi_{\min} = -\frac{\log \zeta}{q_{\max}}$
  Update current time $T = T + \xi_{\min}$
  Update the filtering dsitribution up to T; i.e., $\boldsymbol{\pi}_{[t,T)}$
  Draw $\zeta' \sim \mathrm{Uniform}(\zeta' \mid 0, 1)$
  if $\zeta' \leq \frac{\Lambda(x|\mu(\boldsymbol{\pi}(T))}{q_{\max}}$ **then**
    Calculate the waiting time $\xi = T - t$
    return *Waiting time ξ and filtering distribution $\boldsymbol{\pi}_{[t,t+\xi)}$*
  end
end

FIGURE 6.1: Thinning algorithm adapted to the continuous-time POMDP setting.

6.3.2 *The Hamilton-Jacobi-Bellman Equation in Belief Space*

Next, we discuss how to compute an optimal policy for the described POMDP. For computing an optimal policy, we derive an equation for the value function, which can be solved to obtain the optimal policy.

First, we observe that the optimal discounted cost from Eq. (6.1.4) can be written as

$$\max_{u_{[0,\infty)}} \mathsf{E}\left[\int_0^\infty \frac{1}{\tau} e^{-\frac{s}{\tau}} R(x(s), u(s))\,\mathrm{d}s\right] = \max_{u_{[t,\infty)}} \mathsf{E}\left[\int_t^\infty \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s\right],$$

using a change in variables in $s - t$. Further, we define the infinite horizon optimal value function as the expected discounted cumulative reward

$$V^*(\boldsymbol{\pi}) \coloneqq \max_{u_{[t,\infty)}} \mathsf{E}\left[\int_t^\infty \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s \;\middle|\; \boldsymbol{\pi}(t) = \boldsymbol{\pi}\right],$$

where the value function depends on the belief state $\boldsymbol{\pi}$ which provides a sufficient statistic for the state. By splitting the integral into two terms from t to $t + h$ and from $t + h$ to ∞, we have

$$\begin{aligned} V^*(\boldsymbol{\pi}) = \max_{u_{[t,\infty)}} \mathsf{E}\Bigg[&\int_t^{t+h} \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s \\ &+ \int_{t+h}^\infty \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s \;\middle|\; \boldsymbol{\pi}(t) = \boldsymbol{\pi}\Bigg] \end{aligned}$$

and by identifying the second integral as $e^{-\frac{h}{\tau}}V^*(\boldsymbol{\pi}(t+h))$, we find the *stochastic principle of optimality* as

$$V^*(\boldsymbol{\pi}) = \max_{u_{[t,t+h)}} \mathsf{E}\left[\int_t^{t+h} \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s + e^{-\frac{h}{\tau}} V^*(\boldsymbol{\pi}(t+h)) \;\middle|\; \boldsymbol{\pi}(t) = \boldsymbol{\pi}\right]. \tag{6.3.15}$$

Here, we consider the class of filtering distributions that follow a jump-diffusion process

$$\mathrm{d}\boldsymbol{\pi}(t) = \mathbf{f}(\boldsymbol{\pi}(t), u(t))\,\mathrm{d}t + \mathbf{G}(\boldsymbol{\pi}(t), u(t))\,\mathrm{d}\mathbf{w}(t) + \mathbf{h}(\boldsymbol{\pi}(t), u(t))\,\mathrm{d}N(t), \tag{6.3.16}$$

where $\mathbf{f} : \Delta^n \times \mathcal{U} \to \mathbb{R}^n$ denotes the drift function, $\mathbf{G} : \Delta^n \times \mathcal{U} \to \mathbb{R}^{n\times K}$ the dispersion matrix, with $\mathbf{w}(t) \in \mathbb{R}^K$ being an K dimensional standard Brownian motion and $\mathbf{h} : \Delta^n \times \mathcal{U} \to \mathbb{R}^n$ denotes the jump amplitude. We assume a Poisson counting process $N(t) \sim \mathcal{PP}(N(t) \mid \lambda)$ with rate λ for the observation times $\{t_i\}_{i\in\mathbb{N}}$, which implies that $t_i - t_{i-1} \sim \mathrm{Exp}(t_i - t_{i-1} \mid \lambda)$.

Under the dynamics in Eq. (6.3.16), we apply Itô's formula for jump-diffusion processes to the value function and find

$$\begin{aligned} V^*(\boldsymbol{\pi}(t+h)) =& V^*(\boldsymbol{\pi}(t)) + \int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi}(s), u(s))\,\mathrm{d}s \\ &+ \int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{G}(\boldsymbol{\pi}(s), u(s))\,\mathrm{d}\mathbf{w}(s) \\ &+ \int_t^{t+h} \frac{1}{2}\,\mathrm{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi}(s), u(s)) \mathbf{G}^\top(\boldsymbol{\pi}(s), u(s))\right)\,\mathrm{d}s \\ &+ \int_t^{t+h} \left[V^*(\boldsymbol{\pi}(s) + \mathbf{h}(\boldsymbol{\pi}(s), u(s))) - V^*(\boldsymbol{\pi}(s))\right]\,\mathrm{d}N(s). \end{aligned} \tag{6.3.17}$$

By inserting Eq. (6.3.17) into Eq. (6.3.15) we find

$$\begin{aligned} V^*(\boldsymbol{\pi}) = \max_{u_{[t,t+h)}} \Bigg\{ \mathsf{E}\Bigg[&\int_t^{t+h} \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s), u(s))\,\mathrm{d}s + e^{-\frac{h}{\tau}} \left(V^*(\boldsymbol{\pi}(t))\right. \\ &+ \int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi}(s), u(s))\,\mathrm{d}s + \int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{G}(\boldsymbol{\pi}(s), u(s))\,\mathrm{d}\mathbf{w}(s) \\ &+ \int_t^{t+h} \frac{1}{2}\,\mathrm{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi}(s), u(s)) \mathbf{G}^\top(\boldsymbol{\pi}(s), u(s))\right)\,\mathrm{d}s \\ &\left. + \int_t^{t+h} \left[V^*(\boldsymbol{\pi}(s) + \mathbf{h}(\boldsymbol{\pi}(s), u(s)) - V^*(\boldsymbol{\pi}(s))\right]\,\mathrm{d}N(s)\right) \;\middle|\; \boldsymbol{\pi}(t) = \boldsymbol{\pi}\Bigg]\Bigg\}. \end{aligned}$$

Collecting terms in $V^*(\boldsymbol{\pi})$ and dividing both sides by h we get

$$V^*(\boldsymbol{\pi})\frac{1-e^{-\frac{h}{\tau}}}{h} = \max_{u_{[t,t+h)}} \left\{ \mathsf{E}\left[\frac{1}{h}\int_t^{t+h} \frac{1}{\tau} e^{-\frac{s-t}{\tau}} R(x(s),u(s))\,\mathrm{d}s + \frac{e^{-\frac{h}{\tau}}}{h}\,(\right.\right.$$
$$\int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi}(s),u(s))\,\mathrm{d}s + \int_t^{t+h} \frac{\partial V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}} \mathbf{G}(\boldsymbol{\pi}(s),u(s))\,\mathrm{d}\mathbf{w}(s)$$
$$+ \int_t^{t+h} \frac{1}{2}\,\mathrm{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi}(s))}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi}(s),u(s))\mathbf{G}^\top(\boldsymbol{\pi}(s),u(s)) \right)\,\mathrm{d}s$$
$$\left.\left. + \int_t^{t+h} \left[V^*(\boldsymbol{\pi}(s) + \mathbf{h}(\boldsymbol{\pi}(s),u(s)) - V^*(\boldsymbol{\pi}(s)) \right]\,\mathrm{d}N(s) \right) \,\middle|\, \boldsymbol{\pi}(t)=\boldsymbol{\pi} \right] \right\}.$$

Taking $\lim_{h\to 0}$ and calculating the expectation w.r.t. $\mathbf{w}(t)$ and $N(t)$ we find the PDE

$$\frac{1}{\tau}V^*(\boldsymbol{\pi}) = \max_{u\in\mathcal{U}} \left\{ \frac{1}{\tau}\,\mathsf{E}[R(x(t),u) \mid \boldsymbol{\pi}(t)=\boldsymbol{\pi}] + \frac{\partial V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi},u) \right.$$
$$\left. + \frac{1}{2}\,\mathrm{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi},u)\mathbf{G}^\top(\boldsymbol{\pi},u) \right) + \lambda\left(\mathsf{E}[V^*(\boldsymbol{\pi}+\mathbf{h}(\boldsymbol{\pi},u))] - V^*(\boldsymbol{\pi}) \right) \right\},$$

for further details, see [21].

Finally, analogous to the case of stochastic optimal control [13], we have the HJB equation

$$V^*(\boldsymbol{\pi}) = \max_{u\in\mathcal{U}} \left\{ \mathsf{E}[R(x,u) \mid \boldsymbol{\pi}] + \tau \frac{\partial V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi},u) \right.$$
$$\left. + \frac{\tau}{2}\,\mathrm{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi},u)\mathbf{G}(\boldsymbol{\pi},u)^\top \right) + \tau\lambda\left(\mathsf{E}[V^*(\boldsymbol{\pi}+\mathbf{h}(\boldsymbol{\pi},u)] - V^*(\boldsymbol{\pi}) \right) \right\}, \tag{6.3.18}$$

where $\mathsf{E}[R(x,u) \mid \boldsymbol{\pi}] = \sum_{x\in\mathcal{X}} R(x,u)\pi(x)$. Given this general description, we can hence find the optimality condition for the presented filters.

THE HAMILTON-JACOBI-BELLMAN EQUATION FOR THE CONTROLLED CONTINUOUS DISCRETE FILTER. For the controlled continuous discrete filter as in Section 6.2.1, i.e.,

$$\mathrm{d}\pi(x,t) = \sum_{x'\in\mathcal{X}} \Lambda(x',x \mid u(t))\pi(x',t)\,\mathrm{d}t$$
$$+ \left(\frac{p(\mathbf{y}_{N(t)} \mid x,u(t))\pi(x,t)}{\sum_{x'\in\mathcal{X}} p(\mathbf{y}_{N(t)} \mid x',u(t))\pi(x',t)} - \pi(x,t) \right)\,\mathrm{d}N(t),$$

we can compute the HJB equation (6.3.18), by identifying the components $\{f(\boldsymbol{\pi},u,x) \mid x\in\mathcal{X}\}$ of the drift function $\mathbf{f}(\boldsymbol{\pi},u)$ as

$$f(\boldsymbol{\pi},u,x) = \sum_{x'\in\mathcal{X}} \Lambda(x',x \mid u)\pi(x')$$

and the diffusion term is zero; i.e., $\mathbf{G}(\boldsymbol{\pi},u) = \mathbf{0}$. The components $\{h(\boldsymbol{\pi},u,x) \mid x\in\mathcal{X}\}$ of the jump amplitude $\mathbf{h}(\boldsymbol{\pi},u)$ are

$$h(\boldsymbol{\pi},u,x) = \frac{p(\mathbf{y} \mid x,u)\pi(x)}{\sum_{x'\in\mathcal{X}} p(\mathbf{y} \mid x',u)\pi(x')} - \pi(x),$$

with $\mathbf{y} \sim p(\mathbf{y}) = \sum_{x\in\mathcal{X}} p(\mathbf{y} \mid x, u)\pi(x)$. Thus, the expectation yields

$$\mathsf{E}[V^*(\boldsymbol{\pi} + \mathbf{h}(\boldsymbol{\pi}, u))] = \int \sum_{x\in\mathcal{X}} p(\mathbf{y} \mid x, u)\pi(x)V^*(\boldsymbol{\pi}^+)\,\mathrm{d}\mathbf{y},$$

where the components $\{\pi^+(x) \mid x \in \mathcal{X}\}$ of the vector $\boldsymbol{\pi}^+$ are given by $\pi^+(x) = \frac{p(\mathbf{y}|x,u)\pi(x)}{\sum_{x'\in\mathcal{X}} p(\mathbf{y}|x',u)\pi(x')}$. Finally, we have the corresponding HJB equation

$$V^*(\boldsymbol{\pi}) = \max_{u\in\mathcal{U}} \left\{ \sum_{x\in\mathcal{X}} R(x,u)\pi(x) + \tau \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}} \frac{\partial V^*(\boldsymbol{\pi})}{\partial \pi(x)} \Lambda(x', x \mid u)\pi(x') \right.$$
$$\left. + \tau\lambda \left(\int \sum_{x\in\mathcal{X}} p(\mathbf{y} \mid x, u)\pi(x)V^*(\boldsymbol{\pi}^+)\,\mathrm{d}\mathbf{y} - V^*(\boldsymbol{\pi}) \right) \right\}.$$

THE HAMILTON-JACOBI-BELLMAN EQUATION FOR THE CONTROLLED WONHAM FILTER. Similarly, for the controlled Wonham filter as in Section 6.2.2, i.e.,

$$\mathrm{d}\pi(x,t) = \sum_{x'\in\mathcal{X}} \Lambda(x', x \mid u(t))\pi(x', t)\,\mathrm{d}t$$
$$+ \pi(x,t)\left(\mathbf{g}(x, u(t)) - \bar{\mathbf{g}}(u(t), t)\right)^\top (\mathbf{B}\mathbf{B}^\top)^{-1}(\mathrm{d}\mathbf{y}(t) - \bar{\mathbf{g}}(u(t), t)\,\mathrm{d}t),$$

with $\bar{\mathbf{g}}(u(t), t) = \sum_{x'\in\mathcal{X}} \mathbf{g}(x', u(t))\pi(x', t)$, we have the components $\{f(\boldsymbol{\pi}, u, x) \mid x \in \mathcal{X}\}$ of the drift function $\mathbf{f}(\boldsymbol{\pi}, u)$ as

$$f(\boldsymbol{\pi}, u, x) = \sum_{x'\in\mathcal{X}} \Lambda(x', x \mid u)\pi(x') + \pi(x)(\mathbf{g}(x,u) - \bar{\mathbf{g}}(u))^\top(\mathbf{B}\mathbf{B}^\top)^{-1}(\mathbf{g}(x,u) - \bar{\mathbf{g}}(u)).$$

The row-wise components $\{\mathbf{G}(\boldsymbol{\pi}, u, x) \mid x \in \mathcal{X}\}$ of the dispersion matrix $\mathbf{G}(\boldsymbol{\pi}, u)$ read

$$\mathbf{G}(\boldsymbol{\pi}, u, x) = \pi(x)\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}(\mathbf{g}(x,u) - \bar{\mathbf{g}}(u)),$$

with $\bar{\mathbf{g}}(u) = \sum_{x'} \mathbf{g}(x', u)\pi(x')$. The jump amplitude is zero; i.e., $\mathbf{h}(\boldsymbol{\pi}, u) = \mathbf{0}$.

