Scalable Loss-calibrated Bayesian Decision Theory and Preference Learning

M. Ehsan Abbasnejad

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

April 2017

Declaration

I hereby declare that this thesis is my original work which has been done in collaboration with other researchers. This document has not been submitted for any other degree or award in any other university or educational institution. Parts of this thesis have been published in collaboration to other researchers in international conferences as listed below:

- **(Chapter 3)** E. Abbasnejad, S. Sanner, E. V. Bonilla, P. Poupart, Learning Community-based Preferences via Dirichlet Process Mixtures of Gaussian Processes, *In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013. Beijing, China.
- **(Chapter 4)** E. Abbasnejad, E. V. Bonilla, S. Sanner, Decision-theoretic Sparsification for Gaussian Process Preference Learning, *Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference (ECML PKDD)*, 2013. Prague, Czech Republic.
- **(Chapter 5)** E. Abbasnejad, J. Domke, S. Sanner, Loss-calibrated Monte Carlo Action Selection, *In Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, 2015. Austin, USA.

M. Ehsan Abbasnejad
April 26, 2017

to my parents.

Acknowledgments

I would like to express my gratitude towards all who helped me make this thesis possible.

First and foremost, I wish to thank my supervisor **Scott Sanner** for all his guidance, caring and patience and providing me with the excellent atmosphere for doing research. Scott's intuitive understanding of Bayesian methods and decision theory motivated my research and gave me insights that would have never been possible without him. His encouragements have been invaluable to me during the hard times. I am extremely grateful for his strong support in my academic and personal life. I also wish to thank my advisor **Edwin V. Bonilla**, who gave me the opportunity to learn Gaussian processes. Special thanks to **Justin Domke** whose advice on mathematical formulation and problem solving has been invaluable. His deep and intuitive understanding has been a great help and motivation. I would like to express my warm respects towards **Mark Reid** as my advisor, for his time and helpful suggestions. I would also like to thank **Pascal Poupart** who gave me a better insight into Nonparametric Bayesian methods during my visit. I am eternally in debt to these academic role models.

I acknowledge the financial, academic and technical support provided by the National ICT of Australia (NICTA) and the Australian National University (ANU) and thank them for their support in my research. My fellow postgraduate friends at NICTA and the ANU provided a welcoming collaborative environment. I would like to specially thank Zahra Zamani, Alireza Motevalian, Sarah Taghavi Namin, Mohammad Ismailzadeh, Mohammad Najafi, Ehsan Nabavi, Hadi Afshar, Behrouz Nasihatkon, Salim Masoumi, Alireza Khosravian, Mehdi Shabani, MohammaReza Jouyandeh, Fateme Rajabi, Morteza Azad, Sahba Najafi, Mohsen Zamani, Mona Golestar Far, Suvash Sedhain and many others who always cheered me on and made my life so joyful.

Finally I have to thank my parents who have always supported me and gave me the encouragement to follow my dreams. I thank them for believing in me and for their endless love and support. My brothers, Iman and Amin, I thank you for always being my best friends and supportive of whatever I have done.

Above all I wish to thank the almighty God for his guidance in life and giving me the opportunity to pursue my degree.

Abstract

Bayesian decision theory provides a framework for optimal action selection under uncertainty given a utility function over actions and world states and a distribution over world states. The application of Bayesian decision theory in practice is often limited by two problems: (1) in application domains such as recommendation, the true utility function of a user is *a priori* unknown and must be learned from user interactions; and (2) computing expected utilities under complex state distributions and (potentially uncertain) utility functions is often computationally expensive and requires tractable approximations.

In this thesis, we aim to address both of these problems. For (1), we take a Bayesian non-parametric approach to utility function modeling and learning. In our first contribution, we exploit community structure prevalent in collective user preferences using a Dirichlet Process mixture of Gaussian Processes (GPs). In our second contribution, we take the underlying GP preference model of the first contribution and show how to jointly address both (1) and (2) by sparsifying the GP model in order to preserve optimal decisions while ensuring tractable expected utility computations. In our third and final contribution, we directly address (2) in a Monte Carlo framework by deriving an optimal loss-calibrated importance sampling distribution and show how it can be extended to uncertain utility representations developed in the previous contributions.

Our empirical evaluations in various applications — including multiple preference learning problems using synthetic and real user data and robotics decision-making scenarios derived from actual occupancy grid maps — demonstrate the effectiveness of the theoretical foundations laid in this thesis and pave the way for future advances that address important practical problems at the intersection of Bayesian decision theory and scalable machine learning.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Every day, people and computers are faced with decision making in uncertain circumstances. Decision theory is the study of optimal decision making under uncertainty. It is concerned with the analysis of the consequence of *actions* in an uncertain environment. Decision theory requires three major components: *state*, *action*, and the *utility function*. The utility function is a means of evaluating how valuable an action (i.e. a choice amongst alternatives) is for the decision maker in a given state of nature. The expectation of this utility for an action with respect to the uncertain states is the *expected utility* and maximizing it is the core focus of decision theory.

In this formulation, conventional decision theory assumes the utility is known to the decision maker. However, in many applications this assumption is not valid. The utility may only be formulated by or revealed to the decision maker through an interaction with the environment and may be partially known. Consider the following examples:

- **Recommendation:** The objective of a recommender system is to suggest attractive items (alternatives) to the users that potentially leads to their selection (e.g. purchase). With an abundance of consumer choices, such systems are becoming increasingly popular to direct users to what they might find useful. One way these recommenders provide these suggestions is by learning from the users' feedback that was collected from their previous preferences for items. Under certain assumptions, we can model preferences using utilities. These utilities are unknown and user's preferences provide information to help formulate belief about their true values.

- **Robotic automation:** Navigation and exploration are vital tasks in robotics and entail making choices amongst a variety of alternatives. For example an unmanned aerial vehicle (UAV) for scientific research has to maximize area covered while choosing a path, places to stop for observation, duration of stay while observing, and types of measurements [28]. These all have to be done while carefully avoiding collisions with other objects, such as moving UAVs [5]. Also, one has to consider the fact that certain actions might perform well for a mission and poorly for others. Manually determining the utility function needs a tedious investigation into the mission, the field, and possible consequences of each action. Alternatively, learning it (or improving from what was learned in a similar mission) through the collection of information from the environment can be considered.

- **Automated medical assistant:** Assessing the effectiveness of a medical treatment is generally hard and has to be tailored towards the needs of each individual patient [22]. This is because each patient might feel differently about the process and consequences of each treatment. For instance, one might prefer a short but intensive treatment over a longer, relatively comfortable one. By presenting these alternatives at a given stage to a patient we can construct a personal utility function. We might start with a "general" utility function as a prior and update it as patient's preferences are observed. In this process, we learn the utility of each patient during the course of a treatment.

- **Autonomic computing:** Managing a cluster server can be a tedious and error-prone task. It can ideally be managed automatically by the computers themselves with minimal human intervention [16, 66]. These systems are becoming crucial due to the increasing popularity of cloud computing where users can run their programs on these cloud resources and pay accordingly. An autonomic system seeks to maximize its income by intelligent allocation of all resources while ensuring certain quality constraints (such as maximum wait time in the queue) are satisfied. The utility function measures how much the allocation of a resource (e.g. CPU) to a task in the queue earns for the cloud service provider while maximizing the customer's satisfaction. This utility is unknown in advance because each application's resource requirements are not known and different. Learning the utility of resources for each application ensures a better allocation and consequently a better quality of services.

In all these examples, the utility function is initially unknown or imprecise. The value of utilities is inferred from some form of available information (e.g. preferences). With observing the alternative choices and their consequences, we formulate a "belief" in a utility function. For example in a recommender system, observing a purchase of a comedy movie increases the certainty that the user is interested in this genre and may assign higher utility values to instances of such movies.

With the prerequisite that we need to handle uncertain utilities, we treat the utility function as a random quantity. As such the arsenal of available solutions for probabilistic modeling is at our disposal. Although in some problems, the utilities are assumed to have a particular form, we are interested in a flexible way of modeling the utility function such that its capacity scales with observations. Hence, we contribute novel *nonparametric Bayesian methods* for utility learning. Nonparametric Bayesian models accommodate the needs of such problems by (1) providing an infinite dimensional basis for the problem, (2) expanding its capacity with new observations that enables multi-modal representation and (3) presenting an explicit model of uncertainty (Bayesian posterior) for its predictions.

Once the utility is formulated, finding the action to be performed requires calculation of expected utilities that leads to computationally expensive integrals. This is true if either the utility is manually formulated or learned using probabilistic modeling methods. Efficient computation of the expected utility is a challenging problem that we also investigate in this thesis. In the subsequent section, we provide a more formal description of the fundamental problems that we address in this thesis.

1.2 Basic Framework

Bayesian decision-theory [7, 32, 77] provides a formalization of robust decision-making in uncertain settings by maximizing expected utility. Formally, a utility function $u(\boldsymbol{\theta}, a)$ quantifies the return of performing an action $a \in \mathcal{A} = \{a_1, \ldots, a_k\}$ in a given state $\boldsymbol{\theta}$. When the true state is uncertain and only a belief state distribution $p(\boldsymbol{\theta})$ is known, Bayesian decision-theory posits that an optimal control action a should maximize the *expected utility*

$$\mathrm{EU}_u(a) = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta})}[u(\boldsymbol{\theta}, a)] = \int u(\boldsymbol{\theta}, a) p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{1.1}$$

where by definition, the *optimal action a^** is

$$a^* = \arg\max_a \quad \mathrm{EU}_u(a). \tag{1.2}$$

This conventional formalization assumes the utility is given and completely known. As discussed earlier, this is not always the case and utilities can be unknown or imprecise in various applications. Since we are treating the utility function as a random quantity, we have a belief in its values (technically a random function). Therefore, we are led to the framework known as *expected expected utility* [15] that takes an expectation both over state and utility uncertainty:

$$\boxed{\mathrm{EEU}(a) = \mathbb{E}_{u \sim p(u)}\left[\mathrm{EU}_u(a)\right] = \int \mathrm{EU}_u(a) p(u) du. \tag{1.3}}$$

Here u is a functional and $p(u)$ is a distribution over this functional. As expected, the optimal action is the one that maximizes this expression. For a fully known utility function, that is when the distribution of the utilities is a Dirac delta function centered at this fixed utility, we recover the conventional decision theoretic framework given in Equation 1.1.

When there is no observation in the Bayesian modeling of $p(u)$, the utility value is represented in the *prior* provided by the expert. As more observations become available, the belief in the utility is updated.

There are two major aspects to this formalization:

- **Learning a distribution over utility functions**: In practice, modeling the distribution of the utility function itself is difficult. For instance, utilities have to be inferred from pairwise comparisons in preference learning. Then the first question is how to learn a utility belief model $p(u)$ from observations?
- **Efficient computation of the optimal action:** Once the utility is learned, finding the optimal action is often expensive and needs an efficient computation of the integral in the expected utility. In this case, if we are given a fixed utility, how do we efficiently compute $\mathrm{EU}_u(a)$? On the other hand, if we have a distribution over the utility $p(u)$, then how to compute $\mathrm{EEU}(a)$?

These two issues are the main problems that will be addressed in this thesis.

1.3 Contributions

As outlined previously, the two aims of this thesis are learning a distribution over utility functions and efficient computation of the optimal action. As such, in the following we summarize our main contributions:

- **Learning Community-based Preferences via Dirichlet Process Mixtures of Gaussian Processes:** To formulate our belief in a utility function, we need a flexible model that requires minimal constraints on the structure of utilities. This model has to be able to automatically adapt to the amount of data. Our first contribution is to use nonparametric Bayesian methods, in particular Gaussian Processes (GPs) [74], to model the distribution of the utility. However, as these models can grow prohibitively large when numbers of users grow, we leverage the notation of communities to motivate nonparametric clusterings of users for an efficient representation. This efficient extension reduces cubic operations for computing the posterior to linear ones using the Dirichlet Process (DP) [85]. We provide an application of this method to preference learning and show that our proposed approach is capable of grouping users based on their preference clusters.
- **Decision-theoretic Sparsification for Gaussian Process Preference Learning:** In addition to exploring user communities for efficient inference in GPs using DPs, we further propose a decision-theoretic sparsification method for GP preference learning. Thus, as opposed to the previous approach, we use only a subset of users and items to learn the approximate posterior. This method learns an approximate sparsified distribution of the utility function where our sparsification approach is underpinned by decision theory and directly incorporates the loss function inherent in the underlying preference learning problem. We show that by selecting different specifications of the loss function several popular sparsification methods are recovered from our decision-theoretic framework. We refer to our method as the *Valuable Vector Machine* (VVM) as it selects the most “valuable” items during sparsification to minimize the corresponding loss.

- **Loss-calibrated Monte Carlo Action Selection:** Obtaining the optimal action that maximizes the expected utility requires computationally expensive integrals typically evaluated using Monte Carlo methods in both conventional Bayesian decision theory in Equation 1.1 and EEU in Equation 1.3. We provide an improved Monte Carlo method to find the optimal action in a fraction of the time compared to its conventional counterpart; To do this, we use the calculus of variations to find the optimal distribution that minimizes the *regret*, that is, we minimize the probability of suboptimal action selection when the number of samples is limited. As will be shown, this objective further minimizes the probability of suboptimal action selection in the finite sample case.

Putting these three basic contributions together we lay the foundation for efficient and scalable computation of expected expected utility for optimal Bayesian decision-theoretic action selection. Our empirical evaluations in various applications — including multiple preference learning problems using synthetic and real user data and robotics decision-making scenarios derived from actual occupancy grid maps — demonstrate the effectiveness of the theoretical foundations laid in this thesis and pave the way for future advances that address important practical problems at the intersection of Bayesian decision theory and scalable machine learning.

1.4 Thesis Outline

This chapter introduced our research problem: efficient and scalable decision making using uncertain utilities learned from preference feedback. The remainder of this thesis is structured as follows: Chapter 2 defines all the background material required to understand decision theory, graphical and Bayesian models used in this thesis as well as a review of current approaches. In Chapter 3, we discuss our approach using GPs and DPs for efficient preference learning and of modeling the utility function. Chapter 4 applies GPs to finding the distribution of the utility function for multiple users and introduces a decision theoretic formalization for sparse Gaussian process models. Having discussed approaches to modeling the distribution of the utility function, in Chapter 5 we discuss how to efficiently determine the optimal action using sampling for computation of the expected utilities. Finally, Chapter 6 concludes this thesis with a summary and interesting directions for future work.

Chapter 2

Background

As explained earlier in the introduction, decision theory studies models of decision making for agent(s) under Bayesian models of environment uncertainty. In this section we will provide the background material needed for our contributions in the rest of this thesis. In particular, we will discuss:

- What is decision theory and how it is formally defined? How can we model the uncertainty involved in the decision making process? We will formally introduce decision theory in Section 2.1 and modeling uncertainty in Section 2.1.2
- Amongst the decision theoretic frameworks, Bayesian decision theory is a compelling choice because it models the uncertainty as an explicit probability distribution and, as will be argued in Section 2.1.4, is the framework that we will focus on. Since we will be using Bayesian decision theory, we will discuss its differences with the frequentist's view. We further discuss how the Bayesian models are built and used for learning and inference in Section 2.2. In Section 2.3 we will provide some remarks on approximate Bayesian inference.
- Finally we draw the connection between the Bayesian decision theory and preference learning via a notion of a utility function in Section 2.4.

2.1 Foundation of Decision Theory

In this section we will explain the basics of decision theory. Further reading on decision theory can be found in [7–9, 77][45, Chapter 22][31, 32, Chapter 9].

2.1.1 Basic Definitions

In this thesis, we use a decision making scenario consisting of three basic elements, *state, action* and *utility* as detailed below:

- The state indicates the condition of the nature that the decision maker is in and is typically not fully realized and thus uncertain. This uncertainty can be due to the changing nature of the decision maker's environment or the noisy tools for measurement. As such, the "belief" over the value of the state is typically conditioned on some observations. This uncertain variable, denoted as $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ (depending on the application can be vector of continuous or discrete values), describes the attributes of this environment. When experiments are performed to gather information about the value of $\boldsymbol{\theta}$, it is called the *parameter* in some parameter space. We use the state of nature and the parameter interchangeably to refer to $\boldsymbol{\theta}$.

- The decision maker can affect its environment by taking an *action* $a \in \mathcal{A}$ in the set of all possible actions under consideration $\mathcal{A} = \{a_1, a_2, \dots, a_k\}$. At each state, the action performed produces an outcome $o : \mathcal{A} \times \boldsymbol{\Theta}$. The analysis of this outcome for the selected action is of central interest to decision theory.

- The utility function is a means of evaluating how valuable the outcome of an action is for the decision maker in a given state. This function $u : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}$ returns higher values for the actions that are more favorable in a given state. In the cases where there is no uncertainty (i.e. deterministic environment), the optimal action is the one that has the highest utility. However, in general the environment is uncertain and we need to consider expected utility over all possible states. In some applications it is insightful to think of the utility value in a monetary manner although this is not always the case.

In this thesis, we assume the outcome o is deterministic on the state-action space and therefore the utility of the selected action in the given state $u(\boldsymbol{\theta}, a)$ is equivalent to the utility of the outcome $u(o)$, i.e. $u(o) = u(\boldsymbol{\theta}, a)$.

Given a distribution over the state of the nature and a utility function, we define the *expected utility* as:

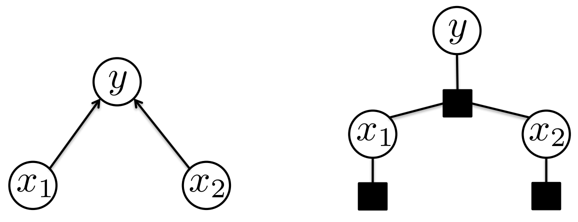
$$\mathrm{EU}_u(a) = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta})}\left[u(\boldsymbol{\theta}, a)\right], \tag{2.1}$$

with the *optimal action* being the one that maximizes this value, i.e.,

$$a^* = \arg\max_a \mathrm{EU}_u(a)$$

That is, with respect to our belief about the state of nature, the action that maximizes the expected utility is optimal. This definition connects three basic elements of decision theory. In the following we further detail how to obtain and model the distribution of states and the utility.

It is noteworthy to add that the notion of utility is closely related to the *loss*. We view loss as a merely different view on evaluation of an action in a given state that measures its cost, as opposed to the gain measured by utility. It can be simply seen as the negative of the utility. Due to this dualistic view, we use maximize negative loss or utility depending on the context interchangeably.



(a) Bayesian network (b) Factor graph

Figure 2.1: A graphical model of three random variables x_1, x_2, y

2.1.2 Modeling Uncertainty

A well-understood mathematical tool for representing and reasoning under uncertainty is probability theory that is foundational to Bayesian methods that underly work in this thesis. In this section we discuss how to represent the probabilities and learn from observations. In subsequent sections where we discuss the uncertainty in the utility function, we employ the same modeling tools for the distribution of the utilities.

2.1.2.1 Graphical Models

Graphical models are the marriage of graph theory and probabilistic modeling. The reason for using such models is more compact representation of the joint probabilities by exploiting the independence of random variables. In particular for modeling the unknown state of nature, we might notice interactions among various variables that lead to a structured graphical model of conditional dependence. In addition, these graphical representations allow exploiting structure for efficient inference and paves the way for use of well-developed graph algorithms for probabilistic reasoning in graphical models. In short, we use a graph representation because:

- the graph structure reveals properties of the model, namely dependence/independence of variables;
- inference and learning can exploit the structure of graphical models for efficiency;
- much more compact than the explicit representation of full joint;
- compact models with fewer parameters have higher bias but can be learned with lower variance than full joint; and,
- it is easier to visualize the structure of the probabilistic models.

In a graphical model, we have a set of nodes corresponding to the random variables in the model and the edges between them representing probabilistic dependence. We have two types of graphical models [12, 45, 51, 90]:

1. **Undirected Graphical Models:** Also known as *Markov Random Fields (MRF)*, represent the distribution that factorizes according to set of functions ψ that define the interaction (or *compatibility*) between the variables in a *clique.* Denoting by C a clique in the graph, the joint probability of d random variables $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d$, can be written as:

$$p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_d) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{\theta}_C),$$

   where $\boldsymbol{\theta}_C$ is the subset of variables belonging to the clique C and their corresponding compatibility function ψ_C.

2. **Directed Graphical Models:** Also known as *Bayesian network*s, represent the connection between variables in a parent-child relationship that is specified by the arrow direction in the graph. In this relation, the children depend on their parents. Therefore, the joint distribution factorizes based on the parents $\text{pa}(\boldsymbol{\theta})$ of a variable $\boldsymbol{\theta}$, that is,

$$p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_d) = \prod p(\boldsymbol{\theta}_i | \text{pa}(\boldsymbol{\theta}_i)).$$

   An example of a directed graph is shown in Figure 2.1(a). In this graph, x_1, x_2 are the parents of y. The direction of arrows in the graph represent the direction of conditional dependency between variables. The joint in this figure represents the following:

$$p(x_1, x_2, y) = p(y|x_1, x_2)p(x_1)p(x_2)$$

Both directed and undirected graphical models can be easily depicted in a *factor graph* as shown in Figure *2.1(b).* The factor graph represents the factorization of a function. In a factor graph, the black squares represent the potential function between the connected random variables. This interaction is the compatibility function in undirected graphs or the joint distribution of the connecting variables in the directed one.

In many problems where the number of random variables is large and there are patterns of conditional independence, a *plate* diagram as shown in Figure 2.2 is conventionally used to provide a simpler representation. A simple example is shown in Figure 2.2 where variable y_i depends on $\mathbf{x}_i$ and $\boldsymbol{\theta}$ for all $i = 1, 2, \dots, n$. The observed variables are normally shaded in graphical models. This graphical model is common in machine learning (regression and classification) where the target value (label) y depends on the observation $\mathbf{x}$ and the state of nature $\boldsymbol{\theta}$. We will discuss a regression example (where y is continuous) in the following sections.

2.1.3 Using Prior and the Bayesian View

Statisticians are generally divided into two camps: *Frequentists* and *Bayesians* [7, 8]:

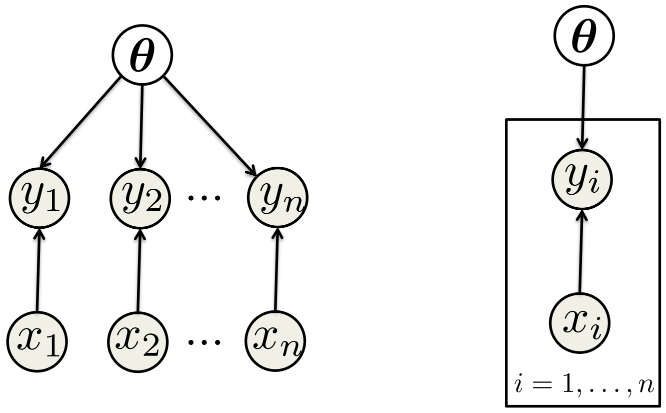


Figure 2.2: Graphical model and the plate representation

- The frequentist view is that the state of nature is an unknown that we have measurements from. One formulates a hypothesis model that minimizes the average loss of the observed measurements. Through a large number of observations, the model that minimizes the average loss on the training observations learns to recognize the unseen samples with a good performance. Similarly, models that find the single parameter that maximizes the probability of the observations are frequentist.

- The Bayesian view on the other hand, thinks of the state of nature as a random variable. Hence as a random variable, there is an initial belief about its value and upon seeing new observations this belief is updated. Bayesian modeling is done with the view that the uncertainty about the unknown state of nature is often reduced as more observations are made.

Bayesian methods use simple probability rules to infer unknown states. In particular, for two random variables A and B, we will use two standard probability rules:

$$\begin{aligned} p(A) &= \int p(A,B)dB && \text{marginalization (sum/integrate out),} \\ p(A,B) &= p(A|B)p(B) && \text{chain rule (product rule).} \end{aligned}$$

These probabilistic principles combined with the basic Bayes rule lay the foundation for Bayesian reasoning about the uncertainty of the unknown state,

$$p(\boldsymbol{\theta}|\mathcal{D}) \quad = \quad \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{Z}, \tag{2.2}$$

where $\mathcal{D}$ denotes the set of observed random variables $\{\mathbf{d}_i\}_{i=1,\ldots,n}$ and

$$Z \ = p(\mathcal{D}) \quad = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.3}$$

Since it is assumed that the observation $\mathbf{d}_i \in \mathcal{D}$ is independently and identically distributed, the likelihood term factorizes, i.e.

$$p(\mathcal{D}|\boldsymbol{\theta}) \quad = \quad \prod_{i=1}^{n} p(\mathbf{d}_i|\boldsymbol{\theta})$$

In Figure 2.2, the graphical model of such factorization where $\mathbf{d}_i = (\mathbf{x}_i, y_i)$ is shown.

Here, $p(\boldsymbol{\theta})$ represents the *prior*, $p(\mathcal{D}|\boldsymbol{\theta})$ the *likelihood* and Z is the normalizer that ensures the *posterior* on the left-hand-side (LHS) sums to one, and hence is a proper distribution. This prior is obtained from a problem-specific expert's knowledge or a similar problem. The distribution of the prior is often determined by its parameters that are called *hyper-parameters* to set apart from the unknown parameters we are looking for. *Inference* is the procedure of obtaining the posterior distribution of these unknown parameters from the prior using the likelihood. The posterior, $p(\boldsymbol{\theta}|\mathcal{D})$, is the distribution of the state variable given the observations. When we are in the *no-data* setting, that is, there are no observation, the posterior is equal to the prior. Upon having new observations we can incrementally update our belief about the state. In each step, the posterior of the previous iteration acts as a prior in the Bayes rule to update the belief about $\boldsymbol{\theta}$, i.e.

$$p(\boldsymbol{\theta}|\mathcal{D}) \quad \propto \quad p(\mathbf{d}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}_{1,\ldots,n-1})$$

where $\mathcal{D}_{1,\ldots,n-1}$is the set of first $n-1$ observations.

When the integral in the normalizer Z is tractable, *exact* inference can be carried out. Often times in practice however, the integral is hard to perform and as such various sampling and approximation methods are developed to estimate the posterior. We will discuss these methods in the subsequent section.

An important advantage of Bayesian modeling is that, as opposed to the frequentest view, *over-fitting* or *under-fitting* is automatically avoided. That is because in Bayesian methods, for prediction we integrate over all parameterizations, i.e.

$$p(\mathbf{d}^*|\mathcal{D}) \quad \propto \quad \int p(\mathbf{d}^*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$$

where $\mathbf{d}^*$ is the unseen sample. This integration acts as an averaging with respect to all possible parameter values as opposed to picking the parameter that maximizes the posterior probability which might overfit the data and lead to poor generalization to unseen examples.

Furthermore, Bayesian models are generally considered *consistent* in the sense that if the true parameter value $\boldsymbol{\theta}^*$ that generates the data is in the prior, as the number of observations approach infinity, the posterior converges to the delta function centered at the true parameter value [31, 32], i.e., for n observations in $\mathcal{D}$,

$$\lim_{n\to\infty} p(\boldsymbol{\theta}|\mathcal{D}) \quad \to \quad \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*).$$

Simply put, the posterior will peak at the true parameter value in the posterior (convergences to the true point in the parameter space).

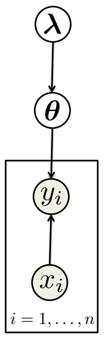


Figure 2.3: Hierarchical Bayesian model

A Bayesian model can be hierarchical by having multiple levels of priors where priors of priors are hyper-priors:

$$p(\boldsymbol{\theta}) = \int p(\boldsymbol{\theta}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda}.$$

where $\boldsymbol{\lambda}$ is the hyper-parameter. A graphical model of this new representation is shown in Figure 2.3. This will add further flexibility in the model. Although hierarchical models are more complex and their inference is more challenging, in problems like topic modeling they proved to be effective [13].

2.1.4 Bayesian Decision Theory

Now that the state distribution in a Bayesian view can be obtained from the observations, we can proceed to introduce *Bayesian decision theory* as:

$$\mathrm{EU}_u(a) = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})}[u(\boldsymbol{\theta}, a)] = \int u(\boldsymbol{\theta}, a)p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{2.4}$$

with the *optimal action* a^* being the one that maximizes this value, i.e.,

$$a^* = \arg\max_a \mathrm{EU}_u(a)$$

Then action a_1 is *preferred* to action a_2 if and only if

$$\mathrm{EU}_u(a_1) > \mathrm{EU}_u(a_2).$$

We say the decision maker is *indifferent* about a_1 and a_2 when $\mathrm{EU}_u(a_1) = \mathrm{EU}_u(a_2)$.

In problem settings where the utility itself is unknown, we can learn the distribution of the utility and compute the *expected expected utility*. In doing so, we learn the belief in both the utility and the state variable and then integrate them out in EEU, i.e.

$$\text{EEU}(a) \;=\; \mathbb{E}_{u\sim p(u)}\left[\text{EU}_u(a)\right] = \int \text{EU}_u(a)p(u)du.$$

Then the utility of an action is a distribution of the utility for that action and there is no independent state variable to consider.

2.2 Bayesian Modeling and Learning

So far we have introduced the general picture of decision theory without discussing the specific models. In this section, we concentrate on the Bayesian modeling for learning the distribution of unknown variables. These variables can be (1) the state of nature, (2) the utility function, or (3) other *latent* variables that the belief over state or utility function depends on.

In Bayesian modeling, we choose the prior and the likelihood and compute the posterior from the Bayes rule. When the product of the prior and the likelihood remains in the same family, computing the posterior is tractable and the prior is called *conjugate*. Formally, if $\mathcal{M}$ is a set of prior distributions on $\boldsymbol{\theta}$, then the prior is called conjugate to the likelihood model if for every observation, the posterior is in $\mathcal{M}$. We will give an example of a tractable problem and then discuss various approximate methods that in subsequent chapters will be used for intractable problems.

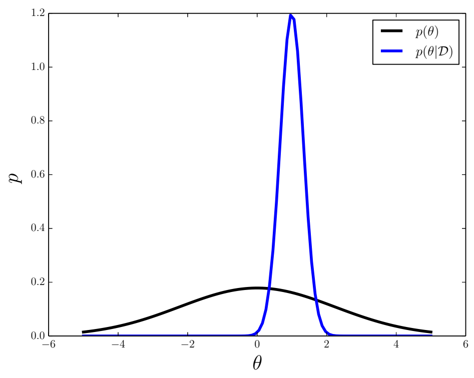
In the following we will discuss parametric and nonparametric Bayesian models. A particular characteristic of the parametric Bayesian modeling is that there is a fixed dimensional parameter whose posterior distribution we are interested to obtain. The posterior in these models takes a fixed size parametric form. In the nonparametric models on the other hand, the dimension of the parameter can grow with the data. In the following, we will discuss them in details.

2.2.1 Parametric Bayesian Modeling

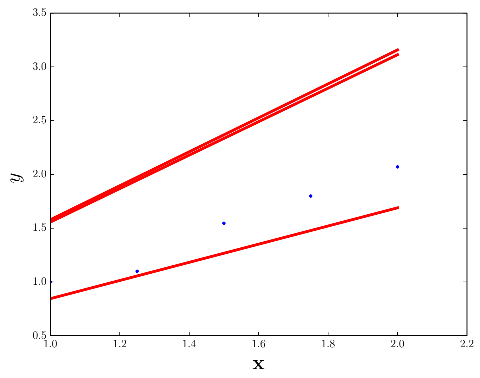
In the Bayesian view we are interested in the distribution of the unknown random variables. There is an initial belief about the value of these unknowns that ultimately become more certain with more observations. In case these likelihoods and the prior are conjugate we can compute the products in closed form and the integral is tractable. For example Gaussian distribution is a conjugate prior for a Gaussian likelihood as in the *Bayesian linear regression* below.

As an example of the parametric Bayesian modeling, consider the observations to be a set of pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ as a regression problem. The graphical model of this regression problem is shown in Figure 2.2 that corresponds to the following:

$$p(\boldsymbol{\theta}|\mathcal{D}) \;=\; \frac{1}{Z}p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{Z}p(\boldsymbol{\theta})\prod_{i=1}^{n} p(\mathbf{x}_i, y_i|\boldsymbol{\theta}).$$



(a) Prior and posterior on weights



(b) Regression model

Figure 2.4: Bayesian linear regression: the prior and the posterior of the parameter $\boldsymbol{\theta}$ are shown in Figure 2.4(a). In Figure 2.4(b), the dots are the observed points. The red lines are three samples from the linear function defined by the posterior distribution. As is shown, the posterior is peaking at the true value (1 in this example).

In case of regression where we are not modeling $p(\mathbf{x}_i)$, we can write the likelihood as $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$ (i.e. since $\mathbf{x}_i$ is observed, its distribution is absorbed in the normalizer). Assuming a Gaussian prior, $\boldsymbol{\theta} \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_d)$ and a Gaussian likelihood $y_i \sim \mathcal{N}(\boldsymbol{\theta}^\top \mathbf{x}_i, \beta^2 \mathbf{I}_d)$ with hyper-parameters $\alpha, \beta \in \mathbb{R}$, the posterior is

$$
\begin{aligned}
p(\boldsymbol{\theta}|\mathcal{D}) &= \mathcal{N}(\beta^{-2}\mathbf{S}\mathbf{X}^\top \mathbf{y}, \mathbf{S}). \\
& \qquad \mathbf{S} = \left(\alpha^{-2}\mathbf{I} + \beta^{-2}\mathbf{X}^\top \mathbf{X}\right)^{-1},
\end{aligned}
$$

where $\mathbf{I}_d$ is the identity matrix of size d, $\mathbf{X}$ is the matrix constructed from observation $\mathbf{x}$ and $\mathbf{y}$ is the vector of labels. The posterior is a distribution over the value of the state of nature which represents the linear functions that generated the label y. Here, if the labels are binary (± 1), the likelihood model is not Gaussian (i.e. not conjugate) and hence the integral in the posterior becomes intractable and requires approximation.

An illustration of this model at work is shown in Figure 2.4. The dots in Figure 2.4(b) represent the observations that are generated from a linear function with a small additional noise. That is, we chose a constant weight that was multiplied by each point on the x-axis and then a random noise was added to the output as shown in the y-axis, i.e. $y \sim \mathcal{N}(\mathbf{x}^\top \boldsymbol{\theta}, \beta^2)$. The prior, a zero mean Gaussian distribution over that unknown weight, and the posterior that was obtained from these observations is shown in Figure 2.4(a) where the posterior is peaked at the true parameter value that generated the data. This posterior represents the belief over the space of weights that could generate the observations (with the given likelihood parameter β that specifies the noise in the observations). Three samples of these linear functions induced by this posterior are also plotted in red. If we increase the number of observations, this

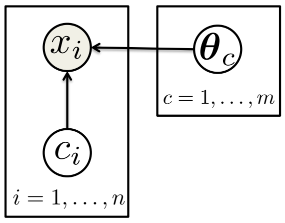


Figure 2.5: Graphical model of the Mixture Model

posterior Gaussian distribution on the weight vectors becomes sharper and looks more like a delta function. It is an important example that shows how the conjugate likelihood can lead to an efficient inference. We will get back to this example again when we discuss Gaussian processes and Bayesian inference.

Bayesian parametric models can be used in the estimation of the mixture models as well. The easiest way to think about the mixture model is in clustering where a subset of observations form a group (k-means can be seen as a non-Bayesian mixture model). Instead of a labeled set, we have an unlabeled set $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a mixture model for the likelihood. That is, there is a parameter vector for each cluster and there is a hidden variable $\mathbf{c}_i$ that specifies which cluster each observation $\mathbf{x}_i$ belongs to. The likelihood of each observation given its cluster is $p(\mathbf{x}_i|\boldsymbol{\theta}_c)$ (distribution of each observation depends on the parameter of that cluster). Then we have,

$$p(\boldsymbol{\theta}|\mathcal{D}) \quad = \quad \frac{1}{Z}\sum_{c=1}^{m} p(\boldsymbol{\theta}_c)\prod_{i=1}^{n} p(\mathbf{c}_i = c)p(\mathbf{x}_i|\boldsymbol{\theta}_c). \tag{2.5}$$

The integration of the posterior distribution becomes intractable in general and needs sampling or approximate inference. If we assume $p(\mathbf{x}_i|\boldsymbol{\theta}_c)$ and $p(\boldsymbol{\theta}_c)$ are conjugates (say a Gaussian distribution like the regression problem), we can use *Expectation Maximization (EM)* to estimate the membership variable $\mathbf{c}_i$ and the parameter $\boldsymbol{\theta}_c$ (if not conjugate we need an approximation step inside EM, e.g. variational EM). Although with this procedure we enter the realm of frequentists by *Maximum a-posteriori (MAP)* estimation (and *Maximum Likelihood* if we ignore the prior on the value of $\boldsymbol{\theta}$) by selecting one set of parameters among many, we can get a feeling of the possible value for the unknowns. A graphical model of the mixture model is shown in Figure 2.5.

2.2.2 Nonparametric Bayesian Modeling

Nonparametric Bayesian analysis has gained contemporary attention due to its flexibility in modeling complex phenomena and scaling with data [29, 67]. Examples of these methods are the Dirichlet process [85], hierarchical Dirichlet process [86], Gaussian process [74], Indian buffet process [35] and etc among many others [39]. Nonparametric methods are flexible and adapt to the observations. They consist of the (1) infinite dimensional parameter space and (2) can use the finite number of observations to explain the sample space. These methods are distributions with infinitely many parameters that can equivalently be thought of as distributions in function spaces. A simple nonparametric method is the Parzen window which estimates the density function as a mixture of Gaussians each centered at one of the observations. In this thesis, we will discuss two prominent nonparametric approaches: Gaussian and Dirichlet processes.

One important underlying assumption about nonparametric Bayesian methods is *exchangeability*. The joint distribution of exchangeable variables is invariant to their permutation. It is a valid and weaker assumption than i.i.d in machine learning. If this joint distribution depends on a variable that is distributed according to a prior and then integrated out, their joint marginal distribution remains exchangeable. The important consequence, due to de Finetti's theorem, proves the converse statement: if a joint distribution $\mathbf{x}_{1,} \ldots, \mathbf{x}_n$ is (infinitely) exchangeable then there is a latent random variable $\boldsymbol{\theta}$ such that

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \int p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

In other words, exchangeability automatically implies existence of a Bayesian model with $\boldsymbol{\theta}$ as its latent random variable. Therefore, any infinitely exchangeable sequence is inherently Bayesian with a process working under the hood. This is particularly important to the nonparametric Bayesian models because these models define a joint distribution of infinite variables. As will be shown below in both Gaussian and Dirichlet processes, there are underlying variables which are integrated out to model the joint distribution.

2.2.2.1 Gaussian Processes

A Gaussian process [51, 74] defines a multivariate Gaussian distribution (similar to the Bayesian linear regression) on functions over an input space. This distribution in the functional space can then be used as a prior. Since Gaussian processes are defined over functions that can represent infinite dimensional spaces, they extend the finite dimensional space in the parametric Bayesian models (e.g. regression model discussed before). We follow [51] in our introduction below.