Hence, we have the resulting HJB equation

$$V^*(\boldsymbol{\pi}) = \max_{u\in\mathcal{U}} \sum_{x\in\mathcal{X}} R(x,u)\pi(x) + \frac{\partial V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}}\mathbf{f}(\boldsymbol{\pi}, u) + \frac{\tau}{2}\left(\frac{\partial^2 V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2}\mathbf{G}(\boldsymbol{\pi}, u)\mathbf{G}^\top(\boldsymbol{\pi}, u)\right).$$

6.3.2.1 *The Advantage Function and the Optimal Policy*

To find the optimal policy $\mu^* : \Delta^n \to \mathcal{U}$ corresponding to the optimal value function $V^*(\boldsymbol{\pi})$, it is useful to define the optimal advantage function coinciding with Eq. (6.3.18) as

$$A^*(\boldsymbol{\pi}, u) \coloneqq \mathsf{E}[R(x,u) \mid \boldsymbol{\pi}] - V^*(\boldsymbol{\pi}) + \tau\frac{\partial V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}}\mathbf{f}(\boldsymbol{\pi}, u)$$
$$+ \frac{\tau}{2}\operatorname{tr}\left(\frac{\partial^2 V^*(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2}\mathbf{G}(\boldsymbol{\pi}, u)\mathbf{G}(\boldsymbol{\pi}, u)^\top\right) + \tau\lambda\left(\mathsf{E}[V^*(\boldsymbol{\pi} + \mathbf{h}(\boldsymbol{\pi}, u))] - V^*(\boldsymbol{\pi})\right). \tag{6.3.19}$$

Note that, given Eq. (6.3.18), $V^*(\boldsymbol{\pi}) = \max_{u \in \mathcal{U}} A^*(\boldsymbol{\pi}, u) + V^*(\boldsymbol{\pi})$. Therefore, to satisfy the HJB equation, the consistency equation

$$\max_{u \in \mathcal{U}} A^*(\boldsymbol{\pi}, u) = 0 \tag{6.3.20}$$

is required. The optimal policy can then be obtained as

$$\mu^*(\boldsymbol{\pi}) = \arg\max_{u \in \mathcal{U}} A^*(\boldsymbol{\pi}, u). \tag{6.3.21}$$

6.3.3 *Algorithms*

For the calculation of an optimal policy in Eq. (6.3.21) we have to find the advantage function in Eq. (6.3.19). As the advantage function depends on the value function, we have to calculate both of them. Since the PDE for the value function in Eq. (6.3.18) does not have a closed-form solution and the evaluation of the advantage function in Eq. (6.3.19) can be costly due to the expectations on the r.h.s. of the equation, we present the following two methods to learn both functions approximately.

6.3.3.1 *Solving the Hamilton-Jacobi Bellman Equation Using a Collocation Method*

For solving the HJB equation (6.3.18) we first apply a collocation method [141, 143, 144] with a parameterized value function $V_\phi(\boldsymbol{\pi})$, which is modeled by means of a neural network. We define the residual without the maximum operator of the HJB equation under the parameterization as the advantage

$$\begin{aligned} A_\phi(\boldsymbol{\pi}, u) \coloneqq{}& \mathsf{E}[R(x, u) \mid \boldsymbol{\pi}] - V_\phi(\boldsymbol{\pi}) + \tau \frac{\partial V_\phi(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi}, u) \\ &+ \frac{\tau}{2} \operatorname{tr}\left(\frac{\partial^2 V_\phi(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi}, u) \mathbf{G}(\boldsymbol{\pi}, u)^\top \right) + \tau\lambda \left(\mathsf{E}[V_\phi(\boldsymbol{\pi} + \mathbf{h}(\boldsymbol{\pi}, u))] - V_\phi(\boldsymbol{\pi}) \right). \end{aligned}$$

For learning, we sample M beliefs $\{\boldsymbol{\pi}^{(i)} \mid i = 1, \dots, M\}$, with $\boldsymbol{\pi}^{(i)} \in \Delta^n$, from some base distribution $\boldsymbol{\pi}^{(i)} \sim p(\boldsymbol{\pi})$. As a base distribution we use in the experiments, e.g., a Dirichlet distribution $\boldsymbol{\pi}^{(i)} \sim \operatorname{Dir}(\boldsymbol{\pi} \mid \boldsymbol{\alpha})$. Further, we estimate the optimal parameters by minimizing the squared loss w.r.t. the residual of the HJB equation (6.3.18); i.e.,

$$\hat{\phi} = \arg\min_{\phi} \sum_{i=1}^{M} \left\{ \max_{u \in \mathcal{U}} A_\phi(\boldsymbol{\pi}^{(i)}, u) \right\}^2.$$

If needed, we can approximate the expectation $\mathsf{E}\left[V_\phi(\boldsymbol{\pi}^{(i)} + \mathbf{h}(\boldsymbol{\pi}^{(i)}, u))\right]$ over the observation space by sampling. For learning it is required to calculate the gradient $\frac{\partial V_\phi(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}}$ and Hessian $\frac{\partial^2 V_\phi(\boldsymbol{\pi})}{\partial \boldsymbol{\pi}^2}$ of the value function w.r.t. the input $\boldsymbol{\pi}$. Generally, this can be achieved by automatic differentiation, but for a fully connected multi-layer feedforward network, the analytic expressions are given in Appendix B.1. The analytic expression makes it possible to calculate the gradient, Hessian, and value function in one single forward pass [151].

Given the learned value function $V_{\hat{\phi}}$, we learn an approximation of the advantage function $A_\psi(\boldsymbol{\pi}, u)$ to obtain an approximately-optimal policy. To this end we use a parametrized advantage function $A_\psi(\boldsymbol{\pi}, u)$ and employ a collocation method to solve Eq. (6.3.19). To ensure the consistency Eq. (6.3.20) during learning, we apply a reparametrization [118, 152] as

$$A_\psi(\boldsymbol{\pi}, u) = \bar{A}_\psi(\boldsymbol{\pi}, u) - \max_{u' \in \mathcal{U}} \bar{A}_\psi(\boldsymbol{\pi}, u').$$

The optimal parameters are found by minimizing the squared loss as

$$\hat{\psi} = \arg\min_\psi \sum_{i=1}^{M} \sum_{u \in \mathcal{U}} \left\{ \bar{A}_\psi(\boldsymbol{\pi}^{(i)}, u) - \max_{u'} \bar{A}_\psi(\boldsymbol{\pi}^{(i)}, u') - A_{\hat{\phi}}(\boldsymbol{\pi}^{(i)}, u) \right\}^2.$$

The corresponding policy can then be easily determined as

$$\mu(\boldsymbol{\pi}) = \arg\max_{u \in \mathcal{U}} A_{\hat{\psi}}(\boldsymbol{\pi}, u),$$

using a single forward pass through the learned advantage function. A pseudo-code for the collocation learning procedure in the example case of a controlled continuous discrete filter (see Section 6.2.1) with a finite dimensional (discrete) observation space, i.e., $\mathcal{Y} = \{y^{(i)} \mid i = 1, \dots, l\}$, is provided in Fig. 6.2.

6.3.3.2 *Advantage Updating*

The HJB equation (6.3.18) can also be solved online using RL techniques. We apply the advantage updating algorithm [123] and solve Eq. (6.3.19) by employing neural network function approximators for both the value function $V_\phi(\boldsymbol{\pi})$ and the advantage function $A_\psi(\boldsymbol{\pi}, u)$. The expectations in Eq. (6.3.19) are estimated using sample runs of the POMDP. Hence, the residual error, using a belief-action-reward trajectory $\{\boldsymbol{\pi}(t), u(t), r(t) \mid t \in [0, T]\}$ of length T, can be calculated as

$$E_{\phi,\psi}(t) = A_\psi(\boldsymbol{\pi}(t), u(t)) - r(t) + V_\phi(\boldsymbol{\pi}(t)) - \tau \frac{\partial V_\phi(\boldsymbol{\pi}(t))}{\partial \boldsymbol{\pi}} \mathbf{f}(\boldsymbol{\pi}(t), u(t))$$
$$- \frac{\tau}{2} \operatorname{tr}\left(\frac{\partial^2 V_\phi(\boldsymbol{\pi}(t))}{\partial \boldsymbol{\pi}^2} \mathbf{G}(\boldsymbol{\pi}(t), u(t)) \mathbf{G}(\boldsymbol{\pi}(t), u(t))^\top \right) - \tau\lambda \left(V_\phi(\boldsymbol{\pi}(t^+)) - V_\phi(\boldsymbol{\pi}(t)) \right),$$

which can also be seen in terms of a continuous-time temporal difference (TD)-error $\delta_\phi(t)$ as $E_{\phi,\psi}(t) = A_\psi(\boldsymbol{\pi}(t), u(t)) - \delta_\phi(t)$, similar to the case of deterministic OC [120]. Again we apply the reparametrization $A_\psi(\boldsymbol{\pi}, u) = \bar{A}_\psi(\boldsymbol{\pi}, u) - \max_{u' \in \mathcal{U}} \bar{A}_\psi(\boldsymbol{\pi}, u')$ to satisfy Eq. (6.3.20). For estimating the optimal parameters $\hat{\phi}$ and $\hat{\psi}$ we minimize the sum of squared residual errors for a mini-batch of size M at time points $\{t^{(j)} \mid j = 1, \dots, M\}$; i.e.,

$$(\hat{\phi}, \hat{\psi}) = \arg\min_{(\phi,\psi)} \sum_{j=1}^{M} \left\{ E_{\phi,\psi}\left(t^{(j)}\right) \right\}^2.$$

The data for learning is generated by simulating episodes under an exploration policy as in [118]. As exploration policy, we employ a time-variant policy

$$\tilde{\mu}(\boldsymbol{\pi}, t) = \arg\max_{u \in \mathcal{U}} \{ A_\psi(\boldsymbol{\pi}, u) + \epsilon(u, t) \},$$

input : M: Number of collocation samples
$p(\boldsymbol{\pi})$: Base distribution for collocation samples
$V_\phi(\boldsymbol{\pi})$: Function approximator for state value function with parameters ϕ
$\bar{A}_\psi(\boldsymbol{\pi}, u)$: Function approximator for reparametrized advantage function with parameters ψ
output : $\{\hat{\phi}, \hat{\psi}\}$: parameters that have been fitted to approximately solve the HJB equation for $V_{\hat{\phi}}(\boldsymbol{\pi})$ and $\bar{A}_{\hat{\psi}}(\boldsymbol{\pi}, u)$, respectively.

for $i = 1$ **to** M **do**
  Sample collocation beliefs $\boldsymbol{\pi}^{(i)} \sim p(\boldsymbol{\pi})$
  for $\forall u \in \mathcal{U}$ **do**
    Compute reset condition $\pi_+^{(i)}(x, y, u) \coloneqq \frac{p(y|x,u)\pi^{(i)}(x)}{\sum_{x' \in \mathcal{X}} p(y|x',u)\pi^{(i)}(x')}, \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$
    Compute Advantage values for the samples

$$\begin{aligned} A_\phi(i, u) = &\sum_{x \in \mathcal{X}} R(x, u)\pi^{(i)}(x) + V_\phi(\boldsymbol{\pi}^{(i)}) \\ &+ \tau \sum_{x \in \mathcal{X}} \sum_{x' \in \mathcal{X}} \frac{\partial V_\phi(\boldsymbol{\pi}^{(i)})}{\partial \pi(x)} \Lambda(x', x \mid u)\pi^{(i)}(x') \\ &+ \tau\lambda(\sum_x \pi^{(i)}(x) \sum_{y \in \mathcal{Y}} V_\phi(\boldsymbol{\pi}_+^{(i)}(y, u)) - V_\phi(\boldsymbol{\pi}^{(i)})), \end{aligned}$$

    where $\{\pi_+^{(i)}(x, y, u) \mid x \in \mathcal{X}\}$ are the components of $\boldsymbol{\pi}_+^{(i)}(y, u)$.
  end
  Compute best action $u^i = \arg\max_{u \in \mathcal{U}} A_\phi(i, u)$
end
Estimate parameters by solving

$$\hat{\phi} = \arg\min_\phi \sum_{i=1}^{M} (A_\phi(i, u^i))^2.$$

Recompute all advantage values $\{A_{\hat{\phi}}(i, u) \mid \forall (i, u) \in \{1, \ldots, M\} \times \mathcal{U}\}$ with fitted $\hat{\phi}$
Solve the non-linear least squares problem

$$\hat{\psi} = \arg\min_\psi \sum_{i=1}^{M} \sum_{u \in \mathcal{U}} (\bar{A}_\psi(\boldsymbol{\pi}^{(i)}, u) - \max_{u' \in \mathcal{U}} \bar{A}_\psi(\boldsymbol{\pi}^{(i)}, u') - A_{\hat{\phi}}(i, u))^2.$$

FIGURE 6.2: Collocation algorithm for the controlled continuous-discrete filter and finite observation space.

where $\epsilon(u, t)$ is a stochastic exploration process which we choose as the Ornstein-Uhlenbeck process $\mathrm{d}\epsilon(u, t) = -\kappa\epsilon(u, t)\,\mathrm{d}t + \sigma\, dw_\epsilon(t)$. Generated trajectories are subsampled and saved to a replay buffer [153] which is used to provide data for the training procedure.