Similar to that of the Bayesian linear regression, we use a linear function of the inputs. However, instead of using the inputs $\mathbf{x}$ directly, we use a linear combination of some basis functions $\phi_h(\mathbf{x})$ in the feature space, i.e

$$f(\mathbf{x}) = \sum_h \boldsymbol{\theta}_h \phi_h(\mathbf{x}).$$

These basis functions map the inputs to a higher dimensional space. Following the same line of argument as before in Bayesian linear regression, we can see that the posterior depends on the inner products of the basis functions for observed values. Denoting by $\boldsymbol{\Phi}$ the matrix with entries $\boldsymbol{\Phi}_{h,i} = \phi_h(\mathbf{x}_i)$ and assuming $\boldsymbol{\theta}$ is a zero mean

Gaussian with variance α^2, we have:

$$\begin{aligned}\mathbb{E}[f] &= \mathbf{\Phi}\mathbb{E}[\boldsymbol{\theta}] = 0\\ \mathrm{cov}[f] &= \mathbf{\Phi}\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^\top]\mathbf{\Phi}^\top = \alpha^2\mathbf{\Phi}\mathbf{\Phi}^\top.\end{aligned}$$

This is a defining characteristics of a Gaussian process, namely the distribution of f is a Gaussian that is defined by the product of the input values. The covariance matrix is defined as:

$$[\mathbf{\Phi}\mathbf{\Phi}^\top]_{i,j} = \sum_h \phi_h(\mathbf{x}_i)\phi_h(\mathbf{x}_j).$$

Increasing the number of basis functions to possibly infinite, this summation becomes integral (there are infinitely many basis function that can be used). This integral of inner products of these mapped functions represents the *kernel function* in the (possibly) infinite dimensional *Hilbert* space. From applying the kernel functions on the input values, we obtain the kernel matrix that replace the covariance matrix, i.e.

$$\mathbb{E}[f(\mathbf{x}_i)f(\mathbf{x}_j)] = k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{K}_{i,j}.$$

Hence, Gaussian process prior, as a joint distribution of the functions $\mathbf{f}$ in the space induced by the mapping function ϕ over the matrix $\mathbf{X}$ built from n inputs $\mathbf{x}$ is:

$$p(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{f}; 0, \mathbf{K})$$

This is the distribution of the functional values in the input space. It should be noted that the state variable $\boldsymbol{\theta}$ is integrated out and we are left with a distribution over functions. Using this notion, we can form a belief over the non-linear functions that can be used for regression, classification and utility modeling. Gaussian process provides a very powerful statistical tool for Bayesian learning in complex domains. For instance in regression, we have the likelihood for the target values $\mathbf{y}$,

$$p(\mathbf{y}|\mathbf{f}, \mathbf{X}) = \mathcal{N}\left(\mathbf{y}; \mathbf{f}, \sigma^2\mathbf{I_n}\right),$$

that is, the likelihood is defined as the value of this *latent* function obtained from the Gaussian process prior with additional noise with variance σ^2. Then the distribution of target values given the observed inputs is obtained by integrating out the latent functions as

$$\begin{aligned}p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|\mathbf{f}, \mathbf{X})p(\mathbf{f}|\mathbf{X})d\mathbf{f}\\ &= \mathcal{N}\left(\mathbf{y}; \mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}_n\right).\end{aligned}$$

For prediction, we have the joint probability of the function values for the observed $\mathbf{y}$ and predictions $\mathbf{f}^*$ for test example $\mathbf{x}^*$:

$$p\left(\left[\begin{array}{c}\mathbf{y}\\ \mathbf{f}^*\end{array}\right]\middle|\mathbf{X},\mathbf{x}^*\right) = \mathcal{N}\left(\left[\begin{array}{c}\mathbf{y}\\ \mathbf{f}^*\end{array}\right];\mathbf{0}, \begin{array}{cc}\mathbf{K}+\sigma^2\mathbf{I_n} & \mathbf{k}_*\\ \mathbf{k}_*{}^\top & \mathbf{k}_{*,*}\end{array}\right),$$

where

$$\begin{aligned}[\mathbf{k}_*]_i &= k(\mathbf{x}^*,\mathbf{x}_i)\\ \mathbf{k}_{*,*} &= k(\mathbf{x}^*,\mathbf{x}^*).\end{aligned}$$

and $[\mathbf{k}_*]_i$ denotes the ith element in the vector. Then the predictive distribution is the conditional obtained from this joint:

$$p(\mathbf{f}^*|\mathbf{y},\mathbf{X},\mathbf{x}^*) = \mathcal{N}\left(\mathbf{f}^*;\mathbf{k}_*(\mathbf{K}+\mathbf{I}_n)^{-1}\mathbf{y},\mathbf{k}_{*,*}-\mathbf{k}_*{}^\top(\mathbf{K}+\mathbf{I}_n)\mathbf{k}_*\right).$$

In the following chapters we will discuss how these functions can be seen as the unknown utilities and be used for preference learning. Unlike the regression model here, because of the form of the likelihood in both classification and preference learning, the posterior becomes intractable and approximate inference has to be carried out, we discuss later.

The difficulty with the Gaussian process is the selection of the kernel function that specifies the prior. The kernel function acts as the basis for the function we are learning. Expert knowledge can be used in kernel selection for the given task. Additionally, most kernel functions have hyper-parameters that have to be tuned to fit the given problem. For instance in Gaussian (or RBF) kernel,

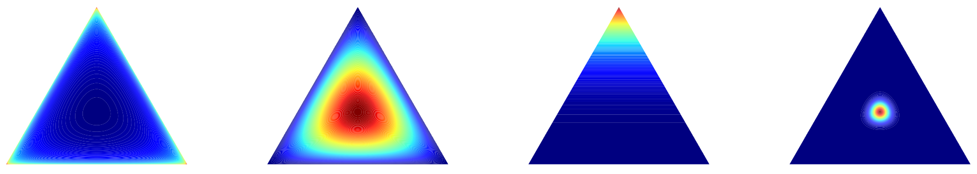
$$\mathbf{K}_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\lambda}\right),$$

the hyper parameter is λ that specifies how sensitive the kernel function is to the distance between inputs. Similarly in squared exponential kernel,

$$\mathbf{K}_{i,j} = \lambda_1^2\exp\left(-(\mathbf{x}_i-\mathbf{x}_j)^\top A(\mathbf{x}_i-\mathbf{x}_j)\right)+\lambda_2\delta_{i,j},$$

hyper-parameter λ_2 puts more emphasis on the same instances corresponding to the diagonal entries in the kernel matrix. The other hyper-parameter A is a squared symmetric matrix that specifies the correlation between input features. This kernel is an instance of *automatic relevance determination* (ARD) [51] in which the matrix A determines the correlation of features of $\mathbf{x}_i$ and $\mathbf{x}_j$ similar to that of a covariance matrix in a Gaussian distribution. Alternatively one can view matrix A as a weighting of the features of the input $\mathbf{x}$.

One advantage of Gaussian processes is that the automatic optimization of the



(a) $\alpha = \{0.99, 0.99, 0.99\}$ (b) $\alpha = \{2, 2, 2\}$ (c) $\alpha = \{1, 1, 5\}$ (d) $\alpha = \{50, 50, 50\}$

Figure 2.6: Dirichlet distribution for various values of α

hyper-parameters can be done by maximizing the log marginal likelihood:

$$\log(p(\mathbf{y}|\mathbf{X})) = \frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y} - \frac{1}{2}\log\det(\mathbf{K} + \sigma^2\mathbf{I}_n) - \frac{n}{2}\log(2\pi).$$

Here, only the first and the second term depend on the kernel matrix and can be easily optimized. For further reading, refer to the Gaussian processes book [74].

2.2.2.2 Dirichlet Processes

The Dirichlet distribution Dir($\boldsymbol{\alpha}$) is defined over a finite simplex parameterized by a variable $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_m\}$. This parameter specifies the concentration of samples in the simplex. Each point in this simplex can be thought of as a probability mass function (pmf) and consequently the joint is weighted sum of these pmfs (by α). In Figure 2.6, we show examples of Dirichlet distribution with various values of parameter α. One important property of the Dirichlet distribution that is used in building the Dirichlet process is *aggregation*. That is, if partitions of the sample space are joined, the resulting sample space is again a Dirichlet distribution where the values of αs for that partition are aggregated.

Dirichlet process is an extension of the mixture model introduced earlier (refer to [39] for a complete introduction). It models the distribution over an infinite sample set as an infinite mixture of weighted Dirac delta functions. This infinite mixture becomes the Dirichlet distribution again when we employ the aggregation property on all the sample space regions. The parameter $\alpha(A)$ is the function of the region A of this Dirichlet distribution and is the integral of all the point masses in the region, i.e. $\alpha(A) = \int \mathbb{I}_A[x] d\alpha(x)$ where $\mathbb{I}_A$is the indicator function of region A for x in the input space. For Dirichlet process, there is an additional parameter G that specifies the base distribution where the samples are generated from. Base distribution is the expected value of the Dirichlet process. It should be noted that while G can be either continuous or discrete, the Dirichlet process is always discrete. Similar to the Gaussian process where the marginals were Gaussian, in Dirichlet process the marginals are Dirichlet.

A simple intuitive view is to think of the sample space as shown in Figure 2.7. The joint distribution is defined with a Dirichlet (for two partitions it is a Beta distribution). The Dirichlet process is defined as the infinite mixture of the data points. With

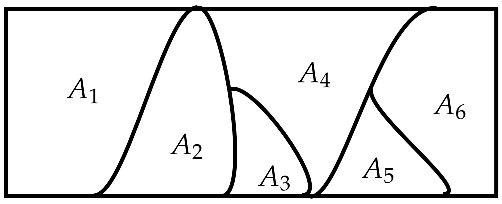


Figure 2.7: An illustration of the Dirichlet distribution and the partitioning of the space.

some probability one partition might break into two. As such, one way to construct the Dirichlet process is to consider the partitioning of the input space $A = \{A_1, A_2\}$. Then the distribution in A is a Beta$(\alpha(A_1), \alpha(A_2))$ with its expectation defined as

$$\frac{\alpha(A_1)}{\alpha(A_1) + \alpha(A_2)} = \frac{\alpha(A_1)}{M}.$$

Here, the denominator M denotes the mass of the region on which the probability is defined. When increasing the number of partitions, the joint distribution becomes Dirichlet with similar property. In particular for finite partitions $A = \{A_1, A_2, \ldots, A_m\}$ the posterior $p(A_1), \ldots, p(A_m)$ on given inputs $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is

$$\mathrm{Dir}(\alpha(A_1) + n_1, \ldots, \alpha(A_m) + n_m)$$

where

$$n_i = \sum_{j=1}^{n} \mathbb{I}[\mathbf{x}_j \in A_i],$$

the count for observations in partition A_i. The expectation of the posterior is [39]:

$$\mathbb{E}[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] = \frac{M}{M+n} G(A) + \frac{n}{M+n} \tilde{p}$$

where $\tilde{p}$ is the empirical distribution, i.e., the distribution of observed points. This equation simply states on expectation a sample is drawn from the base distribution G with a value proportionate to the mass (that is in the heart of sampling for the Dirichlet processes). Similarly, with a value proportionate to the number of observations it is assigned to one of the current partitions. Since this distribution is exchangeable, we have the following conditional:

$$p(\mathbf{x}_{n+1} \in A_m | \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \frac{M}{M+n} G(A_m) + \frac{n_m}{M+n} \tilde{p}.$$

Again, probability of assigning a point $\mathbf{x}_{n+1}$ to a partition is proportionate to the number of instances in that partition. Otherwise, with some probability proportionate to α_0 it is as the distribution of $G(A_m)$. As such, it is more probable for a new instance to be assigned to a more populous partition. Schemes such as Polya urn method, Chinese restaurant process and stick-breaking are developed around this central concept to provide various views for construction of the Dirichlet process [29, 39, 56, 67].

Another important property of the Dirichlet distribution is that it is a conjugate prior for the multinomial distribution. In other words, the simplex defined by the Dirichlet distribution is a distribution over parameter of the multinomial distribution. By integrating over all possible such parameters, we obtain a multinomial posterior. In the same way, one can use the Dirichlet distribution as a prior for the cluster probability $p(\mathbf{c}_i = c)$ in Equation 2.5. We will see an example of using this property in Chapter 3.

2.3 Bayesian Inference

As mentioned before, Bayesian inference is the process of obtaining the posterior from the prior by incorporating the likelihood of the observations. In many practical cases the integral of the posterior is intractable and approximations are needed. Although for simple problems with small dimensions numerical integration (such as quadrature method) can be performed, in practice these methods are infeasible in high dimensions. Also, since the posterior is obtained from the product of all the likelihoods with the prior, it can be highly multimodal. In this section we will discuss key Bayesian inference methods in the literature which will be used throughout the subsequent chapters.

In general, approaches to inference can be either deterministic or stochastic. In deterministic methods, the true posterior is used to constrain the approximate distribution so that is easier to work with. For example in variational methods, the true posterior is used as an upper bound on a predefined family of distributions (such as exponential family). These methods generally enjoy fast inference, however, the effectiveness of the approximation highly depends on the divergence measure and the distribution family used [54]. The stochastic methods typically draw samples directly from the posterior e.g. Markov Chain Monte Carlo methods. These methods are generally slower and computationally more demanding, but can be arbitrarily accurate in estimating the moments of the posterior.

2.3.1 Laplace Method

The first approximate Bayesian inference method that we will consider is the Laplace approximation [12, 55]. Laplace's method uses the Taylor expansion of the log of the posterior at point $\boldsymbol{\theta}^*$ as

$$\log(p(\boldsymbol{\theta}|\mathcal{D})) \approx \log(p(\boldsymbol{\theta}^*|\mathcal{D})) + \mathbf{g}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*),$$

where

$$\begin{aligned}\mathbf{g} &= \left(\frac{\partial \log(p(\boldsymbol{\theta}|\mathcal{D}))}{\partial \boldsymbol{\theta}}\right)_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}\\ \mathbf{H} &= \left(\frac{\partial^2 \log(p(\boldsymbol{\theta}|\mathcal{D}))}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^\top}\right)_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}.\end{aligned}$$

This expansion already looks like the log of Gaussian distribution. If we use $\boldsymbol{\theta}^*$at the mode (obtained from the MAP estimate), then $\mathbf{g} = 0$ and we get

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{\theta}^*, -\mathbf{H}^{-1}).$$

Simply put, we obtain the point estimate of the posterior using MAP inference and use this point to compute the covariance matrix. Then, Laplace's method approximates the posterior with a unimodal Gaussian distribution centered at a mode of that posterior. As such, it ignores the complex multimodal shape of the posterior. Although the curvature of the posterior is captured in the hessian matrix of the log of the posterior, in general the exact shape of the posterior is not preserved in its Laplace's approximation.

2.3.2 Variational Inference

Variational methods [12, 45, 88, 90, 92] have received significant attention in recent years and use the calculus of variations to have a better global approximation. Unlike Laplace's method that is rather simplistic, variational methods can work with a larger family of distributions. Similar to Laplace's method, variational methods involve finding a lower bound to the true posterior. Maximizing this lower bound can be done efficiently using well-established optimization algorithms. Therefore variational methods are efficient inference algorithms.

Since the difficult part of the inference is computing the normalizer, we use calculus of variation to rewrite it as

$$Z = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \int \left(\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\psi})}\right) q(\boldsymbol{\theta}|\boldsymbol{\psi})d\boldsymbol{\theta}$$

where $\boldsymbol{\psi}$ is the collection of parameters that define q. The alternative distribution q is used to approximate the true posterior. Using $-\log$ as a convex transformation of Z and Jensen's inequality, we have

$$\log(Z) \geq \int \log\left(\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\psi})}\right) q(\boldsymbol{\theta}|\boldsymbol{\psi})d\boldsymbol{\theta},$$

therefore,

$$Z \geq \exp\left(\int \log\left(\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\psi})}\right) q(\boldsymbol{\theta}|\boldsymbol{\psi})d\boldsymbol{\theta}\right).$$

Maximizing the right hand side of this equation will achieve a tight lower bound on the normalizer. The tightness of this bound depends on the choice of q. It is easy to see this bound is equal to the negative of the KL-divergence between the two distributions. Therefore, this maximization over parameters $\boldsymbol{\psi}$ is equivalent to minimizing the KL-divergence between the approximate and the true posterior, i.e.

$$\mathrm{KL}(q||p) \quad = \quad \int \log\left(\frac{q(\boldsymbol{\theta}|\boldsymbol{\psi})}{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}\right) q(\boldsymbol{\theta}|\boldsymbol{\psi}) d\boldsymbol{\theta}.$$

The idea is that computing this integral with respect to q and working with the log is simpler because q is potentially a distribution with respect to which this integral becomes tractable. This minimization objective is also known as I-projection (information projection) since it seeks to estimate the regions of the posterior that have higher density.

Various assumptions and constraints on the approximate distribution leads to several extensions namely, mean-field, Bethe method and cavity method [31, 90, 92]. For example, mean-field methods assume the parameters of the true posterior are independent which leads to simplification of the minimization of the KL divergence.

2.3.3 Expectation Propagation (EP)

Another approximate Bayesian inference method is *Expectation Propagation* (EP) [55]. EP is an efficient algorithm for approximating factorized distributions. The general idea is that we can approximate the posterior by replacing each factor with an approximation and computing the product that permits a tractable posterior computation. Iterating through this procedure for all factors will approximate the posterior. EP is presented in Algorithm 2.1 in detail. As is seen, first the approximate posterior $q(\boldsymbol{\theta})$ is constructed from the prior and the initial estimate of each factor. Then iteratively, each factor is replaced with its true likelihood ($p(\mathcal{D}_i|\boldsymbol{\theta})$) and then updated accordingly for all n factors. A graphical model of EP is shown in Figure 2.8.

Algorithm 2.1 Expectation Propagation (EP)

```
$q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^{n} q_i(\boldsymbol{\theta})$
while not converged do
    for i=1,...,n do
        compute cavity distribution $q_{\setminus i}(\boldsymbol{\theta}) = q(\boldsymbol{\theta})/q_i(\boldsymbol{\theta})$
        compute tilted distribution $\tilde{q}_i(\boldsymbol{\theta}) = p(\mathcal{D}_i|\boldsymbol{\theta})q_i(\boldsymbol{\theta})$
        set $q_i^{\text{new}}(\boldsymbol{\theta})$ so that $q_i^{\text{new}}(\boldsymbol{\theta})q_{\setminus i}(\boldsymbol{\theta}) \approx \tilde{q}_i(\boldsymbol{\theta})$
        update $q(\boldsymbol{\theta}) = q_i^{\text{new}} q_{\setminus i}(\boldsymbol{\theta})$
    end for
end while
```

It is insightful to think of EP as a message-passing algorithm: when approximating a factor q_i, the incoming messages from other factors comprise the cavity distribution $q_{-i}(\boldsymbol{\theta}) = q(\boldsymbol{\theta})/q_i(\boldsymbol{\theta})$. Then, the factor is approximated using the tilted

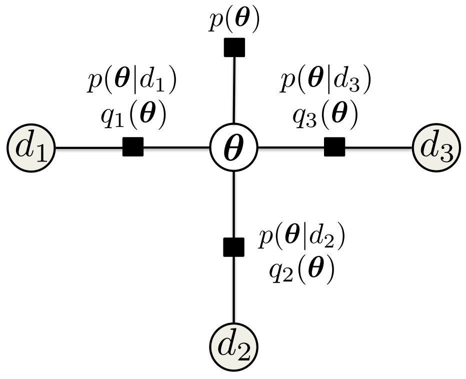

$p(\boldsymbol{\theta})$
$p(\boldsymbol{\theta}|d_1)$
$q_1(\boldsymbol{\theta})$
$p(\boldsymbol{\theta}|d_3)$
$q_3(\boldsymbol{\theta})$
d_1
$\boldsymbol{\theta}$
d_3
$p(\boldsymbol{\theta}|d_2)$
$q_2(\boldsymbol{\theta})$
d_2

Figure 2.8: A graphical model for EP

distribution $q_{\setminus i}(\boldsymbol{\theta})$ by incorporating the given likelihood and the cavity distribution $p(\mathcal{D}_i|\boldsymbol{\theta})q_{-i}(\boldsymbol{\theta})$. This draws further attention to the fact that using simple approximation of each likelihood term (as probability of the conditionally independent variable $\mathcal{D}_i$) and computing the approximate posterior from it is not a good approximation of the true posterior. Instead EP suggest using an approximation of each term that incorporates the approximations to other terms as well (i.e. effect of each observation individually on the posterior distribution). That is why the cavity distribution is required to account for the other variables and their influence. If the approximation at each step is not required (computing tilted distribution is tractable), with this message passing view one can obtain the popular loopy Belief propagation algorithm [95] as a special case of EP.