6.4 EVALUATION

6.4.1 *Experimental Tasks*

We tested our derived methods on several toy tasks of continuous-time POMDPs with discrete state and action space: An adaption of the popular tiger problem [154], a decentralized multi-agent network transmission problem [155] implementing the slotted aloha protocol, and a grid world problem. All problems are adapted to continuous-time and observations at discrete time points. The following provides an overview of the considered problems and how we model them as continuous-time POMDPs. A detailed description of the hyper-parameters used can be found in the supplementary material of the published work [1].

6.4.1.1 *The Tiger Problem*

In the tiger problem, the state consists of the position of a tiger (*left*/*right*), and the agent has to decide between three actions for either improving his belief (*listen*) or exploiting his knowledge to avoid the tiger (*left*/*right*). While listening, the agent can wait for an observation.

MODELING. We use an adaptation of the classical tiger problem for modeling[154], where the agent has to choose between two doors to open. Behind one door, a dangerous tiger is waiting to eat the agent; thus, the agent must choose the other door to become free. Besides opening a door, the agent can also decide to wait and listen to localize the tiger. We adapted the problem to continuous time by defining the POMDP as follows: The state space $\mathcal{X}$ consists of the possible positions of the tiger; i.e., $\mathcal{X} = \{\textit{tiger left}, \textit{tiger right}\}$. Executable actions of the agent are the elements of the action space $\mathcal{U} = \{\textit{listen}, \textit{open left}, \textit{open right}\}$. The tiger always stays at the same position; therefore, all the transition rates $\Lambda(x', x \mid u)$ are set to zero. When executing the action $u = \textit{open}$, the agent receives a reward with a rate of 0.1 for the door without the tiger and a negative reward of -1.0 for the door with the tiger. For executing the hearing action, the agent accumulates rewards of rate -0.01 but receives observations with a rate of 2 by hearing the tiger either on the left side ($y = \textit{hear tiger left}$) or on the right side ($y = \textit{hear tiger right}$). The received information is correct with a probability of 0.85; thus, with a probability of 0.15, one hears the tiger at the opposite side. The discount factor τ is set to 0.9.

6.4.1.2 *The Slotted Aloha Problem*

In the transmission problem, stations have to adjust their sending rate to successfully transmit packages over a single channel as in the slotted Aloha protocol. Each station might have a package to send or not, and the only observation is the past state of the channel, which can be either *idle*, *transmission*, or *collision*. New packages arrive with a fixed rate at the stations. As for each number of available packages – with the exception of no packages available – there is a unique optimal action, a finite number of transmission rates as actions is sufficient.

MODELING. Modeling the slotted aloha transition problem as POMDP was introduced in [155] and dealt with the described decentralized control of stations transmitting packages in a single channel network. The task can be modeled as a POMDP, since adjusting the sending rate of the stations is based only on the information about the past transmission state. The state-space consists of the number of stations that have a package ready for sending and the past transmission state as observation; thus, the state space is given by $\mathcal{X} = \{0, \dots, B\} \times \{\textit{idle}, \textit{transmission}, \textit{collision}\}$, where we choose $B = 9$ sending stations as it limits the total number of states to thirty. For our continuous-time POMDP model, we consider a continuous-time adaptation of [155]: Observations at discrete time points arrive with rate $\lambda = 0.5$ and contain the past transmission state of the system, which is included in the current state; thus $\mathcal{Y} = \{\textit{idle}, \textit{transmission}, \textit{collision}\}$. While the maximum number of packages is not reached, new packages arrive with a rate of 0.5, leaving the past transmission state unchanged. The stations can send a package simultaneously with a rate of 5 but do not need to. The action ρ represents the probability with which a package is sent by a station, resulting in an actual send rate of $5\frac{\rho}{B}$.

The probability for transmission states given that b packages are available are calculated as

$$\begin{aligned} p(\textit{idle} \mid \rho) &= (1-\rho)^b \\ p(\textit{transmission} \mid \rho) &= \rho(1-\rho)^{b-1} \\ p(\textit{collision} \mid \rho) &= 1 - p(\textit{idle} \mid \rho) - p(\textit{transmission} \mid \rho) \end{aligned}$$

In the case of perfect information, we can calculate the optimal probability for transmission ρ^* by maximizing the probability of successful transmission, which can be easily obtained by setting the derivative to zero resulting into

$$\rho^* = \arg\max_{\rho} \rho(1-\rho)^{b-1} = \frac{1}{b}$$

This motivates the discretization of the actions resulting in $\mathcal{U} = \left\{\frac{1}{b} \mid b = 1, \dots, B\right\}$.

6.4.1.3 *The Grid World Problem*

In the grid world problem, an agent has to navigate through a grid world by choosing the directions in which to move next; for an analog discrete-time example, see also Section 5.4

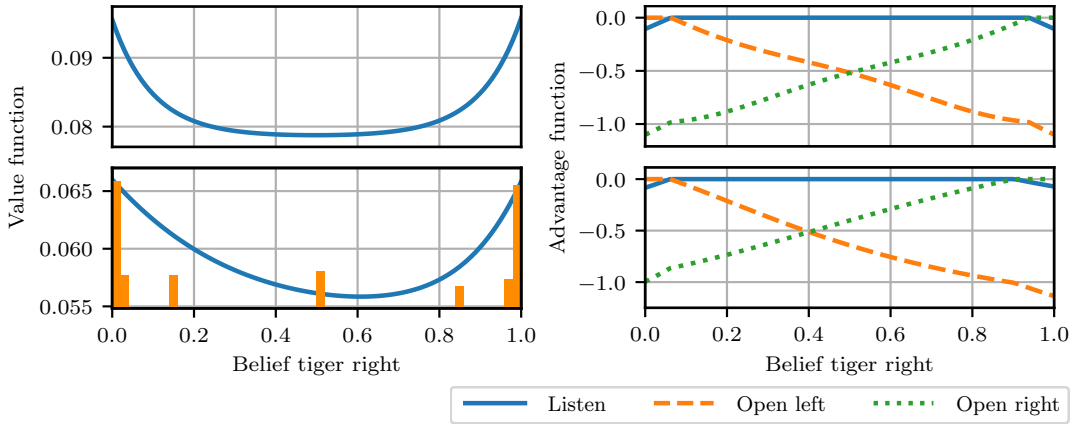


FIGURE 6.3: Learned value and advantage function for the continuous-time tiger problem. The upper plots show the approximated functions learned by the offline collocation method, while for lower plots, the online advantage updating method was applied. The orange bars in the lower-left plot show the proportions of the beliefs encountered during online learning. The advantage functions on the right can be used to determine the policy of the optimal agent.

in the previous chapter. The agent transitions with an exponentially distributed amount of time and, while doing so, can slip with some probability so that he instead moves in another direction. The agent receives only noisy information of his position from time to time.

MODELING. For the continuous-time POMDP model, we consider an agent moving in a 6×6 grid world with a goal at position $(3, 2)$. There are four actions $\mathcal{U} = \{up, down, left, right\}$ indicating the direction the agent wants to move next. In our continuous-time setting, the agent moves at an exponentially distributed amount of time with a rate of 10. With a probability of 0.7, it moves into the indicated direction, but with probability 0.1 moves into one of the other three directions due to slipping. A movement to invalid fields such as walls is not possible; thus, the transition probability is set to zero for those fields, and the remaining probabilities are renormalized. Being at the goal position provides the agent with a reward rate of 1; otherwise, no reward is accumulated. The agent receives a noisy signal about his current position at a rate of 2. The signal indicates a field sampled from a discretized 2D Gaussian distribution with a standard deviation of 0.1 centered at the agent's position.

6.4.2 *Results*

Both the offline collocation method and the online advantage updating method manage to learn reasonable approximations of the value and advantage functions for the considered problems.

6.4.2.1 *The Tiger Problem*

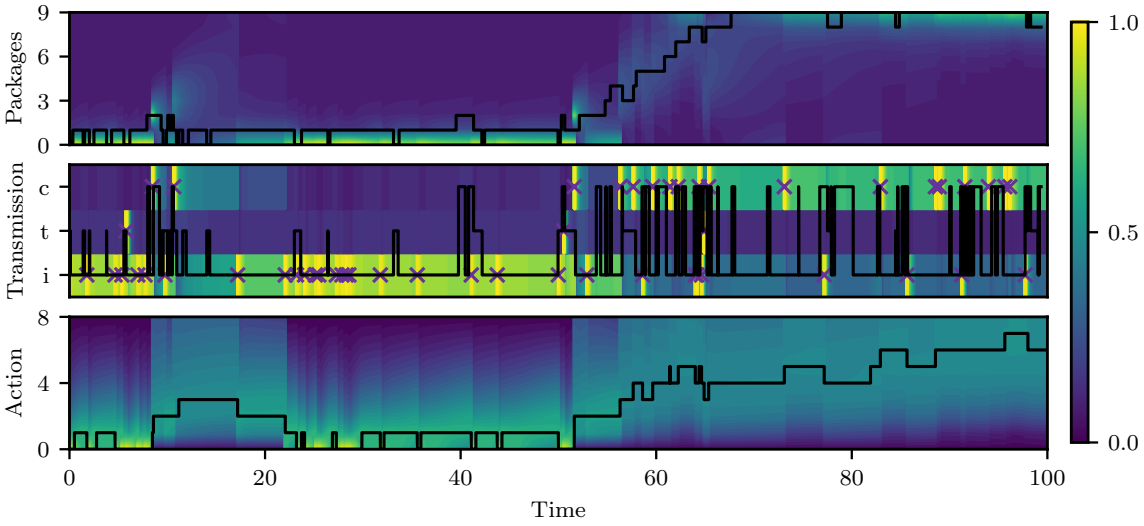
For the tiger problem, the learned value and advantage functions over the whole belief space are visualized in Fig. 6.3. The parabolic shape of the value function correctly indicates that higher certainty of the belief about the tiger's position results in a higher expected discounted cumulative reward as the agent can exploit this knowledge to omit the tiger. Note that the value function learned by the online advantage updating method differs marginally from the one learned by collocation in shape. This is because in the advantage updating method, only actually visited belief states are used in the learning process to approximate the value function, and points in between need to be interpolated. The advantage function correctly indicates that for uncertain beliefs, it is advantageous to first gain more information by executing the action $u = \textit{listen}$. On the other hand, for certain beliefs, directly opening the respective door is more useful in terms of reward for certain beliefs. Therefore, opening the opposed door is considered even more disadvantageous in these cases.

6.4.2.2 *The Slotted Aloha Problem*

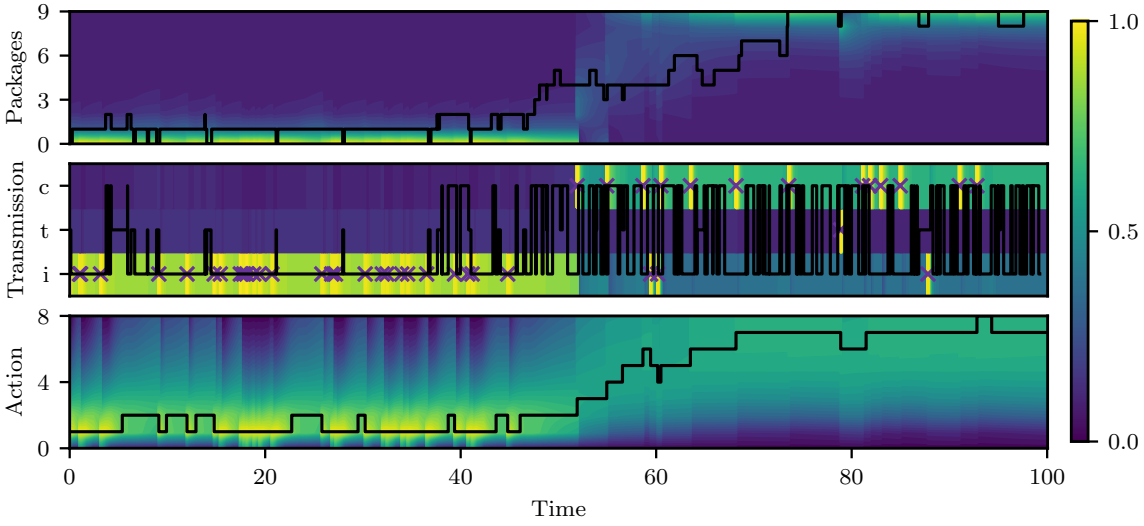
For the slotted Aloha transmission problem, the results are depicted in Fig. 6.4. Here, a random execution of the policy learned by the offline collocation method, and the online advantage updating method is shown in Figs. 6.4a and 6.4b, respectively. The upper two plots of Figs. 6.4a and 6.4b show the true states; i.e., number of packages and past system state, and the agent's beliefs which follow the prior dynamics of the system and jump when new observations are made. The plot at the bottom visualizes the learned advantage function, and the resulting policy for the encountered beliefs is visualized. As derived in Section 6.4.1.2, in case of perfect information of the number of packages b, it is optimal to execute action $b - 1$ with exception of $b = 0$ where the action does not matter. When facing uncertainty, however, an optimistic behavior which is also reflected by the learned policy, is more reasonable: As for a lower number of packages, the probability of sending a package and therefore collecting a higher reward is higher. Thus, in case of uncertainty, one should opt for a lower action than the one executed under perfect information. We observed that the results for the offline collocation method and the online advantage updating method look qualitatively equal, reflecting the same reasonable behavior.

6.4.2.3 *The Grid World Problem*

Results for the grid world problem are visualized in Fig. 6.5 which shows the learned value and advantage function using the offline collocation method (Fig. 6.5a) and the online advantage updating method (Fig. 6.5b). The figures visualize the resulting values for certain beliefs; i.e., being at the respective fields with probability one. As expected, the learned value function assigns higher values to fields that are closer to the goal position $(3, 2)$. Actions leading to these states have higher advantage values. For assessing results for uncertain beliefs which are actually encountered when running the system, the figures also contain a sample run that successfully directs the agent to the goal. Results for both the collocation method and the advantage updating method are found to be very similar.

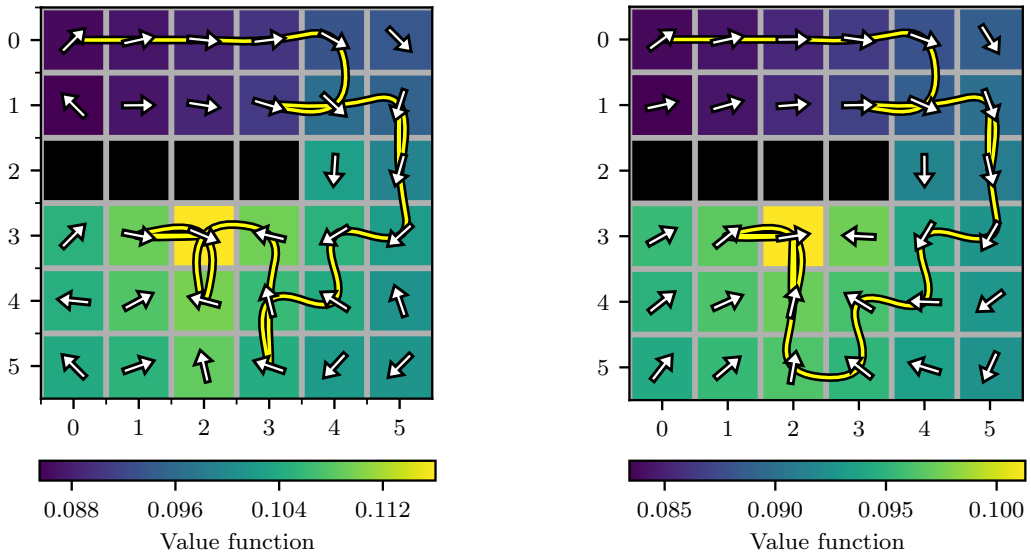


(a) Offline collocation method.



(b) Online advantage updating method.

FIGURE 6.4: A sample run for the slotted Aloha transmission problem using a policy learned by the collocation method (a) and advantage updating method (b). The upper plot (*Packages*) in (a) and (b) show the actual number of packages available in the system while the middle one (*Transmission*) shows the past system state which can be either *idle* (i), *transmission* (t), or *collision* (c). The background of these plots (*Packages* and *Transmission*) indicate the marginal belief of the number of packages and past system state, respectively. A cross indicates the past system states that are observed at discrete time points. The lower plot (*Action*) shows the agent's policy resulting from the learned advantage function, while the background indicates the per time-step normalized advantage function is indicated by the background.



(a) Offline collocation method.

(b) Online advantage updating method.

FIGURE 6.5: Learned value and advantage function for certain beliefs in the continuous-time grid world problem using the collocation method (a) and the advantage updating method (b). Being at the goal at position $(3, 2)$ results in a reward for the agent. The black fields in row 3 represent a wall that cannot be crossed. Colors of the fields indicate the approximated value function, while arrows show the proportions of the advantage functions among different actions. The yellow curvy path indicates a respective random run starting at the field $(0, 0)$ under the resulting policy.

6.5 SUMMARY

In this final chapter of the main part of this thesis, we introduced a new model for decision-making in continuous time under partial observability. We presented (i) a collocation-based offline method and (ii) an advantage updating online algorithm, which both find approximate solutions by the use of deep learning techniques. We qualitatively discussed the found solutions for a set of toy problems adapted from literature for evaluation. The solutions have been shown to represent reasonable optimal decisions for the given problems.

In the future, we are interested in exploring ways to make the proposed model applicable to more realistic large-scale problems. First, throughout this work, we assumed a known model. In many applications, however, this might not be the case and investigating how to realize dual control methods [12] might be a fruitful direction, see also the model from Chapter 5. New scalable techniques for estimating parameters of latent CTMCs, which could be used, are discussed in [156] but also learning the filtering distribution directly from data might be an option if the model is not available [132]. An issue we faced was that for high-dimensional problems, the algorithms seemed to slowly converge to the optimal solution as the belief space grows linearly in the number of dimensions w.r.t. the number of states of the latent process. The introduction of variational and sampling methods seems promising to project the filtering distribution to a lower-dimensional space and make the solution of high-dimensional problems feasible. This could also enable the extension to continuous state spaces, where the filtering distribution is generally infinite-dimensional and has to be projected to a finite-dimensional representation, e.g., as it is done in assumed

density filtering [17]. This will enable the use for interesting applications, for example, in queueing networks or, more generally, in stochastic hybrid systems see also [5].

7

CONCLUSION

7.1 SUMMARY AND CONCLUDING REMARKS

This thesis focused on model-based Bayesian inference, learning, and decision-making and provided applications in communication systems. The specific contributions are (i) methods for probabilistic modeling, including several prior distributions describing an a priori Bayesian belief. (ii) Novel Inference and learning schemes are developed using exact Bayesian inference and tractable approximate posterior inference methods, based on MCMC and VI. (iii) Finally, new decision-making algorithms are presented, which are tractable and approximately optimal w.r.t. a reward function. This thesis established sensible models of real systems by discussing several modeling aspects, such as models for continuous and discrete state spaces, inferring the number of distinct states, and modeling the dynamical processes in episodes (discrete-time) or in continuous time.

A Bayesian model for point process data was presented with a tractable MCMC algorithm in Chapter 3. This chapter focussed explicitly on a non-parametric assumption on the number of discrete states, yielding a countable infinite state space for the latent process. By modeling non-exponential holding times a high predictive performance was established.

Episodic decision-making for the application of ABR video-streaming in NDN networks was considered in Chapter 4. Here, a contextual multi-armed bandit strategy was exploited to optimize a QoE based reward. The reward function was modeled using a Bayesian sparse linear model, and some scalable algorithms for inference were provided. The algorithms are extensively evaluated and are especially fitting for decision-making problems, where decisions must be executed in a time-critical manner.

A more detailed level of modeling was discussed in Chapter 5 by providing a Bayesian model for correlation structures occurring in decision-making problems. A new tractable inference scheme based on mean-field variational inference in a conjugate augmented space is discussed. This chapter focused on several types of correlation models including an evaluation for a vast range of problems occurring in decision-making, such as imitation learning, sub-goal modeling, and BRL. For decision-making, a strategy based on the PSRL

principle was used, which is easy to implement and yields with the provided correlation prior to superior performance in terms of cumulative reward, compared to state-of-the-art naive Bayesian priors.

Finally, in Chapter 6 decision-making in the setup of partial observability of the state variable was discussed. Here, a new continuous-time POMDP model was presented, and the optimality conditions for the resulting OC problem in belief space were derived. Building on prior work, some derivations for optimal filtering dynamics were given, which describe the evolution of the belief state. Additionally, for decision-making, new numerical algorithms to solve for optimality utilizing function approximation methods were derived and evaluated.

Building on the proposed models, inference, and decision-making schemes, we showed the applicability to several examples in communication systems and beyond.

7.2 OUTLOOK

The presented models, inference schemes, and decision-making algorithms provide several opportunities for future research. First, an exciting application is the use of Bayesian non-parametric models as the one discussed in Chapter 3 in the context of decision-making. Here, similar ideas from discrete-time decision-making, as e.g., presented in [157], could serve as a base for calculating optimal decision-making schemes in the continuous-time domain extending the algorithms in Chapter 6.

The correlation model presented in Chapter 5, could serve as a basis for model-learning in other domains. For example, it could be employed to model the dynamic evolution of the network features in Chapter 4. A further promising application would be on learning a transition model, such as those discussed in Chapter 6.

One of the most promising future directions is to make the decision-making scheme from Chapter 6 tractable in high dimensions. One possibility is to look at approximate filtering to describe the dynamic evolution of the Bayesian belief state using a lower-dimensional approximate representation. This could enable the applicability of the method to high-dimensional problems, such as complex queueing networks.

Overall, the methods discussed in this thesis provide opportunities for future applications, e.g., optimal decision-making for the switching between functions in communication services as addressed in [7]. Generally, other application domains are deemed to benefit from using the presented model-based solutions.

APPENDICES

APPENDIX A: PROBABILITY

A.1 DEFINITIONS AND NOTATION

Here, we give some definitions from probability theory; we refer the reader to, e.g., [18–20, 158] for a more rigorous treatment of the discussed topics.

THE PROBABILITY SPACE. We consider a probability space given by the tuple $(\Omega, \Sigma, \mathsf{P})$, where Ω is the sample space, $\Sigma \subseteq 2^\Omega$ is the σ-algebra and $\mathsf{P} : \Sigma \to [0, 1]$ is the probability measure.

RANDOM VARIABLES. Formally, a RV X is a function $X : \Omega \to \mathcal{X}$, which maps an element (event) $\omega \in \Omega$ of the sample space to some space $\mathcal{X}$, e.g., to the space of real numbers, in which case $\mathcal{X} = \mathbb{R}$. For the most part, we omit the argument $\omega \in \Omega$ of the function $X(\omega)$, and we write X. Note that by abuse of notation, we will mainly use small letters, e.g., x for RVs and capital letters, e.g., X only in formulations that explicitly contain the probability measure.

We can assign a probability that X is in a set $\mathcal{A}$ as

$$\mathsf{P}(X \in \mathcal{A}) \equiv \mathsf{P}(\{\omega \in \Omega : X(\omega) \in \mathcal{A}\}).$$

Additionally, we write

$$\mathsf{P}(X \in \mathcal{A}) = \int \mathbb{1}(x \in \mathcal{A}) \, \mathsf{P}(X \in \mathrm{d}x),$$

where we integrate w.r.t. the probability measure $\mathsf{P}(X \in \mathrm{d}x)$, e.g., in the case where $\mathcal{X} = \mathbb{R}^n$, $\mathrm{d}x$ denotes the Lebesque measure on $\mathbb{R}^n$.

PROBABILITY DISTRIBUTION FUNCTIONS. Throughout this thesis, we write for the cumulative distribution function (CDF)

$$P(\mathbf{x}) \coloneqq \mathsf{P}(X_1 \leq x_1, \ldots, X_n \leq x_n),$$

where $\mathbf{x} = [x_1, \ldots, x_n]^\top$ and $\mathbf{X} = [X_1, \ldots, X_n]^\top$ denotes a RV in $\mathbb{R}^n$.