The projection in this procedure is equal to minimizing the KL-divergence between the true posterior and its approximation, i.e.

$$\mathrm{KL}(p||q) \quad = \quad \int \log\left(\frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})}\right) p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}.$$

In particular, in the algorithm when the approximation of each factor is updated in $q_i^{\text{new}}(\boldsymbol{\theta})q_{-i}(\boldsymbol{\theta}) \approx q_{\setminus i}(\boldsymbol{\theta})$, we minimize the KL divergence between this approximation and the true posterior. Interestingly when the distribution of q is in exponential family, it is easy to see that this minimization is equal to matching the expectations under p and q. This is known as *moment matching* (matching mean, variance, etc.). Hence unlike variational inference, in EP the mass of the posterior is more important to be correctly estimated. Furthermore, as opposed to variational methods EP is zero-avoiding, meaning that if there is a non-zero density region in the posterior it will be considered in the approximations [45].

EP is an excellent algorithm for complex methods like Gaussian processes and provides efficient inference for classification. In Chapter 4 we explain in detail how EP can be used for efficient inference in Gaussian processes for preference learning.

2.3.4 Sampling and Markov Chain Monte Carlo (MCMC)

In this section, we discuss stochastic methods for estimating the posterior. The general idea in stochastic methods is to draw samples from the posterior to estimate any desired expectation. For simple distributions, if the cumulative density function (CDF) is known, the samples can be easily drawn from its inverse function. However, in general many distributions that we are interested in (i.e. the posterior) may not have a tractable CDF. For further reading on stochastic methods refer to [32, 57, 79]. We closely follow [57].

2.3.4.1 Monte Carlo Methods

Monte Carlo (MC) methods are stochastic methods for unbiased estimation of the integrals with respect to a distribution by sampling. In particular, let's imagine we are estimating the expectation of the posterior parameter, i.e.

$$\mathbb{E}_p[\boldsymbol{\theta}] \quad = \quad \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}.$$

This integral is then estimated by an unbiased value obtained from the average of the samples, i.e.

$$\mathbb{E}_p[\boldsymbol{\theta}] \quad \approx \quad \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{\theta}_i \qquad \boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\mathcal{D}).$$

As the number of samples grow, this average approaches the true expectation. Unlike the previous approaches, MC methods can be arbitrarily accurate depending on the number of samples.

In simple problems with low dimensions, one can perform rejection sampling, that is, generate samples of $\boldsymbol{\theta}$ from a region that is easy to sample from and contains the posterior (e.g. a hypercube), and with some probability proportionate to the posterior's density accept each sample. In high dimensional problems though, it is computationally costly to perform such sampling and almost all generated samples are rejected.

The difficulty with this approach is that samples generally have to be generated from the prior and weighted by the likelihood (likelihood weighting). Since the priors are typically far from the desired posterior, this process becomes slow and inefficient. The alternative approach is to construct the sequence of samples from the posterior distribution. *Markov Chain Monte Carlo* (MCMC) methods search the space and generate the samples that are provably from the posterior when the number of samples grows. It is known as Markovian since each sample is drawn conditioned on the previous one which forms a chain. In this search, MCMC generates more samples from the regions with higher density.

Formally, a Markov chain is a series of random variables $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N$ in which the influence of values of each sample on the distribution of $\boldsymbol{\theta}_{N+1}$ only depends on

$\boldsymbol{\theta}_N$, i.e.

$$p(\boldsymbol{\theta}_{N+1}|\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_N) \quad = \quad p(\boldsymbol{\theta}_{N+1}|\boldsymbol{\theta}_N).$$

It is important for the MCMC to reach a steady-state or *converge*, that is, the average converges to the true expectation. This usually means taking new samples won't lead to sudden changes in the average. The *stationary* (or invariant) distribution is the one that Markov chain stays in during sampling. That is, distribution π is said to be the stationary distribution with transition probability $T(\boldsymbol{\theta},\boldsymbol{\theta}')$ such that

$$\pi(\boldsymbol{\theta}) \quad = \quad \int \pi(\boldsymbol{\theta}')T(\boldsymbol{\theta}',\boldsymbol{\theta})d\boldsymbol{\theta}'.$$

Often we use the time reversible Markov chains that satisfy the *detailed balance* condition, that is, the probability of transition from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$ is the same as the probability of being in state $\boldsymbol{\theta}'$ and transitioning to $\boldsymbol{\theta}$:

$$\pi(\boldsymbol{\theta})T(\boldsymbol{\theta},\boldsymbol{\theta}^{'}) \quad = \quad \pi(\boldsymbol{\theta}^{'})T(\boldsymbol{\theta}^{'},\boldsymbol{\theta})$$

This property insures the generated samples are unbiased and have the desired stationary distribution. It is also used in devising Markov Chain sampling algorithms such as Metropolis-Hastings that will be discussed. In the rest of this section, the stationary distribution is the posterior of the state $p(\boldsymbol{\theta}|\mathcal{D})$ that we are interested in sampling from.

2.3.4.2 Gibbs Sampling

Gibbs sampling is a widely used MCMC method. Suppose the posterior we are interested in is a joint distribution (e.g. univariate variables that jointly specify $\boldsymbol{\theta}$ as a vector):

$$p(\boldsymbol{\theta}|\mathcal{D}) \quad = \quad p(\{\boldsymbol{\theta}_i\}_{i=1,\ldots,d}|\mathcal{D}).$$

In this case, Gibbs samples each variable $\boldsymbol{\theta}_i$ conditioned on the values of all the other variables $\boldsymbol{\theta}^{\setminus i}$ at a given time step, i.e.

$$p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{\setminus i},\mathcal{D}).$$

Unless we are using *Blocked Gibbs* where variables are sampled jointly, θ^i is a one-dimensional variable. This conditional can be written as

$$p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{\setminus i},\mathcal{D}) \quad = \quad \frac{p(\boldsymbol{\theta},\mathcal{D})}{\int p(\boldsymbol{\theta},\mathcal{D})d\boldsymbol{\theta}_i}$$

which is easy to estimate, in particular, the univariate integral in the denominator is either analytically tractable or numerically computable. For example in the mixture model we discussed earlier, we can sample the probability of each cluster as,

$$p(\mathbf{c}_i = c|\boldsymbol{\theta}, \mathcal{D}, \mathbf{c}_{\setminus i}) \quad = \quad \frac{p(\mathbf{c}_i = c)p(\mathcal{D}|\boldsymbol{\theta}_c)}{\sum_{c'=1}^{m} p(\mathbf{c}_i = c')p(\mathcal{D}|\boldsymbol{\theta}_{c'})},$$

where $p(\mathcal{D}|\boldsymbol{\theta}_c)$ is the likelihood of all the instances in cluster c and $p(\mathbf{c}_i = c)$ is the prior that can be generated from a Dirichlet process.

In an extension, *collapsed Gibbs* marginalizes over one or more variables that are easy to integrate out and perform sampling on the rest.

2.3.4.3 Metropolis-Hastings

Another popular MCMC method is Metropolis-Hastings (MH). It models the selection of every sample by considering the probability of jumping from a given sample to a proposal based on its distribution and some transition probability. For drawing the samples from $p(\boldsymbol{\theta}|\mathcal{D})$, we can use reversibility condition as,

$$p(\boldsymbol{\theta}^t|\mathcal{D})T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) = p(\boldsymbol{\theta}^{t+1}|\mathcal{D})T(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t) \tag{2.6}$$

where $T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ is the probability of moving from $\boldsymbol{\theta}^t$ to $\boldsymbol{\theta}^{t+1}$ at time t and $t+1$. Since $T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ as the transition probability that preserved the detailed balance is unknown, we use an alternative distribution that we can control to preserve the detailed balance. That is, we replace unknown $T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ with $T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})\gamma(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ where $T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ is known as the *proposal distribution* and $\gamma(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})$ is the additional probability of move to make sure the reversibility condition is satisfied:

$$p(\boldsymbol{\theta}^t|\mathcal{D})T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})\gamma(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) = p(\boldsymbol{\theta}^{t+1}|\mathcal{D})T'(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)\gamma(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t) \tag{2.7}$$

Then, we can accept each move with probability equal to γ, i.e,

$$\gamma(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) = \min\left[\frac{p(\boldsymbol{\theta}^{t+1}|\mathcal{D})T'(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)}{p(\boldsymbol{\theta}^t|\mathcal{D})T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})}, 1\right] \tag{2.8}$$

The proposal distribution specifies the "jumps" in this Markov chain. We can use any distribution with symmetry condition $T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) = T'(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)$ as proposal however commonly a Gaussian centered at the previous sample with a constant variance is used, i.e.

$$T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) \quad = \quad \mathcal{N}(\boldsymbol{\theta}^t, \alpha^2).$$

Here, α^2 denotes the variance of this transition and defines how far from the current sample we should traverse to obtain a new sample. Selection of this variance is particularly important and impacts the performance of the sampler. If it is large, then sampler is mostly exploring and a large fraction of these transitions will be rejected. If it is too small, all the samples are taken from a small region and the sampler won't be able to traverse the space adequately. In both cases, the Markov chain won't *converge* efficiently.

In Algorithm 2.2 Metropolis-Hastings is summarized. It is easy to see that the Gibbs sampler is a special case of the Metropolis-Hastings when the conditional distribution is the proposal distribution. Since the proposal distribution is the same as the posterior we are interested in, then all the proposed samples are accepted in Gibbs sampler.

Algorithm 2.2 Algorithm for generating samples using Metropolis-Hastings

```
Input: $\mathcal{A}, n$ // Set of actions and the number of samples
for $t = 1$ to $n$ do
    $\boldsymbol{\theta}' \sim T'(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)$ // Draw next sample
    $u \sim U[0,1]$ // Draw a sample from the uniform distribution
    $a = \min\left[\frac{p(\boldsymbol{\theta}^{t+1}|\mathcal{D})T'(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)}{p(\boldsymbol{\theta}^t|\mathcal{D})T'(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1})}, 1\right]$
    if $u \leq a$ then
        $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{'}$
    else
        $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t$
    end if
end for
```

There are other popular MCMC methods such as Hamiltonian Monte Carlo [60] which uses the gradient of the posterior to navigate in the parameter space. Interestingly in Hamiltonian Monte Carlo, to make sure samples are drawn from the true posterior an internal Metropolis-Hastings step is required.

Finally, the sampling methods in general operate independent of the final expectation that is to be computed. In Chapter 5, we will discuss a Monte Carlo method that improves sampling by considering the loss function involved.

2.4 Bayesian Decision Theory and Preference Learning

2.4.1 Preferences and Existence of the Utility

So far, we have discussed the fundamentals of Bayesian decision theory and modeling state uncertainty and learning posterior belief distribution from data. In this section, we discuss the relation between preference learning and the utility function since utilities are the other major component of expected utilities underlying Bayesian decision theory. [89] have defined axioms that lay the foundation for existence of the utility function when only preferences are observed. In short, given the utility function, the expected utility of the actions indicate the preferences of the decision maker about those actions. These axioms provide conditions under which given preferences indicate existence of an underlying utility function.

As mentioned before, utility function $u(a, \boldsymbol{\theta})$ denotes the "value" of each action at the given state $\boldsymbol{\theta}$. For decision making we are interested in the action with the maximum expected utility: If action a_1 is preferred to action a_2 then $\mathrm{EU}_u(a_1) > \mathrm{EU}_u(a_2)$. This preference is determined based on the known utility function of the decision maker.

Assume we are given an ordering of k actions $a_1, a_2, \dots, a_k$ for a decision maker, then we can define the following properties:

1. **(Completeness)** For *strict orders* and *equivalence*, denoted by $\succ$ and $\sim$ respectively, we have only one of the following:

   - $a_1 \succ a_2$ or $a_2 \succ a_1$ or $a_1 \sim a_2$ (i.e. a_1 is preferred, or a_2 is preferred, or the decision maker is indifferent). Corresponding rewards are then $\mathrm{EU}_u(a_1) > \mathrm{EU}_u(a_2)$ or $\mathrm{EU}_u(a_2) > \mathrm{EU}_u(a_1)$ or $\mathrm{EU}_u(a_1) = \mathrm{EU}_u(a_2)$.

2. **(Transitivity)** for *total ordering* denoted by $\succeq$ (strictly preferred or indifferent), we have

   - If $a_1 \succeq a_2$ and $a_2 \succeq a_3$, then $a_1 \succeq a_3$ (i.e. $\mathrm{EU}_u(a_1) \geq \mathrm{EU}_u(a_2)$ and $\mathrm{EU}_u(a_2) \geq \mathrm{EU}_u(a_3)$, then $\mathrm{EU}_u(a_1) \geq \mathrm{EU}_u(a_3)$). It is a particularly important axiom as it specifies the necessary conditions for decision making. Without transitivity, we could have cycles in the preferences that lead to contradiction and prevent selection of the optimal action.

3. **(Continuity)** If $a_1 \preceq a' \preceq a_2$, then there exists a probability p such that

   - $p\mathrm{EU}_u(a_1) + (1-p)\mathrm{EU}_u(a_2) = \mathrm{EU}_u(a')$. Any action that is between a total ordering of two other actions has a utility that is the convex combination of both.

4. **(Independence)** If $a_1 \prec a_2$, then for any action a' and probability p,

   - $p\mathrm{EU}_u(a_1) + (1-p)\mathrm{EU}_u(a') \prec p\mathrm{EU}_u(a_2) + (1-p)\mathrm{EU}_u(a')$.

Under these conditions, for a rational decision maker there *exists* a utility function for which we have:

$$a_1 \prec a_2 \iff \mathrm{EU}_u(a_1) < \mathrm{EU}_u(a_2).$$

In other words, the outcome reward corresponds to the expected utility of the given action.

In essence, these sets of axioms underpin the existence of the utility function for this set of orderings or preferences. Hence we have now a two way relationship: higher expected utility indicates preference. Conversely, if an action is preferred to another one the expectation of some utility for this action is higher.

It is also noteworthy to consider the *rationality of the decision maker* when dealing with these axioms. Consider the following [45, 77]:

1. Not every decision maker provides the complete ordering.

2. Transitivity is not always preserved with real world scenarios.

3. From few orderings of individual decision makers, we cannot conclude general ordering of actions.

4. People change their preferences towards risk taking at times.

In general, human decision making is not always fully rational in the sense that it not always maximizes an expected utility. As such, maximization of the utility is not a principle that adequately describes human behavior. In spite of these criticisms, decision theory provides the underlying idealized axiomatic framework for rational decision making under uncertainty, regardless of human's notion of rationality.

2.4.2 Risk-seeking vs. Risk-averse Behavior

Although the Von Neumann and Morgenstern axioms specify the existence of the utility, they do not provide any further information as to what that function looks like. Thus, the utility function can be arbitrarily complex. Assume the state is the money, then there is a connection between the shape of the utility function and the attitude of the decision maker: a concave utility function indicate a *risk-averse* behavior. A risk-averse agent dislikes risky cases and prefers actions with a modest certain reward over actions with high uncertain ones. Conversely, a convex utility indicates a *risk-seeking* behavior where agent is happy to bet on large uncertain sums at the risk of no reward. Finally, a *risk-neutral* agent has a linear utility.

2.4.3 Learning the Utility Function

Preference learning can be formulated as the problem of finding this utility function. The common practice is to constrain the utility function of a user to have a specific structure. For instance, one can assume that the utility function is comprised of *additive utilities* as a linear combination of *subutilities*. Each subutility evaluates a single feature (e.g. age) and a linear combination of them form the utility function. Then weights are sought that construct a utility whose expectation matches the preferences. In a more general setting, generalized additive utilities can be learned as a sum over subsets of the features from preferences [18, 22, 38].

One can model the utility of each action by considering the fact that if action a_1 is preferred to a_2, its utility is higher $u(a_1, \boldsymbol{\theta}) > u(a_2, \boldsymbol{\theta})$. Hence, utility $u(a_i, \boldsymbol{\theta})$ becomes a random variable for each action in the *random utility model*. Then, for instance, Bradley-Terry [41] models the preferences as independent pairwise orderings with the probability of action a_i being preferred to a_j as

$$p(a_i \succ a_j) \quad = \quad \frac{u(a_i, \boldsymbol{\theta})}{u(a_i, \boldsymbol{\theta}) + u(a_j, \boldsymbol{\theta})}, \tag{2.9}$$

and for a set of preferences, we have pairwise independence as:

$$p(a_1 \succ \ldots \succ a_k) \quad = \quad p(a_1 \succ a_2) \cdots p(a_1 \succ a_k).$$

In some cases, utility is assumed to have an exponential form as $u(a_i, \boldsymbol{\theta}) = e^{f(\boldsymbol{\theta};\boldsymbol{\lambda}_i)}$ for some function f and the utility is learned by finding the optimal $\boldsymbol{\lambda}$. Through maximizing the probability of the preferred actions in Equation 2.9, the unknown parameters of the utility $\boldsymbol{\lambda}$ can be found.

An extension of Bradley-Terry model is Plackett-Luce [50, 68] where the ordering is defined as a joint probability:

$$p(a_i \succ \ldots \succ a_k) \quad = \quad \frac{u(a_i, \boldsymbol{\theta})}{u(a_i, \boldsymbol{\theta}) + \ldots + u(a_k, \boldsymbol{\theta})}.$$

In these models, the parameters (or the utility itself) are learned so that, among all permutations, the ordering induced maximizes the likelihood of the observed preferences. Placket-Luce model is related to the Gumbel distribution which is in exponential family [96]. Based on this relation, [4] proposed a maximum likelihood approach to learning the preference orderings.

By defining a prior over the parameters, the Bayesian extensions of these approaches are also investigated. For example, [37] used the Gumbel distribution as the likelihood model in the exponential family with non-conjugate priors with EP used for approximate inference. Extensions to nonparametric models have also been developed, in particular, [24] and [19] proposed a preference learning framework based on Gaussian processes. Multi-user Gaussian process-based preference models have been given by [11] and [14]. One advantage of using nonparametric Bayesian models is that they are able to scale with the observations to capture the complexity of the utility function.

Utilities found by these algorithms covered in this section are used for calculating the expected utility. However, this utility function is uncertain by itself and the optimal action has to be selected such that this uncertainty is taken into account. Using the expected expected utility (EEU) framework [15], we can determine the optimal action by using the posterior obtained from an observed set of preferences $\mathcal{D}$ and compute the expected utility of a given action, i.e.

$$\text{EEU}(a) = \int \text{EU}_u(a) p(u|\mathcal{D}) du. \tag{2.10}$$

We will discuss efficient learning of this posterior of the utility given preferences in Chapter 3 and 4. Subsequently in Chapter 5 we discuss how to efficiently compute the expectation using sampling in EEU.

2.5 Summary

In this chapter we covered decision theory and formally introduced its formulation. We also presented modeling of the uncertainty using probability theory and in particular Bayesian methods. We argued Bayesian decision theory is a compelling framework for decision making because of its ability to deal with uncertainty. We also presented the Expected expected utility (EEU) framework as an extension of conventional decision theory for problems where the utility function itself is uncertain. We discussed how to represent and learn the distribution of the unknown variables in parametric and nonparametric Bayesian models. Further, we drew the connection between the Bayesian decision theory and preference learning based on the Von Neumann and Morgenstern axioms.

Building on the foundation of the material covered in this chapter, in the rest of this thesis we will discuss how to learn the distribution of the utilities and then compute the expected utility efficiently. In particular, in the subsequent chapter, we will introduce an efficient Dirichlet Process mixture of Gaussian Processes (GPs) where we leverage the distribution of user communities to learn a compact nonparametric distribution of the utilities from the preferences. In Chapter 4, we use a decision theoretic approach for sparsifying the GP model in order to preserve optimal decisions while ensuring tractable expected utility computations. Finally in Chapter 5, we present a Monte Carlo framework by deriving an optimal loss-calibrated proposal distribution for sampling and show how it can be used for finding the optimal action with a fraction of samples required to compute the expected utilities.

Chapter 3

Learning Community-based Preferences

In the background chapter we covered Bayesian decision theory and preference learning. As discussed, due to the uncertain nature of the utility function in many applications, we need to learn it from the data. In this chapter, we will discuss how to model and learn the belief over the utility function from the preferences.

Preference learning has become an important subfield in machine learning transcending multiple disciplines such as economics, operations research and social sciences. A wide range of applications in areas such as recommender systems, autonomous agents, human-computer interaction and e-commerce has motivated machine learning researchers to investigate flexible and effective ways to construct predictive preference models from preference observations [30].

This is a challenging problem since complex relations between users and their preferred items must be uncovered. Furthermore, flexible and principled ways to handle uncertainty over the users' preferences are required in order to balance what the system knows. To address these challenges, non-parametric Bayesian approaches based on Gaussian processes (GPs) [75] have shown to be effective in real applications [14, 23, 69, 93]. However, one of the major limitations of preference learning approaches based on GPs is their cubic time complexity in both the number of users and items.

While the number of items in many preference prediction applications may be computationally manageable in a GP framework, the number of users may be much larger and often poses the greatest computational challenge. Fortunately, it is well-known that preferences across a user population often decompose into a smaller number of communities of commonly shared preferences [70].

To exploit community structure in preferences, we propose a novel mixture model of GP-based preference learning. While we might first take a finite mixture model approach to modeling community-based preferences, we note that one of the most difficult parts of modeling communities is determining their number. Fortunately, we can exploit the Dirichlet Process [56] framework to build infinite mixture models where the number of mixture components (communities) is inferred automatically. Hence, we model user preferences as an infinite Dirichlet Process (DP) mixture of communities and learn (a) the expected number of preference communities represented in the data, (b) a GP-based preference model over items within each community, and (c) the mixture weights representing each user's fraction of community membership. This results in a learning and inference process that scales linearly in the number of users rather than cubicly and as a side benefit provides the ability to analyze individual communities of preference while also learning how user preferences align with each community and each other.

In the rest of this chapter, first we discuss a Gaussian process framework for learning the distribution of the utilities that utilizes multiple users' information to build the prior. Using multi-user preferences ensures the correlation between users' preferences are jointly modeled. Then we present its extension to a community-based preference leaning model that learns the infinite mixture of Gaussian processes. This model amounts to sharing the utility for community members. We will then discuss the empirical evaluation of our approach and finally conclude with related work and summary.

3.1 Gaussian process for Multi-user Preference Learning

In this section, we define a general approximation framework for Bayesian preference learning via Gaussian Processes that we will adapt for learning communities in the next section.

Let $U = \{\omega_1, \omega_2, \dots, \omega_n\}$ be a set of n users and let $X = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ be a set of m items and denote the set of observed preferences of each user $\omega \in U$ with $\mathcal{D}^\omega = \{\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega\}$ where $1 \leq i \leq m$ and $1 \leq j \leq m$. Given the preferences $\mathcal{D}^\omega$ for ω, satisfaction of the von Neumann-Morgenstern axioms ([63]) justify the existence of utilities $u_i^\omega \in \mathbb{R}$ for each item $\mathbf{a}_i \in X$ s.t. $\mathbf{a}_i \succ \mathbf{a}_j \in D^\omega$ iff $u_i^\omega > u_j^\omega$. In order to model the distribution over these utilities, we build upon the model proposed by [14]. We denote a latent utility vector $\boldsymbol{u}$ for *all* users and items with $\boldsymbol{u} = [u_1^{\omega_1}, u_2^{\omega_1}, \dots, u_m^{\omega_n}]^T$. Then, we can define the likelihood over all the preferences given the latent functions as:

$$p(\mathcal{D}|\boldsymbol{u}) = \prod_{\omega \in U} \prod_{\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega \in \mathcal{D}^\omega} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega), \tag{3.1}$$

with

$$p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega) = \int \mathbb{I}[u_i^\omega - u_j^\omega \geq \epsilon] \mathcal{N}(\epsilon; 0, \alpha^2) = \Phi\left(\frac{u_i^\omega - u_j^\omega}{\alpha}\right) \tag{3.2}$$

where $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(y) dy$ and $\mathcal{N}(y)$ is a zero-mean Gaussian distribution with unit variance. In this model, $p(\boldsymbol{u})$ is the prior over the latent utilities $\boldsymbol{u}$ and is defined via a GP with zero-mean function and a covariance function that factorizes over users and items [14]. Therefore:

$$p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; \mathbf{0}, \mathbf{K}), \quad \mathbf{K} = \mathbf{K}_u \otimes \mathbf{K}_x, \tag{3.3}$$

where $\mathbf{K}$ is the kernel matrix composed of the Kronecker product of the kernel matrix over the users $\mathbf{K}_u$ and the kernel matrix over the items $\mathbf{K}_x$. One interesting feature of this model is the inherent transfer of preferences across users through the correlated prior, which will subsequently help the prediction on those users for which there are not many preferences recorded. Additionally, as we shall see later, having a fully factorized likelihood across users and items will facilitate the application of sequential approximate posterior inference methods such as *Expectation Propagation* (EP) as discussed in Section 2.3.3 of last chapter.

The posterior of the latent functions $\boldsymbol{u}$ given all the preferences is:

$$p(\boldsymbol{u}|\mathcal{D}) = \frac{1}{Z} p(\boldsymbol{u}) p(\mathcal{D}|\boldsymbol{u}), \tag{3.4}$$

with Z being the normalizer. This posterior is analytically intractable due to the non-Gaussian nature of the likelihood. Therefore, we need to resort to approximations. Here we use EP which approximates the posterior $p(\boldsymbol{u}|\mathcal{D})$ by a tractable distribution $q(\boldsymbol{u})$. EP assumes that each likelihood term $p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega)$ can be approximated by a distribution $q(u_i^\omega, u_j^\omega)$ such that the approximated posterior $q(\boldsymbol{u})$ factorizes over $q(u_i^\omega, u_j^\omega)$. Then EP iteratively approximates each $q(u_i^\omega, u_j^\omega)$ in turn by dividing it out from the approximated posterior $q(\boldsymbol{u})$ (obtaining the cavity distribution), multiplying in the true likelihood $p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega)$, and projecting the result back to its factorized form by matching its moments to an updated $q(u_i^\omega, u_j^\omega)$. This overall procedure is motivated by the aim to minimize the KL-divergence between the true posterior $p(\boldsymbol{u}|\mathcal{D})$ and its approximation $q(\boldsymbol{u})$.

In the preference learning case we detailed earlier, we can approximate the posterior with a Gaussian:

$$q(\boldsymbol{u}) = \frac{1}{\tilde{Z}} p(\boldsymbol{u}) \prod_{\omega \in U} \prod_{\{\{i,j\} | \mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega \in \mathcal{D}^\omega\}} q(u_i^\omega, u_j^\omega) = \mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.5}$$

We are interested in locally approximating each likelihood term in Equation 3.1 as:

$$\begin{aligned} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega) &\approx q(u_i^\omega, u_j^\omega) \\ &= \tilde{Z}_{i,j}^\omega \mathcal{N}(u_i^\omega, u_j^\omega; \tilde{\boldsymbol{\mu}}_{\omega,[i,j]}, \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}), \end{aligned} \tag{3.6}$$

where $\mathcal{N}(u_i^\omega, u_j^\omega; \tilde{\boldsymbol{\mu}}_{\omega,[i,j]}, \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]})$ denotes the local two-dimensional Gaussian over $[u_i^\omega, u_j^\omega]^T$ with mean $\tilde{\boldsymbol{\mu}}_{\omega,[i,j]}$ and covariance $\tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}$ corresponding to items i and j.

Hence we can approximate the posterior as:

$$q(\boldsymbol{u}) = \frac{1}{\tilde{Z}} p(\boldsymbol{u}) \prod_{\omega \in U} \prod_{\{i,j\} \in \mathcal{D}} q(u_i^\omega, u_j^\omega) = \mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.7}$$

where

$$\boldsymbol{\mu}_{\omega,[i,j]} = \boldsymbol{\Sigma}_{\omega,[i,j]} \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}^{-1} \tilde{\boldsymbol{\mu}}_{\omega,[i,j]} \tag{3.8}$$

$$\boldsymbol{\Sigma}_{\omega,[i,j]}^{-1} = (\mathbf{K}_{\omega,[i,j]}^{-1} + \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}^{-1}). \tag{3.9}$$

This means that in order to determine the parameters of our approximate posterior, we need to compute estimates of the local parameters $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$. To show these updates, we need to define additional distributions: (a) the *cavity* distribution which we will denote with the backslash symbol "\" and (b) the *unnormalized marginal posterior*, which we will denote with the hat symbol " ˆ ".

Here we only show how to compute the parameters necessary to estimate the posterior[1]. We iterate through the following steps:

1. Update the cavity distribution: The cavity distribution $q_\backslash(u_i^\omega, u_j^\omega)$ results from multiplying the prior by all the local approximate likelihood terms except $q(u_i^\omega, u_j^\omega)$ and marginalizing all latent dimensions except u_i^ω and u_j^ω. This is done in practice simply by removing the current approximate likelihood term from the approximate posterior. Hence we obtain:

$$q_\backslash(u_i^\omega, u_j^\omega) = \mathcal{N}(u_i^\omega, u_j^\omega; \boldsymbol{\mu}_{\backslash\omega,[i,j]}, \boldsymbol{\Sigma}_{\backslash\omega,[i,j]}) \tag{3.10}$$

$$\boldsymbol{\mu}_{\backslash\omega,[i,j]} = \boldsymbol{\Sigma}_{\backslash\omega,[i,j]} (\boldsymbol{\Sigma}_{\omega,[i,j]}{}^{-1} \boldsymbol{\mu}_{\omega,[i,j]} - \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}^{-1} \tilde{\boldsymbol{\mu}}_{\omega,[i,j]}) \tag{3.11}$$

$$\boldsymbol{\Sigma}_{\backslash\omega,[i,j]} = (\boldsymbol{\Sigma}_{\omega,[i,j]}^{-1} - \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}^{-1})^{-1}. \tag{3.12}$$

2. Update the unnormalized marginal posterior: This results from finding the unnormalized Gaussian that best approximates the product of the cavity distribution and the exact likelihood:

$$\hat{q}(u_i^\omega, u_j^\omega) \approx p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega) q_\backslash(u_i^\omega, u_j^\omega) \tag{3.13}$$

$$\hat{q}(u_i^\omega, u_j^\omega) = \hat{Z}^{-1} \mathcal{N}(u_i^\omega, u_j^\omega; \hat{\boldsymbol{\mu}}_{\omega,[i,j]}, \hat{\boldsymbol{\Sigma}}_{\omega,[i,j]}) \tag{3.14}$$

with

$$\hat{Z} = \Phi(r_{i,j})$$

$$\hat{\boldsymbol{\mu}}_{\omega,[i,j]} = \boldsymbol{\mu}_{\backslash\omega,[i,j]} + \boldsymbol{\Sigma}_{\backslash\omega,[i,j]} \mathbf{w}_{\omega,[i,j]} \tag{3.15}$$

$$\hat{\boldsymbol{\Sigma}}_{\omega,[i,j]} = \boldsymbol{\Sigma}_{\backslash\omega,[i,j]} - \boldsymbol{\Sigma}_{\backslash\omega,[i,j]} (\mathbf{w}_{\omega,[i,j]} \mathbf{w}_{\omega,[i,j]}^\top \hat{r}_{i,j} \mathbf{w}_{\omega,[i,j]} \mathbf{1_1}^\top) \boldsymbol{\Sigma}_{\backslash\omega,[i,j]}, \tag{3.16}$$

[1]Similar updates for the single user case are given in [23].

where

$$\mathbf{w}_{\omega,[i,j]} = \frac{\mathcal{N}(r_{i,j})}{\Phi(r_{i,j})(\alpha^2 + \mathrm{tr}(\boldsymbol{\Sigma}_{\backslash\omega,[i,j]}\mathbf{1_2}))}\mathbf{1_1},$$

and

$$\mathbf{1_1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \mathbf{1_2} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

3. Update the local factor approximation: by performing moment matching, we can calculate the corresponding parameters in $q(u_i^\omega, u_j^\omega)$ as:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{\omega,[i,j]} &= \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]}(\hat{\boldsymbol{\Sigma}}^{-1}_{\omega,[i,j]}\hat{\boldsymbol{\mu}}_{\omega,[i,j]} - \boldsymbol{\Sigma}^{-1}_{\backslash\omega,[i,j]}\boldsymbol{\mu}_{\backslash\omega,[i,j]}) \\ \tilde{\boldsymbol{\Sigma}}_{\omega,[i,j]} &= (\hat{\boldsymbol{\Sigma}}^{-1}_{\omega,[i,j]} - \boldsymbol{\Sigma}^{-1}_{\backslash\omega,[i,j]})^{-1}. \end{aligned} \tag{3.17}$$

At each iteration once we have local factor parameters $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$, we can compute the parameters of the full posterior approximation using 3.7. We iterate through all the factors and update the local approximations sequentially.

3.1.1 Prediction

Given a pair of items $\mathbf{a}_1^*, \mathbf{a}_2^*$ for a particular user, we will be able to determine the predictive distribution over the latent utility functions as:

$$p(u_1^*, u_2^*|\mathcal{D}) = \int_{-\infty}^{\infty} p(u_1^*, u_2^*|\boldsymbol{u})p(\boldsymbol{u}|\mathcal{D})d\boldsymbol{u} = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{C}^*) \tag{3.18}$$

$$\text{with} \quad \boldsymbol{\mu}^* = \mathbf{K}^*(\mathbf{K} + \tilde{\boldsymbol{\Sigma}})^{-1}\tilde{\boldsymbol{\mu}} \tag{3.19}$$

$$\boldsymbol{C}^* = \boldsymbol{\Sigma}^* - \mathbf{K}^{*\top}(\mathbf{K} + \tilde{\boldsymbol{\Sigma}})^{-1}\mathbf{K}^*, \tag{3.20}$$

where $\boldsymbol{\Sigma}^*$ is the 2×2 kernel matrix built from the item pair $\mathbf{a}_1^*$ and $\mathbf{a}_2^*$; $\mathbf{K}^* = \mathbf{K}_u^* \otimes \mathbf{K}_x^*$ that represents the kernel matrix of the test user and items with all the users and items in the training set; $\mathbf{K}_u^*$ is the $1 \times n$ kernel matrix of the queried user with other users; and $\mathbf{K}_x^*$ is the $2 \times m$ kernel matrix of the queried pair of items with other items. Subsequently, their preference for a user is determined by integrating out the latent utility functions:

$$\begin{aligned} p(\mathbf{a}_1^* \succ \mathbf{a}_2^*|\mathcal{D}) &= \int\int p(\mathbf{a}_1^* \succ \mathbf{a}_2^*|u_1^*, u_2^*, \mathcal{D})p(u_1^*, u_2^*|\mathcal{D})du_1^*du_2^* \\ &= \Phi\left(\frac{\mu_1^* - \mu_2^*}{\alpha^2 + C_{1,1}^* + C_{2,2}^* - 2C_{1,2}^*}\right). \end{aligned} \tag{3.21}$$

We see that the mean and covariance of the predictive distribution require the inversion of a (possibly) very large matrix. This matrix is, in general, of dimensions $nm \times nm$. Even though the inverse matrix can be reused for multiple query points, this is intractable for any real application. Hence, we will focus on how to sparsify this matrix by selecting a subset of *inducing items*. Note the main difference with other machine learning settings where there is a one-to-one correspondence between the number of observations and the dimensionality of the corresponding matrix. In our case, the observations (preference relations) affect the dimensionality of this matrix only indirectly and we are more concerned with the number of users and items.

Since our focus is on improving prediction time and this scales cubically with the number of items we need to obtain a risk-sensitive posterior approximation. EP is well-suited for this case because it considers all data efficiently and locally.

3.1.2 Optimizing the Kernel Hyper-parameters

One of the inherent advantages of GPs over other non-Bayesian kernel methods is its capability of optimizing the hyper-parameters. This can be easily done by maximizing the marginal likelihood in a gradient descent algorithm. The marginal likelihood can be obtained from the normalizer $\tilde{Z}$ in Equation 3.5 as:

$$\tilde{Z} = \int p(\boldsymbol{u}) \prod_{\omega \in U} \prod_{\{i,j\} \in \mathcal{D}} q(u_i^\omega, u_j^\omega) d\boldsymbol{u} \tag{3.22}$$

where both $p(\boldsymbol{u})$ and $q(u_i^\omega, u_j^\omega)$ are Gaussian distributions and their product produces an unnormalized Gaussian distribution. Therefore, the log likelihood is:

$$\log(\tilde{Z}) = -\frac{1}{2}\tilde{\boldsymbol{\mu}}^\top (\mathbf{K} + \tilde{\boldsymbol{\Sigma}})^{-1} \tilde{\boldsymbol{\mu}} - \frac{1}{2} \log \det(\mathbf{K} + \tilde{\boldsymbol{\Sigma}}) - \frac{n}{2} \log 2\pi \tag{3.23}$$

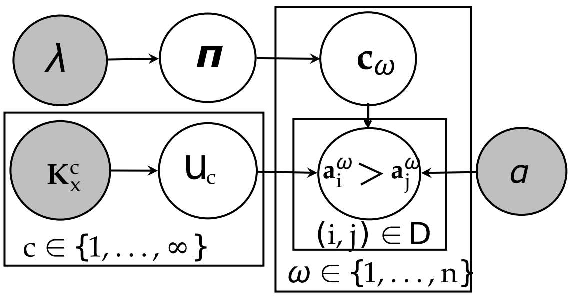
The derivative of the marginal likelihood with respect to the kernel hyper-parameters can be used in a gradient descent algorithm to optimize the kernel.

3.2 Dirichlet Process Mixtures of Community-based Preference GPs

In this section we propose to alter the GP preference based model of Section 3.1 to model users as a (potentially) infinite mixture of GP-based communities. Hence, we now assume there are an infinite number of possible communities of preference $C = \{1 \ldots \infty\}$ where for every user ω there is a latent indicator $c_\omega \in C$ indicating to which community ω belongs. Further, we assume each community $c \in C$ has it's own latent utility function over *items only* $\boldsymbol{u}_c = [u_1^c, u_2^c, \ldots, u_m^c]^T$ since we assume all members of the community share common preferences. Hence, we can now define a likelihood in this model for the preference data conditioned on all latent utilities $\boldsymbol{u} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\infty]$ (where we have redefined $\boldsymbol{u}$ from the previous section) *and* the vector of latent community indicators for each user $\mathbf{c} = [c_{\omega_1}, \ldots, c_{\omega_n}]^\top$:

$$p(\mathcal{D}|\boldsymbol{u}, \mathbf{c}, \alpha) = \prod_{\omega \in U} \prod_{(i,j) \in \mathcal{D}^\omega} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^{c_\omega}, u_j^{c_\omega}, \alpha) \tag{3.24}$$

Figure 3.1: Our proposed generative graphical model for community-based preferences. There is a plate for users ω with community membership indicator $\mathbf{c}_\omega \in C$ and an embedded plate for i.i.d. preference observations $\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega$ of user ω depending on the community assignment $\mathbf{c}_\omega$ of ω, community $\mathbf{c}_\omega$'s latent utility function $\boldsymbol{u}_{c_\omega}$, and the discriminal dispersion parameter α. There is a separate plate for communities $c \in C = \{1 \ldots, \infty\}$ which contains the latent utility function $\boldsymbol{u}_c$ drawn from a Gaussian Process conditioned on local (optimized) community parameters $\mathbf{K}_x^c$. Finally, each $\mathbf{c}_\omega$ is generated i.i.d. from an infinite multinomial distribution with parameters $\boldsymbol{\pi}$ and Dirichlet Process prior with concentration parameter λ.