For the corresponding complementary cumulative distribution function (CCDF) we write

$$\bar{P}(\mathbf{x}) \coloneqq 1 - P(\mathbf{x}).$$

Let X denote a RV in some countable space $\mathcal{X}$, then we write for the PMF

$$p(x) = \mathsf{P}(X = x).$$

Similar, for a RV $\mathbf{X} \in \mathcal{X}$ in some space $\mathcal{X} \subseteq \mathbb{R}^n$ we write for the PDF

$$p(\mathbf{x}) \coloneqq \frac{\partial^n}{\partial x_1 \cdots \partial x_n} P(\mathbf{x}).$$

More formally, for a set $\mathcal{A}$, we can write

$$\begin{aligned} \mathsf{P}(\mathbf{X} \in \mathcal{A}) &= \int \mathbb{1}(\mathbf{x} \in \mathcal{A}) \, \mathsf{P}(\mathbf{X} \in \mathrm{d}\mathbf{x}) \\ &= \int \mathbb{1}(\mathbf{x} \in \mathcal{A}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \end{aligned} \qquad \Leftrightarrow \qquad p(\mathbf{x}) \coloneqq \frac{\mathrm{d}\,\mathsf{P}}{\mathrm{d}\mathbf{x}},$$

where $\frac{\mathrm{d}\,\mathsf{P}}{\mathrm{d}\mathbf{x}}$ is the Radon-Nikodym derivative between the probability measure $\mathsf{P}(\mathbf{X} \in \mathrm{d}\mathbf{x})$ and the Lebesque measure $\mathrm{d}\mathbf{x}$ on $\mathbb{R}^n$.

THE QUANTILE FUNCTION We define the quantile function $Q_x(q, p(x))$ associated to the PMF or PDF $p(x)$ as

$$\mathsf{P}(X \leq Q_x(q, p(x))) = q,$$

where the distribution of the RV X follows $p(x)$.

EXPECTATIONS. For the expectation of some function $f(x)$, with a countable space $x \in \mathcal{X}$ and a PMF $p(x)$ we write

$$\mathsf{E}[f(x)] \coloneqq \sum_{x \in \mathcal{X}} f(x) p(x)$$

and for a PDF $p(x)$ we write

$$\mathsf{E}[f(x)] \coloneqq \int f(x) p(x) \, \mathrm{d}x.$$

More generally, for a probability measure P we have

$$\mathsf{E}[f(x)] \coloneqq \int f(x) \, \mathsf{P}(X \in \mathrm{d}x).$$

Additionally, for the variance, we have

$$\mathsf{Var}[x] \coloneqq \mathsf{E}\big[(x - \mathsf{E}[x])^2\big]$$

and for the covariance matrix, we write

$$\mathsf{Cov}[\mathbf{x}] \coloneqq \mathsf{E}\Big[(\mathbf{x} - \mathsf{E}[\mathbf{x}])\,(\mathbf{x} - \mathsf{E}[\mathbf{x}])^\top\Big].$$

Lastly, we define the entropy of a RV x with a PDF or a PMF $p(x)$ as

$$\mathsf{H}[x] \coloneqq \mathsf{E}[-\log p(x)]$$

and for the KL divergence or *relative entropy* w.r.t. another PDF or a PMF $q(x)$ we have

$$\mathsf{KL}(q(x) \parallel p(x)) \coloneqq \mathsf{E}\left[\log \frac{q(x)}{p(x)}\right],$$

where above expectation is carried out w.r.t. $q(x)$.

RULES OF PROBABILITY. The two most essential rules in probability are the sum rule

$$p(x) = \sum_y p(x, y) \quad \Leftrightarrow \quad p(x) = \int p(x, y)\,\mathrm{d}y$$

and the product rule

$$p(x, y) = p(x \mid y)p(y).$$

As a direct consequence of the product rule and the symmetry in the arguments of $p(x, y)$ we have the *Bayes' rule*

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}.$$

A.2 PROBABILITY DISTRIBUTIONS

This section includes some probability distributions we use in this thesis. For more, see, e.g., [22–24].

A.2.1 *The Gamma Distribution*

The PDF of a gamma distribution is

$$\mathrm{Gam}(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

with support $x \in \mathbb{R}_{>0}$, shape parameter $\alpha \in \mathbb{R}_{>0}$, and rate parameter $\beta \in \mathbb{R}_{>0}$.

The gamma distribution with parameters $\boldsymbol{\theta} = [\alpha, \beta]^\top$ is a member of the exponential family distribution with base measure

$$h(x) = 1,$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = [\alpha - 1, -\beta]^\top,$$

sufficient statistics

$$\mathbf{s}(x) = [\log x, x]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = \log \Gamma(\eta_1 + 1) - (\eta^{(1)} + 1) \log(-\eta_2).$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = [\eta_1 + 1, -\eta_2]^\top.$$

A.2.2 *The Categorical Distribution*

The PMF of a categorical distribution is

$$\mathrm{Cat}(x \mid \mathbf{p}) = \prod_{i=1}^{n} p_i^{\mathbb{1}(x = x^{(i)})},$$

with support $x \in \{x^{(i)} \mid i = 1, \ldots, n\}$ and probability vector $\mathbf{p} = [p_1, \ldots, p_n]^\top \in \Delta^n$.

The categorical distribution with parameters $\boldsymbol{\theta} = \mathbf{p}$ is a member of the exponential family distribution with base measure

$$h(x) = 1,$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = [\log p_1, \ldots, \log p_n]^\top,$$

sufficient statistics

$$\mathbf{s}(x) = [\mathbb{1}(x = x^{(1)}), \ldots, \mathbb{1}(x = x^{(n)})]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = 0.$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = [e^{\eta_1}, \ldots, e^{\eta_n}]^\top.$$

A.2.3 *The Chinese Restaurant Process*

The Chinese restaurant process $\{k_j \mid j \in \mathbb{N}_{>0}\}$ is a discrete-time stochastic process, where the transition distribution is given by the PMF

$$\mathrm{CRP}(k_j \mid \alpha, k_{j-1}, k_{j-2}, \ldots, k_1) = \left(\frac{\alpha}{j-1+\alpha}\right)^{\mathbb{1}(k_j = m_j+1)} \prod_{i=1}^{m_j} \left(\frac{\Psi_{ij}}{j-1+\alpha}\right)^{\mathbb{1}(k_j=i)},$$

with support $k_j \in \{1, \ldots, m_j + 1\}$ and concentration parameter $\alpha \in \mathbb{R}_{>0}$. Here, the total number of previously occupied tables is $m_j = \max(k_{j-1}, \ldots, k_1)$ and the number of customers sitting at table i before customer j arrives is $\Psi_{ij} = \sum_{l=1}^{j-1} \mathbb{1}(k_l = i)$. The initial distribution is given as

$$\mathrm{CRP}(k_1 \mid \alpha) = \mathbb{1}(k_1 = 1),$$

with $m_1 = 0$.

For more details on the Chinese restaurant process see, e.g., [42].

A.2.4 *The Exponential Distribution*

The PDF of an exponential distribution is

$$\mathrm{Exp}(x \mid \lambda) = \lambda e^{-\lambda x},$$

with support $x \in \mathbb{R}_{\geq 0}$, and rate parameter $\lambda \in \mathbb{R}_{>0}$.

The exponential distribution with parameter $\theta = \lambda$ is a member of the exponential family distribution with base measure

$$h(x) = 1,$$

natural parameters

$$\eta(\theta) = -\lambda$$

sufficient statistic

$$s(x) = x$$

and log-normalizer

$$A(\eta) = -\log(-\eta).$$

The inverse transform of the natural parameter is obtained by

$$\theta(\eta) = -\eta.$$

A.2.5 *The Log-Normal Distribution*

The PDF of a log-normal distribution is

$$\text{Lognorm}(x \mid \mu, \sigma^2) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right),$$

with support $x \in \mathbb{R}_{>0}$, mean parameter μ, and variance parameter σ^2.

The log-normal distribution with parameters $\boldsymbol{\theta} = [\mu, \sigma^2]^\top$ is a member of the exponential family distribution with base measure

$$h(x) = \frac{1}{\sqrt{2\pi}x},$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = \left[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right]^\top,$$

sufficient statistics

$$\mathbf{s}(x) = [\log x, (\log x)^2]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = -\frac{{\eta_1}^2}{4\eta_2} - \frac{1}{2}\log(-2\eta_2).$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = \left[-\frac{\eta_1}{2\eta_2}, -\frac{1}{2\eta_2}\right]^\top.$$

A.2.6 *The Uniform Distribution*

The PDF of a (continuous) uniform distribution is

$$\text{Uniform}(x \mid a, b) = \frac{1}{b-a},$$

with support $x \in [a, b] \subset \mathbb{R}$, lower bound $a \in \mathbb{R}$, and upper bound $b \in \mathbb{R}$, with $b > a$.

A.2.7 *The Multivariate Normal Distribution*

The PDF of a multivariate normal (or Gauss) distribution is

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\},$$

with support $\mathbf{x} \in \mathbb{R}^n$, mean parameter $\boldsymbol{\mu} \in \mathbb{R}^n$, and positive semidefinite covariance parameter $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$.

The multivariate normal distribution with parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a member of the exponential family distribution with base measure

$$h(\mathbf{x}) = (2\pi)^{-n/2},$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = \left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, -\frac{1}{2}\boldsymbol{\Sigma}^{-1}\right),$$

sufficient statistics

$$\mathbf{s}(\mathbf{x}) = \left(\mathbf{x}, \mathbf{x}\mathbf{x}^\top\right)$$

and log-normalizer

$$A(\boldsymbol{\eta}) = -\frac{1}{4}\boldsymbol{\eta}^{(1)\top}\boldsymbol{\eta}^{(2)-1}\boldsymbol{\eta}^{(1)} - \frac{1}{2}\log|-2\boldsymbol{\eta}^{(2)}|.$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = \left(-\frac{1}{2}\boldsymbol{\eta}^{(2)-1}\boldsymbol{\eta}^{(1)}, -\frac{1}{2}\boldsymbol{\eta}^{(2)-1}\right).$$

A.2.8 *The Multinomial Distribution*

The PMF of a multinomial distribution is

$$\mathrm{Mult}(\mathbf{x} \mid N, \mathbf{p}) = \frac{N!}{\prod_{i=1}^K x_i!}\prod_{i=1}^K p_i^{x_i},$$

with support $\mathbf{x} = [x_1, \ldots, x_K]^\top \in \{0, \ldots, N\}^K$, with $\sum_{i=1}^K x_i = N$, number of trials $N \in \mathbb{N}_{>0}$, and probability vector $\mathbf{p} = [p_1, \ldots, p_K]^\top \in \Delta^K$.

The multinomial distribution with parameters $\boldsymbol{\theta} = \mathbf{p}$ is a member of the exponential family distribution with base measure

$$h(\mathbf{x}) = \frac{N!}{\prod_{i=1}^K x_i!},$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = [\log p_1, \ldots, \log p_K]^\top,$$

sufficient statistics

$$\mathbf{s}(\mathbf{x}) = [x_1, \ldots, x_K]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = 0.$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = [e^{\eta_1}, \ldots, e^{\eta_K}]^\top.$$

A.2.9 *The Binomial Distribution*

The PMF of a binomial distribution is

$$\text{Bin}(x \mid N, p) = \binom{N}{x} p^x (1-p)^{N-x},$$

with support $x \in \{0, \ldots, N\}$, number of trials $N \in \mathbb{N}_{>0}$, and success probability parameter $p \in [0, 1] \subset \mathbb{R}$.

The binomial distribution with parameter $\theta = p$ is a member of the exponential family distribution with base measure

$$h(x) = \binom{N}{x},$$

natural parameters

$$\eta(\theta) = \log \frac{p}{1-p},$$

sufficient statistic

$$s(x) = x$$

and log-normalizer

$$A(\eta) = N \log \left(1 + e^{\eta}\right).$$

The inverse transform of the natural parameters is obtained by

$$\theta(\eta) = \frac{1}{1 + e^{-\eta}}.$$

A.2.10 *The Poisson Distribution*

The PMF of a Poisson distribution is

$$\text{Pois}(x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda},$$

with support $x \in \mathbb{N}_{\geq 0}$, and rate parameter $\lambda \in \mathbb{R}_{>0}$.

The Poisson distribution with parameter $\theta = \lambda$ is a member of the exponential family distribution with base measure

$$h(x) = \frac{1}{x!},$$

natural parameters

$$\eta(\theta) = \log \lambda,$$

sufficient statistic

$$s(x) = x$$

and log-normalizer

$$A(\eta) = e^{\eta}.$$

The inverse transform of the natural parameters is obtained by

$$\theta(\eta) = e^{\eta}.$$

A.2.11 *The Dirichlet Distribution*

The PDF of a Dirichlet distribution is

$$\mathrm{Dir}(\mathbf{p} \mid \boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} p_i^{\alpha_i - 1},$$

with support $\mathbf{p} = [p_1, \ldots, p_K]^\top \in \Delta^K$, and concentration parameters $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]^\top \in \mathbb{R}^K_{>0}$.