λ
$\boldsymbol{\pi}$
$\mathbf{c}_\omega$
K_x^c
U_c
$\mathbf{a}_i^\omega > \mathbf{a}_j^\omega$
α
$(i, j) \in D$
$c \in \{1, \ldots, \infty\}$
$\omega \in \{1, \ldots, n\}$

with $p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^{c_\omega}, u_j^{c_\omega}, \alpha)$ as defined in the previous section and a community-specific kernel prior $\mathbf{K}_x^c$ for *each* $\boldsymbol{u}_c$ ($c \in C$) over *items only*.

$$p(\boldsymbol{u}_c | \mathbf{K}_x^c) = \mathcal{N}(\boldsymbol{u}_c; \mathbf{0}, \mathbf{K}_x^c). \tag{3.25}$$

It is important to emphasize here that one important feature of our infinite community-based mixture model is that the hyper-parameters K_x^c of the utilities u_c are community-dependent and can be optimized as part of the learning process as described in Section 3.1.2 leading to improved community modeling and generalization over all item preferences within that community.

We assume that $\mathbf{c}$ is generated according to an infinite multinomial distribution with parameters $\boldsymbol{\pi} \in \mathbb{R}^{|C|}$ ($\sum_{i=1}^{|C|} \pi_i = 1$), hence

$$p(\mathbf{c}|\boldsymbol{\pi}) = \prod_{i=1}^{|C|} \pi_i^{n_i} \tag{3.26}$$

where

$$n_i = \sum_{\omega \in U} \delta_{c_\omega = i}.$$

indicating the number of users in the community c_ω.

Since the number of possible communities $|C|$ is infinite, we resort to Dirichlet processes by defining an infinite Dirichlet prior on the community distribution with concentration parameter λ [58, 72]:

$$\begin{aligned} p(\boldsymbol{\pi}|\lambda) = p(\pi_1,\ldots,\pi_{|C|}|\lambda) &= \text{Dirichlet}(\lambda/|C|,\ldots,\lambda/|C|) \\ &= \frac{\Gamma(\lambda)}{\Gamma(\lambda/|C|)^{|C|}} \prod_{j=1}^{|C|} \pi_j^{\lambda/|C|-1}. \end{aligned} \tag{3.27}$$

Altogether this generative framework is represented in the graphical model of Figure 3.1.

Our primary goal in inference is to obtain a sample posterior estimate over $\boldsymbol{u}$ and $\mathbf{c}$ to be used for future prediction. To this end, we begin by multiplying all of the likelihood and prior factors of our generative model in Figure 3.1 to obtain a superset of the desired joint posterior parameters:

$$p(\boldsymbol{u},\mathbf{c},\boldsymbol{\pi}|\mathcal{D},\mathbf{K}_x,\lambda,\alpha) \;\propto\; \left[\prod_{c\in C} p(\boldsymbol{u}_c|\mathbf{K}_x^c)\right] \cdot \tag{3.28}$$
$$\left[\prod_{\omega\in U} \underbrace{\left[\prod_{(i,j)\in\mathcal{D}^\omega} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | f_i^{c_\omega}, f_j^{c_\omega}, \alpha)\right]}_{p(\mathcal{D}^\omega|\boldsymbol{u},\mathbf{c}_\omega,\alpha)} p(\mathbf{c}_\omega|\boldsymbol{\pi})\right] p(\boldsymbol{\pi}|\lambda)$$

Here of course, we don't necessarily require a posterior estimate over $\boldsymbol{\pi}$ and we will see in Section 3.2.2 that it is in fact important to marginalize over $\boldsymbol{\pi}$ in the posterior to facilitate Gibbs sampling inference for Dirichlet processes.

However, before we dive into specific details, we first provide a general overview of our posterior inference framework. To perform inference in this Dirichlet Process mixture of GPs, we utilize the joint probability in Equation 3.28 and devise a collapsed, blocked Gibbs sampler that breaks the Gibbs sampling inference into two distinct steps, for which *different* inference algorithms are appropriate. Specifically, collapsing comes from marginalizing over $\boldsymbol{\pi}$ and blocking stems from repeating joint inference of $\boldsymbol{u}$ given $\mathbf{c}$. Gibbs sampling then repeats as follows until convergence:

- For each $\mathbf{c}_\omega$, infer $p(\mathbf{c}_\omega|\mathbf{c}_{\backslash\omega},\boldsymbol{u},\mathcal{D},\lambda,\alpha,\mathbf{K})$ via Gibbs sampling as discussed in Section 3.2.2.

- Infer $p(\boldsymbol{u}|\mathbf{c},\mathcal{D},\lambda,\alpha,\mathbf{K})$ via Expectation Propagation (EP) as discussed in Section 3.2.1 with hyper-parameters optimized as discussed in Section 3.1.2.

Here we have merged all kernel hyper-parameters into the set $\mathbf{K} = \{\mathbf{K}_x^c | c \in C\}$.

3.2.1 Inferring Community utilities

The posterior of the utility (latent) functions $\boldsymbol{u}$ given all the preferences and parameters $\boldsymbol{\theta}$ is:

$$p(\boldsymbol{u}|\mathbf{c},\mathcal{D},\boldsymbol{\theta}) = \frac{1}{Z} p(\mathcal{D}|\boldsymbol{u},\mathbf{c},\boldsymbol{\theta}) p(\boldsymbol{u}|\mathbf{c}) \tag{3.29}$$

Since the likelihood is factorized we again take advantage of the sequential approximation EP. EP approximates the posterior $p(\boldsymbol{u}|\mathbf{c}, \mathcal{D}, \boldsymbol{\theta})$ by a tractable distribution $q(\boldsymbol{u}|\mathbf{c})$ which this time depends on the community. EP assumes that each likelihood term $p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega, \mathbf{c}_\omega = c, \mathbf{K}_\mathbf{a})$ can be approximated by a distribution $q(u_i^\omega, u_j^\omega | \mathbf{c}_\omega = c, \mathbf{K}_\mathbf{a})$ such that the approximated posterior $q(\boldsymbol{u}|\mathbf{c})$ factorizes over $q(u_i^\omega, u_j^\omega|\mathbf{c}_\omega)$. Then EP iteratively approximates each $q(u_i^\omega, u_j^\omega|\mathbf{c}_\omega)$ in turn by dividing it out from the approximated posterior $q(\boldsymbol{u}|\mathbf{c})$ (obtaining the cavity distribution), multiplying in the true likelihood $p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | u_i^\omega, u_j^\omega, \mathbf{c}_\omega, \mathbf{K}_\mathbf{a})$, and projecting the result back to its factorized form by matching its moments to an updated $q(u_i^\omega, u_j^\omega|\mathbf{c}_\omega)$.

In the preference learning case we detailed earlier, we can approximate the posterior with a Gaussian:

$$\begin{aligned} q(\boldsymbol{u}|\mathbf{c}) &= \prod_c \frac{1}{\tilde{Z}^c} \prod_{\omega \in U} p(\boldsymbol{u}|\mathbf{c}_\omega = c) \prod_{\{\{i,j\}|\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega \in \mathcal{D}^\omega\}} q(f_i^\omega, f_j^\omega | \mathbf{c}_\omega = c) \\ &= \mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}^c, \boldsymbol{\Sigma}^c). \end{aligned} \tag{3.30}$$

where $\boldsymbol{\mu}^c$ and $\boldsymbol{\Sigma}^c$ denote the mean and covariance of the Gaussian distribution for the community user ω belongs to corresponding to $\boldsymbol{\theta}_{\mathbf{c}_\omega}$. We are interested in locally approximating each likelihood term as:

$$\begin{aligned} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega | f_i^\omega, f_j^\omega, \mathbf{c}_\omega) &\approx q(f_i^\omega, f_j^\omega|\mathbf{c}_\omega) \qquad (3.31) \\ &= \tilde{Z}_{i,j}^\omega \mathcal{N}(f_i^\omega, f_j^\omega; \tilde{\boldsymbol{\mu}^c}_{\omega,[i,j]}, \tilde{\boldsymbol{\Sigma}^c}_{\omega,[i,j]}), \end{aligned}$$

where $\mathcal{N}(f_i^\omega, f_j^\omega; \tilde{\boldsymbol{\mu}^c}_{\omega,[i,j]}, \tilde{\boldsymbol{\Sigma}^c}_{\omega,[i,j]})$ denotes the local two-dimensional Gaussian over $[f_i^\omega, f_j^\omega]^\top$ with mean $\tilde{\boldsymbol{\mu}^c}_{\omega,[i,j]}$ and covariance $\tilde{\boldsymbol{\Sigma}^c}_{\omega,[i,j]}$ corresponding to items i and j.

Hence we can approximate the posterior in Equation 3.5 with the following parameters:

$$\boldsymbol{\mu}^c{}_{\omega,[i,j]} = \boldsymbol{\Sigma}^c{}_{\omega,[i,j]} \tilde{\boldsymbol{\Sigma}}^{c}{}^{-1}_{\omega,[i,j]} \tilde{\boldsymbol{\mu}^c}_{\omega,[i,j]} \tag{3.32}$$

$$\boldsymbol{\Sigma}^{c}{}^{-1}_{\omega,[i,j]} = (\mathbf{K}^{c}{}^{-1}_{\omega,[i,j]} + \tilde{\boldsymbol{\Sigma}}^{c}{}^{-1}_{\omega,[i,j]}). \tag{3.33}$$

One advantage of our model is that the hyper-parameters of the model can be learned independently for each community. Even though we can define distributions over covariance (Inverse-Wishart), here due to the computational cost of a full Bayesian update of $\mathbf{K}$ we instead optimize the hyper-parameters by maximizing the marginal likelihood in a gradient descent algorithm. The marginal likelihood can be obtained from the normalizer $\tilde{Z}^c$ in Equation 3.30 similar to that of explained in Section 3.1.2 although the kernels are now smaller and is only defined on the members of the community.

3.2.2 Inferring Community Membership

In our Gibbs sampler, given $\boldsymbol{u}$, we now wish to sample $\mathbf{c}$ – the community memberships for all users. Assuming that our blocked Gibbs sampler has already provided us with a sample of $\boldsymbol{u}$ for some fixed $\mathbf{c}^*$ sampled on the previous iteration, we now wish to sample each new $\mathbf{c}_\omega$ in turn for the current iteration provided that we can define $p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \mathcal{D}, \lambda, \alpha)$.

While we could sample $\boldsymbol{u}$ from $p(\boldsymbol{u}|\mathbf{c}^*, \mathcal{D}, \lambda, \alpha)$ to compute $p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \mathcal{D}, \lambda, \alpha)$, this seems inefficient given that we can derive the full posterior $p(\boldsymbol{u}|\mathbf{c}^*, \mathcal{D}, \lambda, \alpha)$ in closed-form given our Gaussian Process inference machinery in Section 3.2.1. So instead we propose to compute $\mathbb{E}_{p(\boldsymbol{u}|\mathbf{c}^*,\mathcal{D},\lambda,\alpha)}[p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \mathcal{D}, \lambda, \alpha)]$.[2]

Now we derive an efficiently computable closed-form for sampling $\mathbf{c}_\omega$ where we abbreviate the previous expectation to the shorter form $\mathbb{E}_{\boldsymbol{u}|\mathbf{c}^*}[p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \mathcal{D}, \lambda, \alpha)]$:[3]

$$\mathbb{E}_{\boldsymbol{u}|\mathbf{c}^*}[p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \mathcal{D}, \lambda, \alpha)] \propto \mathbb{E}_{\boldsymbol{u}|\mathbf{c}^*}\left[\int p(\underbrace{\mathbf{c}_\omega, \mathbf{c}_{\setminus\omega}}_{\mathbf{c}}, \boldsymbol{u}, \boldsymbol{\pi}|\mathcal{D}, \lambda, \alpha) d\boldsymbol{\pi}\right] \tag{3.34}$$

$$= \mathbb{E}_{\boldsymbol{u}|\mathbf{c}^*}\left[\underbrace{\prod_{\omega\in U}\prod_{(i,j)\in\mathcal{D}^\omega} p(\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega|u_i^\omega, u_j^\omega, \alpha)}_{p(\mathcal{D}^\omega|\boldsymbol{u},\mathbf{c}_\omega,\alpha)} p(\mathbf{c}_\omega|\boldsymbol{\pi})\right] p(\boldsymbol{\pi}|\lambda) d\boldsymbol{\pi} \tag{3.35}$$

$$\propto \mathbb{E}_{\boldsymbol{u}|\mathbf{c}^*}[p(\mathcal{D}^\omega|\boldsymbol{u}, \mathbf{c}_\omega, \alpha)] \underbrace{\int p(\mathbf{c}|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\lambda) d\boldsymbol{\pi}}_{p(\mathbf{c}|\lambda)\propto p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega},\lambda)} \tag{3.36}$$

$$\propto \left[\int \underbrace{p(\mathcal{D}^\omega|\boldsymbol{u}, \mathbf{c}_\omega, \alpha)}_{\text{Likelihood}} \underbrace{p(\boldsymbol{u}|\mathbf{c}^*, \mathcal{D}, \lambda, \alpha)}_{\text{Gaussian Process}}] d\boldsymbol{u}\right] \underbrace{p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega}, \lambda)}_{\text{Dirichlet Process}} \tag{3.37}$$

In this derivation, we note the proportionalities can be introduced anytime a rewrite induces a constant normalizer that is independent of $\mathbf{c}_\omega$ since this can be absorbed into the global normalizer. Since $\mathbf{c}_\omega$ is discrete, the normalization can be easily computed on demand when required.

In (3.34), we rewrote the conditional in terms of the full joint since the normalizer required to obtain the LHS is $p(\mathbf{c}_{\setminus\omega}, \boldsymbol{u}, \boldsymbol{\pi}|\mathcal{D}, \lambda, \alpha)$, which is independent of $\mathbf{c}_\omega$.

In (3.35), we expanded the full joint into its definition from our graphical model in Figure 3.1.

In (3.36), we removed the product for $\omega' \in U \setminus \{\omega\}$ since the likelihood of each user ω''s preferences only depend on $\mathbf{c}_{\omega'}$ and hence these likelihoods are a constant

[2] Of course, one can always sample $\boldsymbol{u}$ and avoid this expectation if preferred, but we conjecture that using the expectation will induce a lower-variance Gibbs sampling process with faster convergence.

[3] A detailed derivation of all math derived in these sections is provided in an online appendix at the authors' web pages.

w.r.t. $\mathbf{c}_\omega$ for $\omega \neq \omega'$. We also note that the expectation over $\boldsymbol{u}$ only applies to terms involving $\boldsymbol{u}$ and likewise the integral over $\boldsymbol{\pi}$ only applies to terms involving $\boldsymbol{\pi}$.

In (3.37), we expanded the definition of the expectation and replaced $p(\mathbf{c}|\lambda)$ with $p(\mathbf{c}_\omega|\mathbf{c}_{\setminus\omega},\lambda)$ since the latter only has a normalizer $p(\mathbf{c}_{\setminus\omega}|\lambda)$ which is independent of $\mathbf{c}_\omega$.

Thus in (3.37) we arrive at a closed-form computation that is straightforward to compute. In the square brackets, we need only use our GP posterior $f|\mathbf{c}^*$ to compute the product of the probabilities of each of user ω's preferences $\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega \in \mathcal{D}^\omega$ as defined in the next sections. This leaves us only to compute $p(\mathbf{c}_\omega = c|\mathbf{c}_{\setminus\omega},\lambda)$ as is standard in Gibbs sampling for Dirichlet processes:

1. If c is an *active community* ($\exists \mathbf{c}_\omega \in \mathbf{c}_{\setminus\omega}$ s.t. $\mathbf{c}_\omega = c$), then

$$p(\mathbf{c}_\omega = c|\mathbf{c}_{\setminus\omega},\lambda) = \frac{\sum_{\omega'\neq\omega}\mathbb{I}[\mathbf{c}_{\omega'} = c]}{N-1+\lambda} \tag{3.38}$$

   where N is the number of non-empty communities.

2. Else c is a *new community* so

$$p(\mathbf{c}_\omega = c|\mathbf{c}_{\setminus\omega},\lambda) = \frac{\lambda}{N-1+\lambda} \tag{3.39}$$

Hence all quantities required to sample $\mathbf{c}_\omega$ have now been defined permitting sampling of each $\mathbf{c}_\omega$ in turn to complete the community process sampling portion of the Gibbs sampling inference for our model. And the result is intuitive: a user ω is more likely to join a community which provides a higher likelihood on its preference data D^ω. Additionally, this sampling process displays the well-known "rich-get-richer" effect of Dirichlet Processes since communities with more members have a higher probability of being selected.

3.2.3 Prediction

Given a pair of items $\mathbf{a}_1^*, \mathbf{a}_2^*$ for a particular user, we will be able to determine the predictive distribution over the latent utility functions as:

$$\begin{aligned} p(u_1^*, u_2^*|\mathcal{D},\alpha,\mathbf{K},\omega) &= \int_{-\infty}^{\infty} p(u_1^*,u_2^*|\boldsymbol{u}^{\mathbf{c}_\omega},\alpha,\mathbf{K}^c)p(\boldsymbol{u}^{\mathbf{c}_\omega}|\mathcal{D},\alpha,\mathbf{K}^c)d\boldsymbol{u}^{\mathbf{c}_\omega} \\ &= \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{C}^*), \end{aligned}$$

with

$$\begin{aligned} \boldsymbol{\mu}^* &= \mathbf{K}^*(\mathbf{K}^c + \tilde{\boldsymbol{\Sigma}^c})^{-1}\boldsymbol{\mu}^c & (3.40) \\ \boldsymbol{C}^* &= \boldsymbol{\Sigma}^* - \mathbf{K}^{*\top}(\mathbf{K}^c + \tilde{\boldsymbol{\Sigma}^c})^{-1}\mathbf{K}^*, & (3.41) \end{aligned}$$

where $\boldsymbol{\Sigma}^*$ is the 2×2 kernel matrix built from the item pair $\mathbf{a}_1^*$ and $\mathbf{a}_2^*$; $\mathbf{K}^*$ represents the kernel matrix of test items with all the items in the training set; $\mathbf{K}_c^*$ is the kernel matrix of the queried user with other users in the same community; and $\mathbf{K}_x^*$ is the $2 \times m$ kernel matrix of the queried pair of items with other items. Subsequently, their preference for a user is determined by integrating out the latent utility functions we have $p(\mathbf{a}_1^* \succ \mathbf{a}_2^* | \mathcal{D}, \alpha, \mathbf{K})$ equals:

$$\sum_{\mathbf{c}} p(\mathbf{c}|\lambda) \int \int p(\mathbf{a}_1^* \succ \mathbf{a}_2^* | u_1^*, u_2^*, \mathbf{c}, \alpha, \mathbf{K}) p(u_1^*, u_2^* | \mathcal{D}, \mathbf{c}, \alpha, \mathbf{K}) du_1^* u_2^*$$
$$= \sum_{\mathbf{c}} p(\mathbf{c}|\lambda) \Phi \left(\frac{\mu_1^* - \mu_2^*}{\alpha^2 + C_{1,1}^* + C_{2,2}^* - 2C_{1,2}^*} \right).$$

We see that the mean and covariance of the predictive distribution require the inversion of a much smaller matrix because only observations in each community have to be considered. This leads to $O(n^3)$ time complexity (cubic in the number of observations) compared to the case where the community of the users is not considered in Section 3.1.1 which has $O(n^3 m^3)$ time complexity.