The Dirichlet distribution with parameters $\boldsymbol{\theta} = \boldsymbol{\alpha}$ is a member of the exponential family distribution with base measure

$$h(\mathbf{p}) = \frac{1}{\prod_{i=1}^{K} p_i},$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = [\alpha_1, \ldots, \alpha_K]^\top,$$

sufficient statistics

$$\mathbf{s}(\mathbf{x}) = [\log p_1, \ldots, \log p_K]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = \sum_{i=1}^{K} \log \Gamma(\eta_i) - \log \Gamma\left(\sum_{i=1}^{K} \eta_i\right).$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = [\eta_1, \ldots, \eta_K]^\top.$$

A.2.12 *The Pólya-Gamma Distribution*

Let $X \sim \mathrm{PG}(x \mid b, c)$ be a RV distributed according to the Pólya-Gamma (PG) distribution, with parameters $b \in \mathbb{R}_{>0}$ and $c \in \mathbb{R}$, then X is equal in distribution to an infinite sum of gamma RVs, to be precise

$$X \overset{\mathrm{D}}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{G_k}{(k - 1/2)^2 + c^2/(4\pi^2)},$$

where $G_k \sim \mathrm{Gam}(g \mid b, 1)$ and $\overset{\mathrm{D}}{=}$ indicates equality in distribution.

Hence, the PDF of the PG distribution can be expressed as an infinite convolution of gamma PDFs. Alternatively, we have a density formula using an alternating infinite sum, as

$$\mathrm{PG}(x \mid b, c) = \left\{\cosh^b(c/2)\right\} \frac{2^{b-1}}{\Gamma(b)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(n+b)}{\Gamma(n+1)} \frac{(2n+b)}{\sqrt{2\pi x^3}} \cdot \exp\left\{-\frac{(2n+b)^2}{8x} - \frac{c^2}{2} x\right\},$$

with support $x \in \mathbb{R}_{>0}$ and parameters $b \in \mathbb{R}_{>0}$ and $c \in \mathbb{R}$, for more see [103].

A.2.13 *The Generalized Inverse Gaussian Distribution*

The PDF of a generalized inverse Gaussian (GIG) distribution is

$$\mathcal{GIG}(x \mid p, a, b) = (a/b)^{p/2}(2\mathrm{K}_p(\sqrt{ab}))^{-1}x^{p-1}\exp\{(ax + b/x)/2\},$$

with support $x \in \mathbb{R}_{>0}$ and parameters $p \in \mathbb{R}$, $a \in R_{>0}$, and $b \in R_{>0}$. The GIG distribution with parameters $\boldsymbol{\theta} = [p, a, b]^\top$ is a member of the exponential family distribution with base measure

$$h(x) = 1,$$

natural parameters

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = [p - 1, -a/2, b/2]^\top,$$

sufficient statistics

$$\mathbf{s}(x) = [\log x, x, 1/x]^\top$$

and log-normalizer

$$A(\boldsymbol{\eta}) = \log\left(\left(-\eta^{(2)}/\eta^{(3)}\right)^{(\eta^{(1)}+1)/2}\left\{2\mathrm{K}_{\eta^{(1)}+1}\left(\sqrt{-4\eta^{(2)}\eta^{(3)}}\right)\right\}^{-1}\right).$$

The inverse transform of the natural parameters is obtained by

$$\boldsymbol{\theta}(\boldsymbol{\eta}) = [\eta^{(1)} + 1, -2\eta^{(2)}, 2\eta^{(3)}]^\top.$$

A.3 SPECIAL FUNCTIONS

In this section, we give some definitions of some known special functions used in this thesis.

THE GAMMA FUNCTION. The gamma function $\Gamma : \mathbb{R} \to \mathbb{R}$ is given by the integral definition

$$\Gamma(x) = \int_0^\infty z^{x-1}e^{-z}\,\mathrm{d}z.$$

THE ERROR AND INVERSE ERROR FUNCTION. The error function $\mathrm{erf} : \mathbb{R} \to [-1, 1]$ is given as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-z^2}\,\mathrm{d}z.$$

The inverse error function $\mathrm{erf}^{-1} : [-1, 1] \to \mathbb{R}$ is then implicitly given by

$$\mathrm{erf}\left(\mathrm{erf}^{-1}(x)\right) = x.$$

THE ONE-HOT FUNCTION. The one-hot function $\mathrm{OneHot} : \mathcal{K} \to \Delta^K$ maps a category $k \in \mathcal{K}$ of a finite set, e.g., $\mathcal{K} = \{1, \ldots, K\}$, of K categories to the corresponding unit vector; i.e.,

$$\mathrm{OneHot}(k) = \mathbf{e}_k,$$

where $\mathbf{e}_k$ is the kth unit vector; i.e.,

$$(\mathbf{e}_k)_i = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{else.} \end{cases}$$

THE LOGISTIC FUNCTION. The logistic function $\sigma : \mathbb{R} \to [0, 1]$ is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

THE BINARY ENTROPY FUNCTION. The binary entropy function $H_B : [0, 1] \to [0, 1]$ is defined as

$$H_B(x) = -x \log x - (1 - x) \log(1 - x).$$

THE MODIFIED BESSEL FUNCTIONS The modified Bessel function of the first kind $\mathrm{I}_p : \mathbb{R} \to \mathbb{R}$, with $p \in \mathbb{R}$, is defined as

$$\mathrm{I}_p(x) = \sum_{m=0}^{\infty} \frac{1}{m! \Gamma(m + p + 1)} \left(\frac{x}{2}\right)^{2m+p}.$$

The modified Bessel function of the second kind $\mathrm{K}_p : \mathbb{R} \to \mathbb{R}$, with $p \in \mathbb{R}_{\geq 0}$, is defined as

$$\mathrm{K}_p(x) = \frac{\pi}{2} \frac{I_{-p}(x) - I_p(x)}{\sin(p\pi)}.$$

B

APPENDIX B: FURTHER RESULTS

B.1 BACKPROPAGATION FOR THE INPUT OF A NEURAL NETWORK

In this section, we derive the gradient and Hessian w.r.t. the input of a neural network, using backpropagation.

Consider a H-Layer deep neural network parametrizing a function $f(\mathbf{x})$ as

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{W}^H \mathbf{z}^{H-1} + \vartheta^H \\ \mathbf{z}^h &= \sigma(\mathbf{a}^h), \quad h = 0, \dots, H-1 \\ \mathbf{a}^h &= \mathbf{W}^h \mathbf{z}^{h-1} + \boldsymbol{\vartheta}^h, \quad h = 1, \dots, H-1 \\ \mathbf{a}^0 &= \mathbf{W}^0 \mathbf{x} + \boldsymbol{\vartheta}^0, \end{aligned}$$

with input $\mathbf{x}$, weights $\mathbf{W}^h$, biases $\boldsymbol{\vartheta}^h$, and activation function σ. Component wise, the equations are given by

$$\begin{aligned} z_k^h &= \sigma(a_k^h), \quad h = 0, \dots, H-1 \\ a_k^h &= \sum_n w_{kn}^h z_n^{h-1} + \vartheta_k^h, \quad h = 1, \dots, H-1 \\ a_k^0 &= \sum_n w_{kn}^0 x_n + \vartheta_k^0. \end{aligned}$$

B.1.1 *The Gradient Computation*

Next we calculate the gradient w.r.t. the input; i.e., $\partial_{\mathbf{x}} f(\mathbf{x})$. The component wise calculation yield

$$\begin{aligned} \partial_{x_i} f(\mathbf{x}) &= \partial_{x_i} \sum_n w_n^H z_n^{H-1} + \vartheta^H \\ &= \sum_n w_n^H \partial_{x_i} z_n^{H-1}, \end{aligned}$$

where we compute

$$\partial_{x_i} z_k^h = \sigma'(a_k^h) \sum_n w_{kn}^h \partial_{x_i} z_n^{h-1},$$

and σ' denotes the derivative of the activation function σ. For the first layer we compute

$$\partial_{x_i} z_k^0 = \sigma'(a_k^0) w_{ki}^0.$$

We define the messages for the partial derivative as $m_{ki}^h \coloneqq \partial_{x_i} z_k^h$. Therefore, we calculate the message passing as

$$m_{ki}^h = \sigma'(a_k^h) \sum_n w_{kn}^h m_{ni}^{h-1}$$

and

$$m_{ki}^0 = \sigma'(a_k^0) w_{ki}^0.$$

Thus, the final equations for the backpropagation of the gradient are given by

$$\boxed{\begin{aligned} & m_{ki}^0 = \sigma'(a_k^0) w_{ki}^0 \\ & \text{For } h = 1, \ldots, H-1 \\ & m_{ki}^h = \sigma'(a_k^h) \sum_n w_{kn}^h m_{ni}^{h-1} \\ & \partial_{x_i} f(\mathbf{x}) = \sum_n w_n^H m_{ni}^{H-1}. \end{aligned}}$$

B.1.2 *The Hessian Computation*

For the Hessian, we compute

$$\begin{aligned} \partial_{x_j} \partial_{x_i} f(\mathbf{x}) &= \partial_{x_j} \{ \sum_n w_n^H m_{ni}^{H-1} \} \\ &= \sum_n w_n^H \partial_{x_j} m_{ni}^{H-1}. \end{aligned}$$

The partial derivatives of the messages are

$$\partial_{x_j} m_{ki}^h = \sigma''(a_k^h) \left(\sum_n w_{kn}^h m_{ni}^{h-1} \right) \left(\sum_n w_{kn}^h m_{nj}^{h-1} \right) + \sigma'(a_k^h) \sum_n w_{kn}^h \partial_{x_j} m_{ni}^{h-1},$$

where σ'' denotes the second derivative of the activation function σ. For the first layer we compute

$$\partial_{x_j} m_{ki}^0 = \sigma''(a_k^0) w_{ki}^0 w_{kj}^0.$$

Next, we define the messages for the second order partial derivative as $\tilde{m}_{kij}^h \coloneqq \partial_{x_j} m_{ki}^h = \partial_{x_j} \partial_{x_i} z_k^h$. Therefore, we calculate the message passing as

$$\tilde{m}_{kij}^h = \sigma''(a_k^h) \left(\sum_n w_{kn}^h m_{ni}^{h-1} \right) \left(\sum_n w_{kn}^h m_{nj}^{h-1} \right) + \sigma'(a_k^h) \sum_n w_{kn}^h \tilde{m}_{nij}^{h-1}$$

and

$$\tilde{m}^0_{kij} = \sigma''(a^0_k) w^0_{ki} w^0_{kj}.$$

Finally, the equations for the backpropagation of the Hessian are given by

$$\tilde{m}^0_{kij} = \sigma''(a^0_k) w^0_{ki} w^0_{kj}$$

For $h = 1, \ldots, H-1$

$$\tilde{m}^h_{kij} = \sigma''(a^h_k) \left(\sum_n w^h_{kn} m^{h-1}_{ni} \right) \left(\sum_n w^h_{kn} m^{h-1}_{nj} \right) + \sigma'(a^h_k) \sum_n w^h_{kn} \tilde{m}^{h-1}_{nij}$$

$$\partial_{x_j} \partial_{x_i} f(\mathbf{x}) = \sum_n w^H_n \tilde{m}^{H-1}_{nij}.$$

SYMBOL	DESCRIPTION
$\mathbb{N}$	The set of natural numbers.
$\mathbb{N}_{>i}$, $\mathbb{N}_{\geq i}$	The set of natural numbers with elements greater/ greater or equal than i.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}_{>t}$, $\mathbb{R}_{\geq t}$	The set of real numbers with elements greater/ greater or equal than t.
Δ^n	The n-dimensional probability simplex; i.e., $\Delta^n = \{\mathbf{x} \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1 \wedge x_j \geq 0, \forall j \in \{1, \ldots, n\}\}$.
$\mathbb{1}(\cdot)$	The indicator function.
$f(x)$	A function f of a variable x.
$J[f]$	A functional J of a function f.
$\nabla_x(\cdot)$ or $\frac{\partial}{\partial x}(\cdot)$	The gradient w.r.t. to x as column or row vector, respectively.
$\hat{\nabla}_x(\cdot)$	The natural gradient w.r.t. to x.
$\frac{\partial^2}{\partial x^2}(\cdot)$	Second order derivative or Hessian w.r.t. to x.
$\frac{\delta}{\delta f}(\cdot)$	The functional derivative w.r.t. to the function f.
$\mathsf{P}(\cdot)$	A probability measure.
$P(\cdot)$	A cumulative distribution function.
$\bar{P}(\cdot)$	A complementary cumulative distribution function.
$p(\cdot)$	A probability density function or probability mass function.
$\mathsf{E}[\cdot]$	The expectation operator.
$\mathsf{Var}[\cdot]$	The variance operator.
$\mathsf{Cov}[\cdot]$	The covariance operator.
$\mathsf{H}[\cdot]$	The entropy operator.
$\mathsf{KL}(p(x) \parallel q(x))$	The Kullback–Leibler divergence between $p(x)$ and $q(x)$.
$\mathsf{L}[q]$	The evidence lower bound dependent on the distribution q.
$\mathrm{Gam}(\cdot \mid \alpha, \beta)$	Probability density function of the gamma distribution with shape parameter α and rate parameter β.
$\mathrm{Cat}(\cdot \mid \mathbf{p})$	Probability mass function of the categorical distribution with probability vector $\mathbf{p}$.
$\mathrm{Exp}(\cdot \mid \lambda)$	Probability density function of the exponential distribution with rate parameter λ.
$\mathrm{CRP}(\cdot \mid \mathbf{k}, \alpha)$	Probability mass function of the Chinese restaurant process distribution with assignment parameter vector $\mathbf{k}$ and concentration parameter α.

SYMBOL	DESCRIPTION
$\text{Lognorm}(\cdot \mid \mu, \sigma^2)$	Probability density function of the log-normal distribution with mean parameter μ and variance parameter σ^2.
$\text{Uniform}(\cdot \mid a, b)$	Probability density function of the uniform distribution with lower bound a and upper bound b.
$\mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Probability density function of the (multivariate) normal (or Gaussian) distribution with mean parameter $\boldsymbol{\mu}$ and variance/covariance matrix $\boldsymbol{\Sigma}$.
$\mathcal{GIG}(\cdot \mid p, a, b)$	Probability density function of the generalized inverse Gaussian distribution with index parameter p and parameters a and b corresponding to the concentration $\sqrt{ab}$ and the scale b/a.
$\text{Bin}(\cdot \mid N, p)$	Probability mass function of the binomial distribution with number of trials N and success probability p.
$\text{Mult}(\cdot \mid N, \mathbf{p})$	Probability mass function of the multinomial distribution with number of trials N and probability vector $\mathbf{p}$.
$\text{Dir}(\cdot \mid \boldsymbol{\alpha})$	Probability density function of the Dirichlet distribution with concentration parameter vector $\boldsymbol{\alpha}$.
$\text{PG}(\cdot \mid a, b)$	Probability density function of the Pólya-Gamma distribution with shape parameter a and tilting parameter b.
$\text{Pois}(\cdot \mid \lambda)$	Probability mass function of the Poisson distribution with rate parameter λ.
$\mathcal{PP}(\cdot \mid \lambda)$	Probability measure of the Poisson process with rate parameter λ.

ACRONYMS

ABR	adaptive bitrate
BAMCP	Bayes-adaptive Monte Carlo planning
BAMDP	Bayes-adaptive Markov decision process
BBA	buffer-based adaptation
BEETLE	Bayesian exploration exploitation tradeoff in learning
BOLA	buffer occupancy-based Lyapunov algorithm
BRL	Bayesian reinforcement learning
CBA	contextual-based adaptation
CCDF	complementary cumulative distribution function
CDF	cumulative distribution function
CDN	content delivery network
CGP-UCB	contextual Gaussian process upper confidence bound
CRP	Chinese restaurant process
CTMC	continuous-time Markov chain
DASH	dynamic adaptive streaming over HTTP
ELBO	evidence lower bound
GIG	generalized inverse Gaussian
GP	Gaussian process
HJB	Hamilton-Jacobi-Bellman
HTTP	hypertext transfer protocol
IP	internet protocol
ICP	interest control protocol
IRL	inverse reinforcement learning
ISO	international organization for standardization
KL	Kullback–Leibler
LQ	linear quadratic
LQG	linear quadratic Gaussian
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
MDP	Markov decision process
MFT	multiple filter test
MOS	mean opinion score
MPD	media presentation description
NDN	named data networking
NFD	NDN forwarding daemon
OC	optimal control
OS-SVI	one-step stochastic variational inference
PANDA	probe and adapt
PDE	partial differential equation
PDF	probability density function
PGM	probabilistic graphical model
PG	Pólya-Gamma
PMF	probability mass function
POMDP	partially observable Markov decision process
POSMDP	partially observable semi-Markov decision process
PSRL	posterior sampling for reinforcement learning
QoE	quality of experience
QoS	quality of service
RL	reinforcement learning
RTT	round trip time
RV	random variable
SAND	server and network assisted DASH
SDE	stochastic differential equation
SMDP	semi-Markov decision process
SOC	stochastic optimal control
SVI	stochastic variational inference
TBA	throughput-based adaptation
TCP	transmission control protocol
TD	temporal difference
TPBN	three parameter beta normal
UCB	upper confidence bound
VB	variational Bayes
VEM	variational expectation-maximization
VI	variational inference
XML	extensible markup language

BIBLIOGRAPHY

[1] B. Alt, M. Schultheis, and H. Koeppl, "POMDPs in continuous time and discrete spaces", in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 13 151–13 162.

[2] B. Alt, A. Šošić, and H. Koeppl, "Correlation priors for reinforcement learning", in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 14 155–14 165.

[3] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual quality adaptation for adaptive bitrate video streaming", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1000–1008.

[4] B. Alt, M. Messer, J. Roeper, G. Schneider, and H. Koeppl, "Non-parametric Bayesian inference for change point detection in neural spike trains", in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, 2018, pp. 258–262.

[5] L. Köhs, B. Alt, and H. Koeppl, "Variational inference for continuous-time switching dynamical systems", in *Advances in Neural Information Processing Systems*, vol. 34, 2021, to appear, arXiv:2109.14492 [cs.LG].

[6] W. R. KhudaBukhsh, B. Alt, S. Kar, A. Rizk, and H. Koeppl, "Collaborative uploading in heterogeneous networks: Optimal and adaptive strategies", in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 1–9.

[7] B. Alt, M. Weckesser, C. Becker, M. Hollick, S. Kar, A. Klein, R. Klose, R. Kluge, H. Koeppl, B. Koldehofe, *et al.*, "Transitions: A protocol-independent view of the future internet", *Proceedings of the IEEE*, vol. 107, no. 4, pp. 835–846, 2019.

[8] S. Kar, R. Rehrmann, A. Mukhopadhyay, B. Alt, F. Ciucu, H. Koeppl, C. Binnig, and A. Rizk, "On the throughput optimization in large-scale batch-processing systems", *Performance Evaluation*, vol. 144, p. 102 142, 2020.

[9] W. R. KhudaBukhsh, S. Kar, B. Alt, A. Rizk, and H. Koeppl, "Generalized cost-based job scheduling in very large heterogeneous cluster systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2594–2604, 2020.

[10] J. Machkour, M. Muma, B. Alt, and A. M. Zoubir, "A robust adaptive Lasso estimator for the independent contamination model", *Signal Processing*, vol. 174, p. 107 608, 2020.

[11] J. Machkour, B. Alt, M. Muma, and A. M. Zoubir, "The outlier-corrected-data-adaptive Lasso: A new robust estimator for the independent contamination model", in *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, 2017, pp. 1649–1653.

[12] A. Feldbaum, "Dual control theory. i", *Avtomatika i Telemekhanika*, vol. 21, no. 9, pp. 1240–1249, 1960.

[13] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 1994.

[14] M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, *et al.*, "Bayesian reinforcement learning: A survey", *Foundations and Trends in Machine Learning*, vol. 8, no. 5-6, pp. 359–483, 2015.

[15] S. L. Lauritzen, *Graphical models*. Clarendon Press, 1996, vol. 17.

[16] L. Dietz, "Directed factor graph notation for generative models", Max Planck Inst. Informatics, Tech. Rep., 2010.

[17] S. Särkkä and A. Solin, *Applied stochastic differential equations*. Cambridge University Press, 2019, vol. 10.

[18] B. Øksendal, "Stochastic differential equations", in *Stochastic differential equations*, Springer, 2003, pp. 65–84.

[19] J. R. Norris, *Markov chains* (Cambridge Series in Statistical and Probabilistic Mathematics 2). Cambridge university press, 1998.

[20] P. Del Moral and S. Penev, *Stochastic Processes: From Applications to Theory*. Chapman and Hall/CRC, 2017.

[21] F. B. Hanson, *Applied stochastic processes and control for jump-diffusions: Modeling, analysis and computation*. SIAM, 2007.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer, 2006.

[23] L. Wasserman, *All of statistics: A concise course in statistical inference*. Springer, 2004, vol. 26.

[24] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[25] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning", *Machine learning*, vol. 50, no. 1, pp. 5–43, 2003.

[26] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians", *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[27] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 2.

[28] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[29] S. Stidham and R. Weber, "A survey of Markov decision models for control of networks of queues", *Queueing systems*, vol. 13, no. 1, pp. 291–314, 1993.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[31] D. Liberzon, *Calculus of variations and optimal control theory*. Princeton university press, 2011.

[32] L. S. Pontryagin, *The mathematical theory of optimal processes*. John Wiley, 1962.

[33] R. Bellman, "Dynamic programming", *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[34] W. H. Fleming and H. M. Soner, *Controlled Markov processes and viscosity solutions*. Springer Science & Business Media, 2006, vol. 25.

[35] D. Bertsekas, *Reinforcement and Optimal Control*. Athena Scientific, 2019.

[36] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing networks and Markov chains: Modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[37] C. Donner and M. Opper, "Efficient bayesian inference of sigmoidal gaussian cox processes", *Journal of Machine Learning Research*, vol. 19, pp. 1–34, 2018.

[38] S.-H. Lee and Y. Dan, "Neuromodulation of brain states", *Neuron*, vol. 76, no. 1, pp. 209–222, 2012.

[39] C. D. Brody, "Correlations without synchrony", *Neural computation*, vol. 11, no. 7, pp. 1537–1551, 1999.

[40] S. Grün, A. Riehle, and M. Diesmann, "Effect of cross-trial nonstationarity on joint-spike events", *Biological cybernetics*, vol. 88, no. 5, pp. 335–351, 2003.

[41] F. Stimberg, A. Ruttor, and M. Opper, "Poisson process jumping between an unknown number of rates: Application to neural spike data", in *Advances in Neural Information Processing Systems*, 2014, pp. 730–738.

[42] S. J. Gershman and D. M. Blei, "A tutorial on Bayesian nonparametric models", *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.

[43] A. Aue and L. Horváth, "Structural breaks in time series", *Journal of Time Series Analysis*, vol. 34, no. 1, pp. 1–16, 2013.

[44] M. Messer, M. Kirchner, J. Schiemann, J. Roeper, R. Neininger, and G. Schneider, "A multiple filter test for change point detection in renewal processes with varying variance", *The Annals of Applied Statistics*, vol. 8, no. 4, pp. 2027–2067, 2014.

[45] M. Messer, K. M. Costa, J. Roeper, and G. Schneider, "Multi-scale detection of rate changes in spike trains with weak dependencies", *Journal of computational neuroscience*, vol. 42, no. 2, pp. 187–201, 2017.

[46] A.-C. Camproux, F. Saunier, G. Chouvet, J.-C. Thalabard, and G. Thomas, "A hidden Markov model approach to neuron firing patterns", *Biophysical journal*, vol. 71, no. 5, pp. 2404–2412, 1996.

[47] S. Tokdar, P. Xi, R. C. Kelly, and R. E. Kass, "Detection of bursts in extracellular spike trains using hidden semi-markov point process models", *Journal of computational neuroscience*, vol. 29, no. 1, pp. 203–212, 2010.

[48] R. J. Boys, D. J. Wilkinson, and T. B. Kirkwood, "Bayesian inference for a discretely observed stochastic kinetic model", *Statistics and Computing*, vol. 18, no. 2, pp. 125–135, 2008.

[49] B. Pradhan and D. Kundu, "Bayes estimation and prediction of the two-parameter gamma distribution", *Journal of Statistical Computation and Simulation*, vol. 81, no. 9, pp. 1187–1198, 2011.

[50] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual quality adaptation for adaptive bitrate video streaming (extended version)", *arXiv preprint arXiv:1901.05712*, 2019.

[51] *Information technology, dynamic adaptive streaming over HTTP (DASH), part 5: Server and network assisted DASH (SAND)*, ISO/IEC 23009-5:2017, Standard, Geneva, CH, Feb. 2017.

[52] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi, "Dynamic adaptive video streaming: Towards a systematic comparison of ICN and TCP/IP", *IEEE transactions on Multimedia*, vol. 19, no. 10, pp. 2166–2181, 2017.

[53] M. K. Mukerjee, I. N. Bozkurt, B. Maggs, S. Seshan, and H. Zhang, "The impact of brokers on the future of content delivery", in *proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 127–133.

[54] A. Bentaleb, A. C. Begen, and R. Zimmermann, "SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking", in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1296–1305.

[55] L. Wang, I. Moiseenko, and D. Wang, "When video streaming meets named data networking: A case study", in *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2016, pp. 166–173.

[56] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation", in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[57] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire, "Taming the monster: A fast and simple algorithm for contextual bandits", in *International Conference on Machine Learning*, PMLR, 2014, pp. 1638–1646.

[58] I. Urteaga and C. Wiggins, "Variational inference for the multi-armed contextual bandit", in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 698–706.

[59] H. Ishwaran and J. S. Rao, "Spike and slab variable selection: Frequentist and Bayesian strategies", *The Annals of Statistics*, vol. 33, no. 2, pp. 730–773, 2005.

[60] A. Armagan, D. B. Dunson, and M. Clyde, "Generalized beta mixtures of Gaussians", in *Advances in neural information processing systems*, vol. 24, NIH Public Access, 2011, p. 523.

[61] C. M. Carvalho, N. G. Polson, and J. G. Scott, "Handling sparsity via the horseshoe", in *Artificial Intelligence and Statistics*, PMLR, 2009, pp. 73–80.