Algorithm 3.1 Blocked Gibbs Sampling Routine

```
input: $X, U, \mathcal{D}, \lambda, \alpha$
Initialize $\mathbf{c}$ to arbitrary assignments for each user
while not converged do
    // Infer community utilities and hyper-parameters
    for $\mathbf{c} \in C$ do
        1. Obtain $\mathbf{K}^c$ similar to Section 3.1.2 conditioned on the $\mathbf{c}$.
        2. Perform EP to infer $p(\boldsymbol{u}|\mathbf{c}, \mathcal{D}, \alpha, \mathbf{K})$ (Section 3.2.1).
    end for
    // Sample community membership assignments
    for $\omega \in U$ do
        3. Sample $\mathbf{c}_\omega$ from Eq. 3.37.
    end for
end while
```

3.2.4 Final Algorithm

Having found the communities and users' utilities, the algorithm for community-based preference learning is presented in Algorithm 3.1. After initializing with class assignments for each user, hyper-parameters for each community GP are optimized followed by inference of $\boldsymbol{u}$ and then $\mathbf{c}$, which repeats until convergence.

3.3 Empirical Evaluation

In this section we empirically evaluate the performance of our algorithm to determine (1) how well the DP mixture of GPs is capable of reducing prediction time vs. Full-GP defined in Section 3.1, (2) how well the DP mixture of GPs works in accurately learning the preferences of each user compared to Full-GP, (3) whether

Table 3.1: Performance results for Synthetic, Sushi and AMT Car datasets.

Algorithm	**Dataset**	**Accuracy %**	**Time(s)**
Full-GP, Section 3.1	Synthetic	95.17 ± 3.33	0.05
	Sushi	62.13 ± 5.69	2.10
	Car	64.00 ± 8.94	0.89
DP Mixture of GPs Section 3.2	Synthetic	100 ± 0	0.04
	Sushi	62.04 ± 5.41	0.09
	Car	64.17 ± 6.97	0.10

each community learned has a distinct set of preferences, and (4) how effectively the true number of communities in data is recovered. We perform our evaluation on three datasets: one synthetic and two real-world datasets. The synthetic experiment assesses the effectiveness of our approach in a controlled setting and the real-world datasets include the data obtained from preferences over cars that we have created using Amazon Mechanical Turk as well as a publicly available sushi preference dataset. We compute the accuracy as the percentage of correctly predicted test preferences to the whole set.

It is assumed that we are given a set of users and items with their corresponding features. For each user and item we also augment their features with their ID index, that is, a vector that is only one for that particular user or item (this is a common practice in collaborative filtering). The pairwise item preferences of each user are split into two sets for performing the inference (60%), and testing (40%) where the algorithm performance is evaluated. We use the squared exponential covariance with automatic relevance determination (ARD) for both users and items. We manually tuned λ and α (for Full-GP as well).

Synthetic Dataset: We generated a hypothetical set of 60 users and 10 items and assigned each user to one of four communities. We randomly assigned each item to be liked by one of the communities (high utility) and disliked by the other three communities (low utility). From these communities and their associated utility functions, we generate the preferences of each user (without noise to make a pure synthetic test data set). Our approach is able to converge to exactly four communities with the correct memberships (as shown in Figure 3.3), demonstrating that our algorithm is effective in recovering latent community structure present in data. As observed in Table 3.1, the accuracy and the time consumed by the proposed approach is also improved compared to Full-GP.

Sushi Dataset: Here we describe the results on the *sushi* dataset [43]. It is a dataset of preferences of people about 10 types of sushis which leads to 45 preferences (pairs of sushis) per user. Each user and item (sushi) is specified by a set of features and where we have categorical features, they are converted to binary ones. Moreover, similar to the collaborative filtering setting, we included the IDs of the users and items as well. We discovered 8 communities in this dataset (as shown in Figure 3.3 while performing as accurately as Full-GP with two orders of magnitude less running time as shown in Table 3.1.

Figure 3.2: The distribution of preferred cars in each community for the AMT car dataset. Each x-axis position represents a different car and the y-axis the normalized frequency with which that car was preferred to another by a user in the community. Each community is distinct and differs in at least one car attribute.

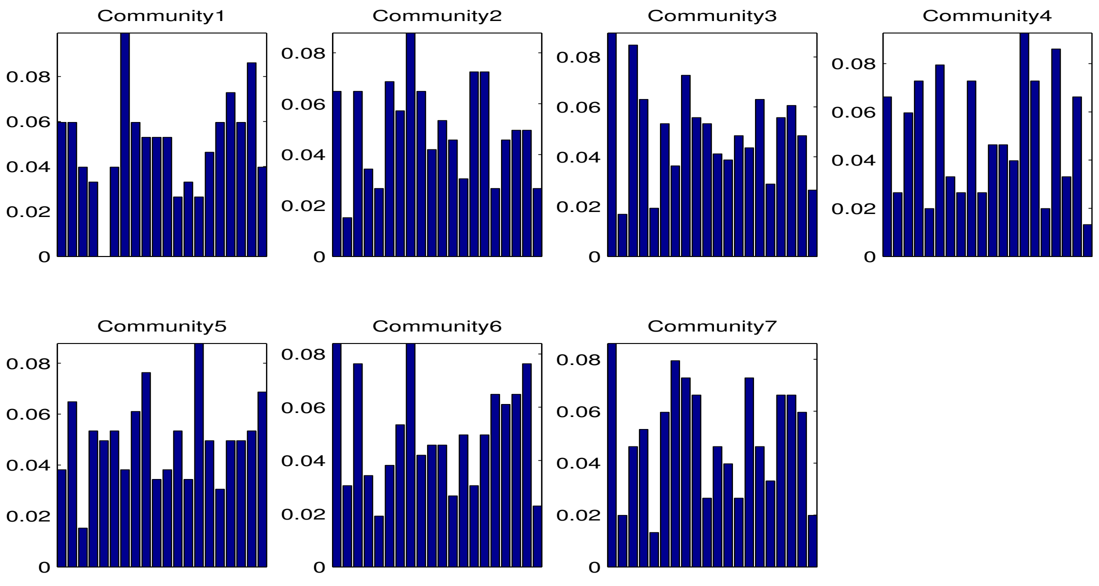
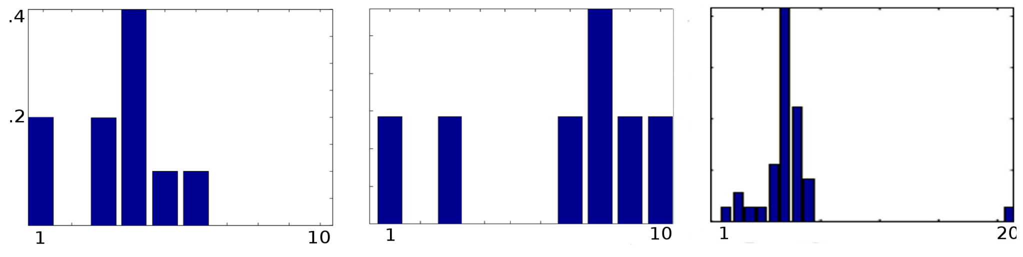


Table 3.2: Most likely attributes selected in each category by communities discovered in the AMT Car dataset. As observed in Figure 3.2, some communities are very different while others are similar. For example, community 3 and 7 are only different in the body type they prefer, but are both quite different from community 4.

	SUV, Sedan	**Automatic, Manual**	**Engine Capacity**	**Hybrid, Non-Hybrid**
1	SUV	Automatic	3.5L	Non-Hybrid
2	Sedan	Manual	2.5L	Non-Hybrid
3	Sedan	Automatic	2.5L	Non-Hybrid
4	SUV	Manual	4.5L	Non-Hybrid
5	Sedan	Manual	2.5L	Hybrid
6	SUV	Automatic	4.5L	Non-Hybrid
7	SUV	Automatic	2.5L	Non-Hybrid

Figure 3.3: Distribution of communities in the datasets . The x-axis is the number of communities; the y-axis is the posterior probability at the last sampling iteration. These values are obtained at iteration 15 of Synthetic and Sushi and iteration 36 of AMT Car.



(a) Synthetic (b) Sushi (c) AMT Car

Car Preference Dataset using Amazon Mechanical Turk[4]: Amazon Mechanical Turk[5] (AMT) provides an excellent crowdsourcing opportunity for performing online experiments and user studies. Since the number of publicly available preference learning datasets are limited, we set up an experiment in AMT to collect real pair-wise preferences over users. In this experiment users are presented with a choice to prefer a car over another based on their attributes. The car attributes used are:

- Body type: Sedan, SUV
- Engine capacity: 2.5L, 3.5L, 4.5L, etc.
- Transmission: Manual, Automatic
- Fuel consumed: Hybrid, Non-Hybrid

The set is then split into two sets with 60% of all the preferences kept for inference and the rest for testing. The dataset is collected so that 10 unique cars (items) are considered and users are required to answer all 45 possible pair-wise preferences. We targeted US users mainly to have a localized and consequently more meaningful preference dataset. For each user, a set of attributes in terms of general questions (age range, education level, residential region and gender) is collected. Every categorical attribute is converted to binary ones. We collected these preferences from 60 unique users.

As observed in Table 3.1, the proposed approach is as accurate and faster than Full-GP. Through learning the communities, we can also analyze the most frequently chosen attributes selected by a community as shown in Table 3.2, where inference converged to 7 communities.

[4]http://users.cecs.anu.edu.au/~u4940058/CarPreferences.html
[5]http://mturk.com

3.4 Related Work

Probabilistic models for utility functions in preference learning and elicitation have previously been proposed in the machine learning community [21, 38]. Extensions to non-parametric models have also been developed. In particular, [25] proposed a preference learning framework based on Gaussian processes and [27] used this model for active learning with discrete choice data. Multi-user GP-based preference models have been given by [11] and [14]. However, none of these methods directly address the efficiency problem nor discovered any community structure.

Stochastic blocked models [1, 65] are another class of related approaches that model the dependencies between users and infer the graphical structure under which users interact. Such models implicitly find the communities users belong to, even though not directly applied to preference learning in the setting we discussed here.

Furthermore, [73] proposed the infinite mixture of Gaussian processes for regression and later extended in [53] that is similar to the mixture of Gaussian processes detailed here. Our work however further extends the mixture of GPs to social preference learning.

3.5 Conclusion

Considering user populations often comprises of communities of shared preferences, we modeled user utilities as an infinite Dirichlet Process (DP) mixture of communities. The resulting inference algorithm scales linearly in the number of users unlike previous Gaussian Process preference learning approaches that scaled cubicly in the number of users. We evaluated our approach on a variety of preference data sources including Amazon Mechanical Turk showing that our method is more scalable and as accurate as previous work with only a small number of inferred communities, validating our community-based modeling approach. The posterior of the utilities learned can be used in a decision theoretic framework for applications such as recommendation. The framework discussed in this chapter, provides an efficient approach for learning uncertain utilities from preferences which can further be used in EEU framework for decision making.

Since the number of users in a single community in our model can grow arbitrarily large, the inference in each component of this mixture model can become difficult. One solution is to use a hierarchical representation that may better reflect the reality of users' clusters in communities albeit at the cost of more complex modeling. Another way to avoid this issue is to use a sparse representation of the GPs model of user utilities in each community which can further improve efficiency. In the subsequent chapter we look into using a decision theoretic framework to sparsify GPs for preference learning. This sparsification approach uses a subset of users and items as determined by the expected loss incurred from excluding them in the posterior distribution over utilities.

Chapter 4

Decision-theoretic Sparsification for Gaussian Process Preference Learning

In the previous chapter, we detailed an approach for efficient learning of the distribution of utilities using the structure of the problem represented in the communities. In this chapter we discuss another approach motivated by decision theory to improve scalability considered in the approximate posterior GP model of utility.

Scalability issues in Gaussian process are not exclusive to preference learning and they are common in other settings such as regression and classification. It is customary in Gaussian processes to adopt *sparsification* approaches, where a subset of training examples is selected as inducing points, considerably reducing the time complexity of posterior approximation and prediction [44, 81, 82]. A popular approach to GP sparsification is the Informative Vector Machine [48], where inducing points are selected according to an information-theoretic criterion. A key characteristic of the IVM is that it can be embedded naturally in sequential algorithms such as assumed density filtering (ADF) or EP. These algorithms provide efficient computation of the quantities of interest (i.e., posterior variances) to be used by the IVM's sparsification criterion.

Nevertheless, the IVM's purely entropic sparsification criterion fails at addressing the varying loss functions that may be of interest to the final decision-theoretic task — especially those tasks that naturally arise in preference learning.

For example, we might be interested in

1. optimizing the utility of the best recommendation,
2. giving a ranking of all items (or a subset), or
3. correctly classifying all pairwise preferences.

In each case we seek to optimize a loss for a different decision-theoretic task and when we need to approximate in a Bayesian setting, it is important that our approximation is loss-calibrated [47], i.e. the inference is performed with the task-specific loss taken into account in approximation. We note that the uncertainty reduction principle inherent to the IVM approximation is not loss-calibrated for all tasks (1)–(3).

In this chapter, we continue to bridge the gap between decision theory and approximate Bayesian inference [47] in a direction that leverages the efficiency of the IVM approach for GP sparsification, while overcoming its loss-insensitive approximation. We show that the IVM's differential entropy criterion, a value of information criterion, and an upper confidence bound [84] criterion can *all* be recovered in our framework by specifying the appropriate loss.

An additional important aspect of the preference learning problem that distinguishes it from standard machine learning settings is that the complexity of making predictions does not directly depend upon the number of observations (i.e. preference relations), but rather the number of users and items. Our method takes this into consideration and adopts an item-driven sparsification strategy that retains the items that best encode the users' preferences. Our experiments show that this is an effective way of reducing the complexity of inference in preference learning with GPs while addressing the objective function of interest directly. We refer to our generic method as the *Valuable Vector Machine* (VVM) since it incorporates the loss function directly into its sparsification mechanism.

The rest of this chapter is organized as follows: we propose our VVM sparsification framework in Section 4.1 built upon the GPs for preference learning framework introduced in Section 3.1. The empirical evaluation is presented in Section 4.2. We differentiate our approach from related work in Section 4.3 and conclude in Section 4.4.

4.1 Decision-theoretic Sparsification

To recap Section 3.1, our multi-user preference learning objective is to approximate a posterior $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ over latent utilities $\boldsymbol{u} = [u_1^{\omega_1}, u_2^{\omega_1}, \ldots, u_m^{\omega_n}]^T$ for users $\omega \in U$ and items $\mathbf{a}_i \in X$. In the previous section, we showed how to learn $q(\boldsymbol{u})$ from preference data by EP; in this section due to computational considerations, we wish to sparsify this Gaussian posterior in a loss-calibrated manner. We note that in the special case of GP-based preference learning, there are at least two different ways one might approach sparsification: observation-driven sparsification and item-driven sparsification.

4.1.1 Observation-driven Sparsification

In this approach, we incrementally select a subset of observations (in this case preferences) in order to approximate the posterior $q(\boldsymbol{u})$. More formally, recall that $\mathcal{D}^\omega = \{\mathbf{a}_i^\omega \succ \mathbf{a}_j^\omega\}$ and let $\mathcal{D}' \subseteq \mathcal{D}$ be a subset of selected preferences. Observation-driven sparsification simply chooses the data subset $\mathcal{D}'$ according to some criterion to obtain a posterior approximation $q_{\mathcal{D}'}(\boldsymbol{u}) \approx p(\boldsymbol{u}|\mathcal{D}')$ (e.g., via EP as outlined in the last section).

Table 4.1: Loss and corresponding risk to minimize for VVM variants. Let $q(\boldsymbol{u}) := q_{S''}(\boldsymbol{u})$ and $a \in \{\mathbf{a}_i^{\omega}\}$.

Algorithm	IVM (item)	VVM-VOI	VVM-UCB
Loss Type	Logarithmic loss	Regret	Exponential Loss
$\mathcal{L}(\boldsymbol{u}, \mathbf{a}_i, \omega)$	$-\log(q(f_i^{\omega}))$	$-\mathbb{I}[u_i^{\omega} > f^{\omega,*} u](u_i^{\omega} - u^{\omega,*})$	$-\exp(\beta u_i^{\omega}), \beta > 0$
$Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega)$	$H(q(u_i^{\omega}))$	$\sigma(\omega, \mathbf{a}_i)\,[c\Phi(c) + \mathcal{N}(c)]$	$1 + \beta\mu(\omega, \mathbf{a}_i) + \frac{\beta^2}{2}\bar{\sigma}^2(\omega, \mathbf{a}_i)$
Selection Time	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

As a concrete example, the original *Informative Vector Machine* [48] initializes $\mathcal{D}'$ to a small random subset and then incrementally builds $\mathcal{D}' := \mathcal{D}' \cup \{d^*\}$ for the d^* that maximizes information gain

$$d^* = \operatorname*{arg\,max}_{d \in \mathcal{D} \setminus \mathcal{D}'} H(q_{\mathcal{D}' \cup \{d\}}(\boldsymbol{u})) - H(q_{\mathcal{D}'}(\boldsymbol{u})), \tag{4.1}$$

where $q_{\mathcal{D}'}(\boldsymbol{u}) \approx p(\boldsymbol{u}|\mathcal{D}')$, $q_{\mathcal{D}' \cup \{d\}}(\boldsymbol{u}) \approx p(\boldsymbol{u}|\mathcal{D}' \cup \{d\})$ and H is the entropy. This repeats until the desired level of observation sparsity has been reached. Since $\mathcal{D}'$ is fixed at each iteration and thus $H(q_{\mathcal{D}'}(\boldsymbol{u}))$ is a constant, each incremental selection in the IVM is equivalent to choosing the d^* that maximizes entropy, i.e., $d^* = \arg\max_{d \in \mathcal{D} \setminus \mathcal{D}'} H(q_{\mathcal{D}' \cup \{d\}}(\boldsymbol{u}))$.

4.1.2 Item-driven Sparsification: Valuable Vector Machine

Inclusion of a preference observation entails a 2-dimensional update to our GP posterior; however, since preferences may overlap, there is not a direct relationship between the number of included preferences and the dimensionality of the posterior and hence the computational complexity of prediction described in section 3.2.3. A more direct way to control the sparsity level of our Gaussian posterior is to simply retain the *items* for users (equivalently dimensions f_i^{ω} of our Gaussian posterior) that minimizes some criterion.

This item-driven approach underlies the *Valuable Vector Machine* (VVM) that we propose for different decision-theoretic settings. First we introduce some notation. Let $S' \subseteq S = \{(\mathbf{a}_i, \omega)\}$ be a selected subset of user-item pairs corresponding to latent utility dimensions f_i^{ω} of $\boldsymbol{u}$ with cardinality $|S'|$. Let $q(\boldsymbol{u}_{[S']}) = \mathcal{N}(\boldsymbol{u}_{[S']}; \boldsymbol{\mu}_{[S']}, \boldsymbol{\Sigma}_{[S',S']})$ where $\boldsymbol{u}_{[S']}$ and $\boldsymbol{\mu}_{[S']}$ respectively represent the subvectors of $\boldsymbol{u}$ and $\boldsymbol{\mu}$ for selected dimensions S' (i.e., selected users and items in set S') and $\boldsymbol{\Sigma}_{[S',S']}$ the corresponding submatrix of $\boldsymbol{\Sigma}$. Motivated by the observation-driven IVM, after running EP, let us incrementally select dimensions $s^* \in S$ of our Gaussian posterior to retain so that initializing $S' = \emptyset$, at each iteration we update $S' := S' \cup \{s^*\}$ to obtain an improved posterior $q_{S'}(\boldsymbol{u})$ until some dimensionality limit has been reached.

In decision theory, our objective is to select an action $a^* \in A$ from a possible space of actions A so as to minimize the expectation of some loss $\mathcal{L}(a)$ w.r.t. uncertainty (in this case utility uncertainty over $\boldsymbol{u}$), i.e. $a^* = \arg\min_a \mathbb{E}_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{u}, a)$. Our specific task at each iteration of the VVM is to propose an item-user dimension $\mathbf{a}_i^{\omega}$ for inclusion in the posterior — hence the action space $A = \{\mathbf{a}_i\}$ — and to select the item s^* that minimizes expected loss (risk)

$$s^* = \underset{(\mathbf{a}_i,\omega)\in S\setminus S'}{\arg\min}\ Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega);$$
$$\text{where } Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega) := \mathbb{E}_{\boldsymbol{u}\sim q_{S''}}\left[\mathcal{L}(\boldsymbol{u}, \mathbf{a}_i, \omega)\right], \tag{4.2}$$

$S'' = S' \cup \{\mathbf{a}_i^{\omega}\}$. In the following, we will detail choices of loss functions and their respective $Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega)$ yielding the VVM variants as summarized in Table 4.1 and its corresponding method in Algorithm 3.1.