[62] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference.", *Journal of Machine Learning Research*, vol. 14, no. 5, 2013.

[63] H. Robbins and S. Monro, "A stochastic approximation method", *The annals of mathematical statistics*, pp. 400–407, 1951.

[64] S.-I. Amari, "Natural gradient works efficiently in learning", *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[65] H. Robbins, "Some aspects of the sequential design of experiments", *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[66] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems", *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[67] H. Bastani and M. Bayati, "Online decision making with high-dimensional covariates", *Operations Research*, vol. 68, no. 1, pp. 276–294, 2020.

[68] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs", in *International Conference on Machine Learning*, PMLR, 2013, pp. 127–135.

[69] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs", *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.

[70] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions", in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 208–214.

[71] A. Krause and C. S. Ong, "Contextual Gaussian process bandit optimization", in *Advances in Neural Information Processing Systems*, 2011, pp. 2447–2455.

[72] E. Kaufmann, O. Cappé, and A. Garivier, "On Bayesian upper confidence bounds for bandit problems", in *International Conference on Artificial Intelligence and Statistics (AISTAT)*, 2012, pp. 592–600.

[73] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for machine learning*. MIT Press, 2005.

[74] D. Stohr, A. Frömmgen, A. Rizk, M. Zink, R. Steinmetz, and W. Effelsberg, "Where are the sweet spots? a systematic approach to reproducible DASH player comparisons", in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1113–1121.

[75] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale", *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[76] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos", in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, IEEE, 2016, pp. 1–9.

[77] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP", in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[78] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives", *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016.

[79] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve", in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[80] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video ABR algorithms to network conditions", in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.

[81] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "Want to play DASH? A game theoretic approach for adaptive streaming over HTTP", in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 13–26.

[82] C. Wang, D. Bhat, A. Rizk, and M. Zink, "Design and analysis of QoE-aware quality adaptation for DASH: A spectrum-based approach", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 3s, pp. 1–24, 2017.

[83] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation", in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[84] S. Gupta, G. Joshi, and O. Yağan, "Correlated multi-armed bandits with a latent random source", *arXiv e-prints*, arXiv–1808, 2018.

[85] M. Hoffman, B. Shahriari, and N. Freitas, "On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning", in *Artificial Intelligence and Statistics*, 2014, pp. 365–374.

[86] G. Chalkiadakis and C. Boutilier, "Coordination in multiagent reinforcement learning: A Bayesian approach", in *International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, 2003, pp. 709–716.

[87] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives", in *Advances in Neural Information Processing Systems*, 2013, pp. 2616–2624.

[88] J. D. Lafferty and D. M. Blei, "Correlated topic models", in *Advances in Neural Information Processing Systems*, 2006, pp. 147–154.

[89] S. W. Linderman, M. J. Johnson, and R. P. Adams, "Dependent multinomial models made easy: Stick breaking with the Pólya-Gamma augmentation", in *Advances in Neural Information Processing Systems*, vol. 2015, 2015, pp. 3456–3464.

[90] M. K. Titsias, "The infinite Gamma-Poisson feature model", in *Advances in Neural Information Processing Systems*, 2008, pp. 1513–1520.

[91] F. Wenzel, T. Galy-Fajou, C. Donner, M. Kloft, and M. Opper, "Efficient Gaussian process classification using Pòlya-Gamma data augmentation", in *AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5417–5424.

[92] P. Ranchod, B. Rosman, and G. Konidaris, "Nonparametric Bayesian reward segmentation for skill discovery using inverse reinforcement learning", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 471–477.

[93] A. Šošic, A. M. Zoubir, E. Rueckert, J. Peters, and H. Koeppl, "Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling", *Journal of Machine Learning Research*, vol. 19, no. 69, pp. 1–45, 2018.

[94] A. Šošić, A. M. Zoubir, and H. Koeppl, "Policy recognition via expectation maximization", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4801–4805.

[95] A. Šošić, A. M. Zoubir, and H. Koeppl, "A Bayesian approach to policy recognition and state representation learning", *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1295–1308, 2017.

[96] Y. Engel, S. Mannor, and R. Meir, "Bayes meets Bellman: The Gaussian process approach to temporal difference learning", in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 154–161.

[97] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search", in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, Citeseer, 2011, pp. 465–472.

[98] T. Hester and P. Stone, "Texplore: Real-time sample-efficient reinforcement learning for robots", *Machine learning*, vol. 90, no. 3, pp. 385–429, 2013.

[99] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning", in *International Conference on Machine Learning*, ACM, 2006, pp. 697–704.

[100] A. Guez, D. Silver, and P. Dayan, "Efficient Bayes-adaptive reinforcement learning using sample-based search", in *Advances in Neural Information Processing Systems*, 2012, pp. 1025–1033.

[101] M. Pavlov and P. Poupart, "Towards global reinforcement learning", in *NIPS Workshop on Model Uncertainty and Risk in Reinforcement Learning*, 2008.

[102] H. Ishwaran and L. F. James, "Gibbs sampling methods for stick-breaking priors", *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 161–173, 2001.

[103] N. G. Polson, J. G. Scott, and J. Windle, "Bayesian inference for logistic models using Pólya-Gamma latent variables", *Journal of the American statistical Association*, vol. 108, no. 504, pp. 1339–1349, 2013.

[104] K. P. Murphy, *Machine learning: A probabilistic perspective*. MIT Press, 2012.

[105] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, "Robust and efficient transfer learning with hidden parameter Markov decision processes", in *Advances in Neural Information Processing Systems*, 2017, pp. 6250–6261.

[106] M. O. Duff, "Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes", Ph.D. dissertation, University of Massachusetts Amherst, 2002.

[107] I. Osband, B. Van Roy, and D. Russo, "(More) efficient reinforcement learning via posterior sampling", in *Advances in Neural Information Processing Systems*, 2013, pp. 3003–3011.

[108] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling", *arXiv preprint arXiv:1707.02038*, 2017.

[109] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning.", in *International Conference on Machine Learning*, 2000, pp. 663–670.

[110] C. A. Rothkopf and C. Dimitrakakis, "Preference elicitation and inverse reinforcement learning", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2011, pp. 34–48.

[111] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills", *arXiv preprint arXiv:1802.09564*, 2018.

[112] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains", *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[113] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[114] C. Wang, D. Blei, and D. Heckerman, "Continuous time dynamic topic models", in *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2008, pp. 579–586.

[115] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process", in *Advances in Neural Information Processing Systems*, 2017, pp. 6754–6764.

[116] L. Huang, L. Pauleve, C. Zechner, M. Unger, A. S. Hansen, and H. Koeppl, "Reconstructing dynamic molecular states from single-cell time series", *Journal of The Royal Society Interface*, vol. 13, no. 122, p. 20 160 533, 2016.

[117] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[118] C. Tallec, L. Blier, and Y. Ollivier, "Making deep q-learning methods robust to time discretization", in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, Long Beach, California, USA: PMLR, 2019, pp. 6096–6104.

[119] R. Kalman, "The theory of optimal control and the calculus of variations", in *Mathematical optimization techniques*, University of California Press Los Angeles, CA, 1963, pp. 309–331.

[120] K. Doya, "Reinforcement learning in continuous time and space", *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[121] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, "A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems", *Automatica*, vol. 49, no. 1, pp. 82–92, 2013.

[122] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem", *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[123] L. C. Baird, "Reinforcement learning in continuous time: Advantage updating", in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, IEEE, vol. 4, 1994, pp. 2448–2453.

[124] M. E. Harmon and L. C. Baird III, "Multi-player residual advantage learning with general function approximation", *Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH*, pp. 45 433–7308, 1996.

[125] S. J. Bradtke and M. O. Duff, "Reinforcement learning methods for continuous-time Markov decision problems", in *Advances in neural information processing systems*, 1995, pp. 393–400.

[126] H. Xie, Y. Li, and J. C. Lui, "Optimizing discount and reputation trade-offs in e-commerce systems: Characterization and online learning", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7992–7999.

[127] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning", *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[128] A. Jain and D. Precup, "Eligibility traces for options", in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1008–1016.

[129] C. G. Cassandras and J. Lygeros, *Stochastic hybrid systems*. CRC Press, 2018.

[130] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs", in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[131] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online POMDP planning with regularization", *Journal of Artificial Intelligence Research*, vol. 58, pp. 231–266, 2017.

[132] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, "Deep variational reinforcement learning for POMDPs", in *International Conference on Machine Learning*, 2018, pp. 2117–2126.

[133] P. Chaudhari, S. Karaman, D. Hsu, and E. Frazzoli, "Sampling-based algorithms for continuous-time POMDPs", in *2013 American Control Conference*, IEEE, 2013, pp. 4604–4610.

[134] S. Mahadevan, "Partially observable semi-Markov decision processes: Theory and applications in engineering and cognitive science", in *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes, Orlando, FL, USA*, 1998, pp. 113–120.

[135] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Graph-based cross entropy method for solving multi-robot decentralized POMDPs", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 5395–5402.

[136] R. Srinivasan and A. K. Parlikad, "Semi-Markov decision process with partial information for maintenance decisions", *IEEE Transactions on Reliability*, vol. 63, no. 4, pp. 891–898, 2014.

[137] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach", *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 411–440, 1968.

[138] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming", in *Advances in neural information processing systems*, 2008, pp. 1465–1472.

[139] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory", *Journal of statistical mechanics: Theory and experiment*, vol. 2005, no. 11, P11011, 2005.

[140] E. A. Theodorou and E. Todorov, "Stochastic optimal control for nonlinear Markov jump diffusion processes", in *2012 American Control Conference (ACC)*, IEEE, 2012, pp. 1633–1639.

[141] A. Simpkins and E. Todorov, "Practical numerical methods for stochastic optimal control of biological systems in continuous time and space", in *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, IEEE, 2009, pp. 212–218.

[142] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[143] Y. Tassa and T. Erez, "Least squares solutions of the HJB equation with neural network value-function approximators", *IEEE transactions on neural networks*, vol. 18, no. 4, pp. 1031–1041, 2007.

[144] M. Lutter, B. Belousov, K. Listmann, D. Clever, and J. Peters, "HJB optimal feedback control with deep differential value functions and action constraints", in *Conference on Robot Learning*, PMLR, 2020, pp. 640–650.

[145] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning", *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018.

[146] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations", *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[147] W. M. Wonham, "Some applications of stochastic differential equations to optimal nonlinear filtering", *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 2, no. 3, pp. 347–369, 1964.

[148] R. Van Handel, "Filtering, stability, and robustness", Ph.D. dissertation, California Institute of Technology, 2007.

[149] H. J. Kushner, "On the differential equations satisfied by conditional probablitity densities of Markov processes, with applications", *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 2, no. 1, pp. 106–119, 1964.

[150] P. W. Lewis and G. S. Shedler, "Simulation of nonhomogeneous Poisson processes by thinning", *Naval research logistics quarterly*, vol. 26, no. 3, pp. 403–413, 1979.

[151] M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian networks: Using physics as model prior for deep learning", in *International Conference on Learning Representations*, 2019.

[152] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning", in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, 2016, pp. 1995–2003.

[153] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[154] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains", in *AAAI*, vol. 94, 1994, pp. 1023–1028.

[155] A. R. Cassandra, "Exact and approximate algorithms for partially observable Markov decision processes", Ph.D. dissertation, Department of Computer Science, Brown University, Providence, RI, 1998.

[156] C. Wildner and H. Koeppl, "Moment-based variational inference for Markov jump processes", in *International Conference on Machine Learning*, 2019, pp. 6766–6775.

[157] F. Doshi-Velez, "The infinite partially observable Markov decision process", in *Advances in neural information processing systems*, vol. 22, 2009, pp. 477–485.

[158] E. Çinlar, *Probability and stochastics*. Springer, 2011, vol. 261.

CURRICULUM VITÆ

BASTIAN ALT

PERSONAL INFORMATION

DATE OF BIRTH	November 3, 1989
PLACE OF BIRTH	Erbach (Odenwald)
HOMEPAGE	https://github.com/bastianalt

EDUCATION

Master of Science (M.Sc.) — 10/2013 – 12/2016
Electrical Engineering and Information Technology
Technische Universität Darmstadt, Darmstadt, Germany

Bachelor of Science (B.Sc.) — 10/2010 – 10/2013
Electrical Engineering and Information Technology
Technische Universität Darmstadt, Darmstadt, Germany

Abitur (General Qualification for University Entrance) — 06/2009
Max Planck Schule, Groß-Umstadt, Germany

WORK EXPERIENCE

Research Associate — since 01/2017
Bioinspired Communication Systems
Technische Universität Darmstadt, Darmstadt, Germany

Internship — 10/2015 – 03/2016
Daimler AG, Sindelfingen, Germany

Civil Service 10/2009 – 06/2010
Johanniter-Unfall-Hilfe e.V., RV Darmstadt-Dieburg
Dieburg, Germany

ERKLÄRUNG LAUT PROMOTIONSORDNUNG

§ 8 Abs. 1 lit. c PromO
Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO
Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO
Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO
Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 18. Oktober 2021