In each iteration, VVM selects the action (i.e. item) that minimizes the expected loss for each user until desired predefined dimensionality is reached. Our experiments with a variable number of items per user led to worse performance since it often overemphasizes item selection for the noisiest users. Hence, we found that a constant number of items enforces fairness of GP modeling effort per user.

It should also be noted that the greedy selection here is fairly general and in special cases such as submodular losses, one can prove further convergence guarantees [46].

Algorithm 4.1 Valuable Vector Machine

```
input: $X, U, \mathcal{D}, r$ // r is the number of items selected for each user
while not converged do
  for $\mathbf{a}_i^{\omega} \succ \mathbf{a}_j^{\omega} \in \mathcal{D}$ do
    1. Update the cavity distribution $\boldsymbol{\mu}_{\setminus\omega,[i,j]}$, $\boldsymbol{\Sigma}_{\setminus\omega,[i,j]}$ from Equation 3.11 and 3.12.
    2. Update the unnormalized marginal posterior $\hat{\boldsymbol{\mu}}$,$\hat{\boldsymbol{\Sigma}}$ from Equation 3.15 and 3.16.
    3. Update the local factor approximation $\tilde{\boldsymbol{\mu}}$, $\tilde{\boldsymbol{\Sigma}}$ from Equation 3.17.
    5. Update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from Equation 3.7.
  end for
end while
for each $\omega \in U$ // Selection of best items for each user do
  $S'^{\omega} = \{\}$ // The user ω's subset
  while $|S'^{\omega}| < r$ do
    $\mathbf{a}_i^* = \arg\min_{\mathbf{a}_i \in X, \mathbf{a}_i \notin S'^{\omega}} Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega)$// Table 4.1
    $S'^{\omega} = S'^{\omega} \cup \{\mathbf{a}_i^*\}$
  end while
  $S = S \cup S'^{\omega}$
end forreturn $\boldsymbol{\mu}_{[S]}, \boldsymbol{\Sigma}_{[S,S]}$ // Subset of posterior parameters
```

4.1.3 Loss Functions and Risk

4.1.3.1 Log loss and IVM

Log-loss is appropriate when we want to maximize the log posterior over all preferences. Here we see that we can actually recover an item-based variant of the IVM when using log-loss. Specifically, letting $q(f_i^\omega)$ refer to the marginal of $q(\boldsymbol{u})$ over f_i^ω then

$$\begin{aligned} Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega) &= \mathbb{E}\mathcal{L}(\boldsymbol{u}_i^\omega \int_{-\infty}^{\infty} -q_{S''}(u_i^\omega)[\log q_{S''}(u_i^\omega)]du_i^\omega \\ &= H(q_{S''}(u_i^\omega)), \end{aligned} \tag{4.3}$$

which corresponds to the entropic criterion used by the IVM. Recall that the second entropy term in the standard IVM information gain calculation is constant and can be omitted as noted for (4.1).

4.1.3.2 Valuable Vector Machine – Value Of Information

In the case that our end objective is to predict or recommend the best item $\mathbf{a}_i$ for user ω, the loss we might consider minimizing is the *regret*, $\mathbb{I}[f_i^\omega - f^{\omega,*} > 0](f_i^\omega - f^{\omega,*})$ where we could define $f^{\omega,*} = \arg\max_i f_i^\omega$; in words, we want to minimize how much utility we lose for recommending a suboptimal item. In expectation, we might simply fix $f^{\omega,*} = \max_i \mathbb{E}_{q_{S''}}[f_i^\omega]$ (the best current item in expectation) where expected loss minimization leads us to the following risk:

$$\begin{aligned} Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega) &= \int_{-\infty}^{\infty} \mathbb{I}[u_i^\omega > u^{\omega,*}](u_i^\omega - u^{\omega,*})q_{S''}(u_i^\omega)du_i^\omega \\ &= \underbrace{\sigma(\omega, \mathbf{a})\left[c\Phi(c) + \mathcal{N}(c)\right]}_{VOI} \end{aligned} \tag{4.4}$$

where $q_{S''}(u_i^\omega) = \mathcal{N}(u_i^\omega; \mu(\omega, \mathbf{a}), \sigma^2(\omega, \mathbf{a}))$ and $c = \frac{\mu(\omega,\mathbf{a}) - u^{\hat{\omega}}}{\sigma(\omega,\mathbf{a})}$ [83]. This is precisely the statement of *Value of Information* (VOI) [40] under a Gaussian assumption of uncertainty — quite simply, the more probability mass an item utility has in its tail above the best item in expectation, the higher its chance of being the best item — hence the higher VOI associated with selecting this item in S''. The minimization of risk in this case is equivalent to maximizing the VOI.

4.1.3.3 Valuable Vector Machine – Upper Confidence Bound

It is well-known that a concave or convex valuation of underlying utility respectively encourages risk-averse or risk-seeking behavior w.r.t. utility uncertainty. Risk-seeking behavior from a convex utility function will encourage including "potentially optimal" items according to how uncertain we are regarding their utility function value. A natural convex utility transformation is $\exp(\beta u_{\mathbf{a}}^\omega)$, which leads to the following

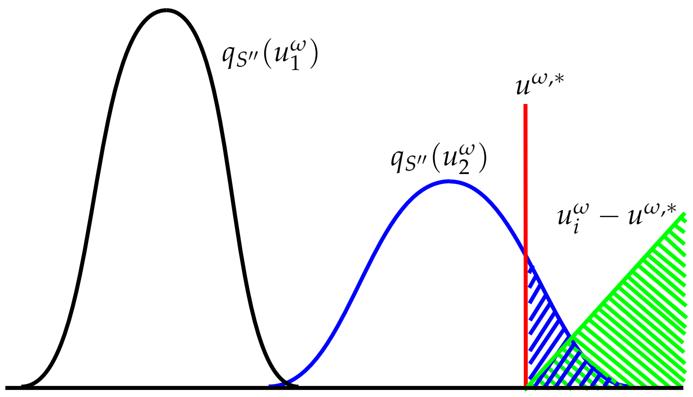


Figure 4.1: Illustration of Value of Information: it is the product of the shaded area under the normal curve when the utility is higher than the optimal value and the linear function of their difference. This value corresponds to the expectation of the difference of the utility of the item and the optimal under the shaded mass. As it is observed, u_1^ω has negligible mass above $u^{\omega,*}$ point, therefore the item corresponding to u_2^ω is selected.

risk

$$\begin{aligned} Risk_{\mathcal{L}}(S', \mathbf{a}_i, \omega) &= -\int_{-\infty}^{\infty} q_{S''}(u_i^\omega) \exp(\beta u_i^\omega) du_i^\omega \\ &= -\int_{-\infty}^{\infty} q_{S''}(u_i^\omega) \left(1 + \beta u_i^\omega + \frac{\beta^2}{2} u_i^{\omega 2} + \dots\right) du_i^\omega \\ &\approx 1 + \beta \cdot UCB(\omega, \mathbf{a}_i) \end{aligned}$$

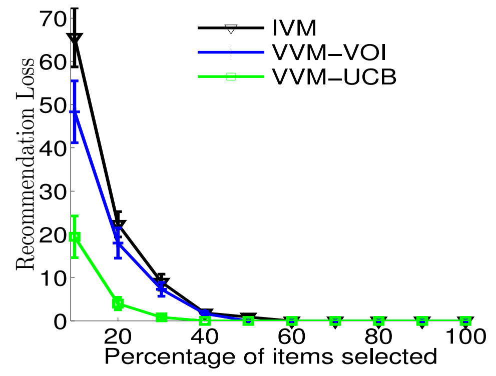
where

$$UCB(\omega, \mathbf{a}_i) \quad = \quad \mu(\omega, \mathbf{a}_i) + \frac{\beta}{2} \bar{\sigma}^2(\omega, \mathbf{a}_i). \tag{4.5}$$

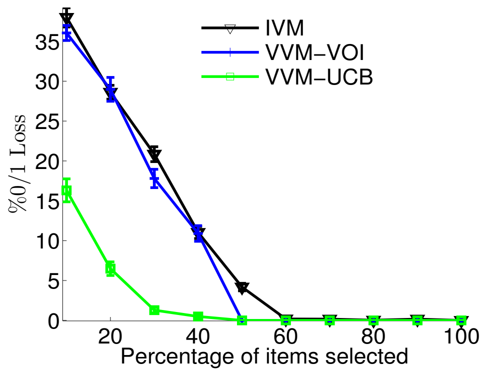
and $\bar{\sigma}^2(\omega, \mathbf{a}_i) = \mathbb{E}_{q_{S''}}[u_i^{\omega 2}]$. Here, we first replaced $\exp(\beta u_i^\omega)$ with its Taylor expansion and approximated it by truncating third-order terms and above. When doing this, we see that the dimension $\mathbf{a}_i^u$ selected by the VVM will be the one with the greatest *Upper Confidence Bound* (UCB) [3] used in bandit problems, where larger $\beta > 0$ encourages more risk-seeking behavior. As seen, maximizing UCB is equal to minimizing the risk in our general framework.

4.2 Empirical Evaluation

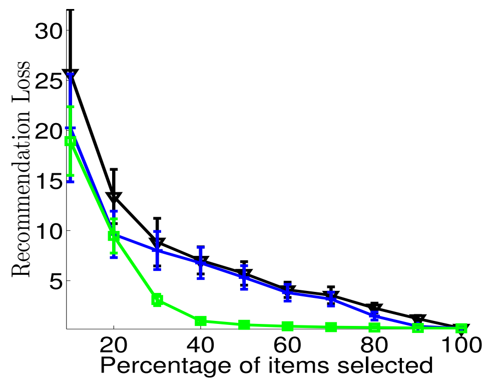
In this section we evaluate the performance of our algorithms (VVM-VOI and VVM-UCB) compared to the IVM and the full GP, i.e. a GP-preference model that does not use sparsification, in terms of two losses: the $0/1$ loss and recommendation loss. The *0/1 loss* is the percentage of incorrectly predicted preferences and the *recommendation loss* is the proportion of items that are incorrectly predicted as the best for recom-



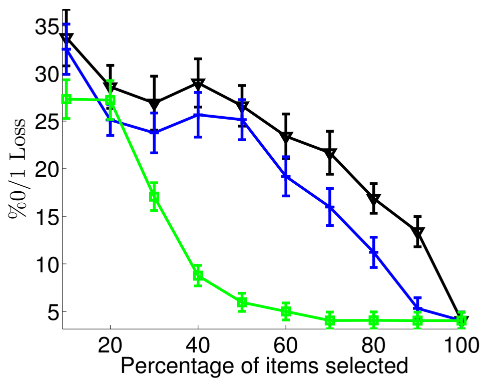
(a) Synthetic dataset



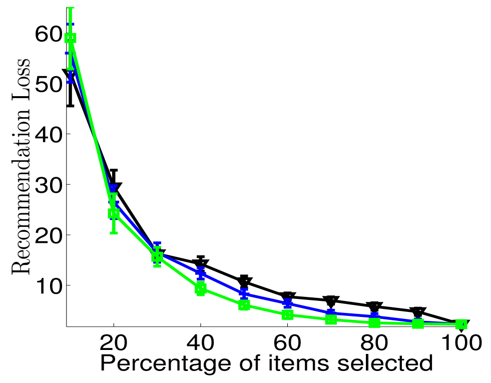
(b) Synthetic dataset



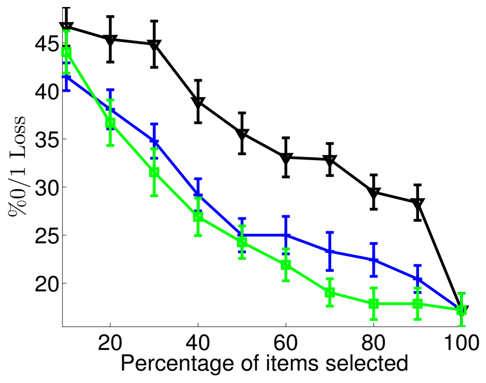
(c) Facebook dataset



(d) Facebook dataset



(e) AMT Car dataset



(f) AMT Car dataset

Figure 4.2: Performance of the sparsification methods in terms of the *recommendation loss* (the proportion of items that are incorrectly predicted as the best item for recommendation) in the first column and the 0/1 *loss* (percentage of wrongly predicted preferences) in the second, as a function of the proportion of items selected for sparsification. The larger the number of items the lower the level of sparsification and the closer the algorithms are to the Full-GP method.

mendation. In other words, if the set of items that a user considers to be the best (as induced by her preferences) is denoted by T and the predicted set of best items is T^*, the recommendation loss is $\frac{|\bar{T}|}{m} \times 100$, where m is the number of items and $\bar{T} = \{\mathbf{a} \in T^* | \mathbf{a} \notin T\}$. We report the results as a function of the proportion of items selected for sparsification.

Our experimental rationale is to exhibit how different risk-sensitive sparsifications perform across two different important losses related to preference learning compared to IVM. As such, we use IVM in its original form that works with ADF since it was argued by [80] that it performs better than running full EP, which we observed as well. Hence we chose the IVM variant that offered best performance and compared it against VVM.

We consider three datasets: a synthetic dataset and two real-world datasets. The synthetic experiment assesses the effectiveness of our approach in a controlled setting and the real-world datasets include users' preferences over cars that we have collected using Amazon Mechanical Turk and a Facebook dataset that we have obtained via an in-house application that collects user preferences over web links.

In all these datasets we are given a set of users and items and their corresponding features along with each user's preferences over item pairs. For each user and item we augment their feature vectors with their ID index (this is a common practice in collaborative filtering) and transform their categorical features into binary variables. We split each user's set of preferences into 60% for training and 40% for testing.

We use the squared exponential covariance kernel with automatic relevance determination (ARD) (see [75], Page 106) for both users and items and optimize the hyper-parameters by maximizing the marginal likelihood under the EP approximation as detailed in section 3.1.2. we have set $\alpha = 3$ (see Equation (3.2)) and $\beta = 1$ (see Equation (4.5)) for all experiments on all datasets.

4.2.1 Datasets

Synthetic Dataset: In this experiment we created a synthetic dataset where the utility function value for each item is known beforehand and is subsequently used to generate users' preferences. A set of hypothetical users and items are created and identified by their IDs. For each user, items are randomly split into two sets to indicate the ones that are liked (with a constant utility value of 10) and disliked (with a constant utility value of 5). From these utility functions we generate full sets of preferences for 10 items and 50 users. Consequently, for each user, 5 items have higher utility value and are naturally preferred to the other half.

Facebook Data: This dataset has been created using a Facebook App that recommends web links to users every day. The users may give their feedback on the links indicating whether they liked/disliked them. At its peak usage, 111 users had elected to install the Facebook app developed for this project. also collected user information consisting of ID, age and gender and the link features including count of link's total "likes", count of link's "shares" and count of total link comments. The Facebook App recommended three links per day to avoid position bias and information overload.

The preference set is built such that the links that are liked are considered *preferred* to the ones disliked in the batch of three recommended each day. We used 20% of users with the highest number of preferences over 50 links commonly recommended to all users.

Car Preference Dataset using Amazon Mechanical Turk: We used the car preference dataset explained in the previous chapter.

4.2.2 Results

We evaluate our algorithms in a cross-validation setting using 60% of preferences for training and 40% for testing and repeated each experiment 40 times. Results are averaged over the number of test users. We analyze the performance of the algorithms as a function of the level of sparsification, as given by the percentage of items selected for inference. The larger the percentage of items selected, the smaller level of sparsification and the closer the algorithms are to the Full-GP method. The performances of the different algorithms on all datasets using the recommendation loss and the 0/1 loss are shown in Figure 4.2.

Figures 4.2(a) and 4.2(b) show the results on the synthetic dataset. Because of the clear distinction between the items that are preferred for each user, all algorithms perform very well when using at least 40% of the items. While IVM and VVM-VOI have very similar performance, VVM-UCB's performance is outstanding, requiring only a very small number of items to achieve perfect prediction.

As seen in Figures 4.2(c) and 4.2(d), on the Facebook dataset both VVM-VOI and VVM-UCB outperform (or have equal performance to) IVM when using at least 30% of the items.

It is interesting to note here that the risk-seeking behavior of VVM-UCB leads to a better approximation of the Full-GP which is particularly visible in the Facebook dataset where the number of items are larger. We conjecture that the excellent performance of VVM-UVB with this larger number of items is because it manages to quickly find and refine the set of highest value items, more effectively than even VVM-VOI. This simultaneously lowers recommendation loss by finding a near-optimal item and 0/1 loss since the best items can then be identified with certainty in most pairwise comparisons.

Figures 4.2(e) and 4.2(f) show the results on the AMT Car dataset. In this dataset, where true preferences have been collected, VVM-VOI and VVM-UCB consistently outperform IVM. Similar to the Facebook results, we conjecture that VVM-VOI's and VVM-UCB's better performance than the IVM (most notably on 0/1 loss where all preferences matter) stems from the fact that they both select the potentially best items first and this helps identify the dominant item in all pairwise preferences. However it seems that identifying the single best item among the potentially best items is difficult in this particular dataset, requiring a large proportion of data to identify the best item with high accuracy.

It is interesting to mention that while VVM-VOI and VVM-UCB outperform IVM in most cases when using the recommendation loss, a similar trend is seen when

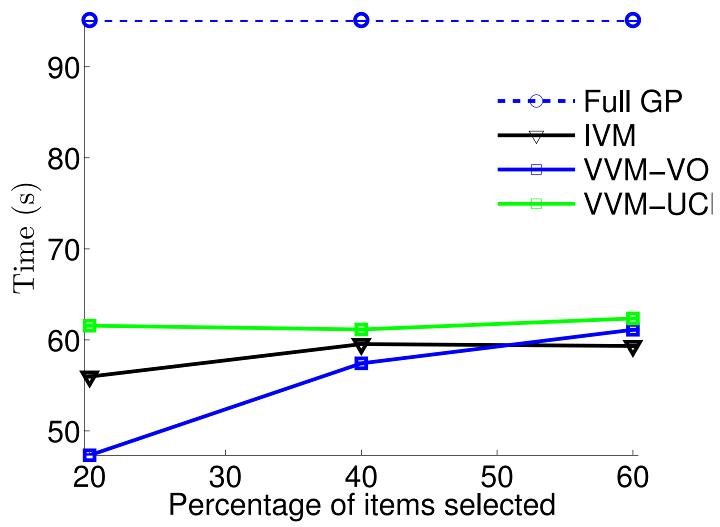


Figure 4.3: Average prediction time for inference with 200 users and 10 items. The number calculated as the time consumed to make a series of predictions on the preferences of the test set.

using the 0/1 loss. Although this result may look unexpected, it is important to emphasize that neither the VVM or the IVM are designed to optimize the 0/1 loss. In fact, the risk-seeking nature of the VVM-UCB loss, as a consequence of the exponential transformation of the utility functions, may be better aligned with the 0/1 loss than the entropic criterion used by the IVM.

Another issue worth mentioning is the computational cost of running the different approximation algorithms. As a reference of the time spent by our algorithms compared to the Full-GP (where no sparsification is done), Figure 4.3 shows the prediction time for an indicative experiment. We see that – while IVM and VVM-UCB may enjoy very similar prediction time and similar structure in the posterior – sparsification improves prediction time significantly and that all approximation algorithms have roughly the same computational cost. Small variations as that observed when using 60% of the items can be explained by the different sparsity properties of the posterior covariance obtained when selecting a distinct subset of items.

4.3 Related Work

In standard machine learning settings, low-rank approximations to the Gram matrix are commonly used by practitioners and researchers (see e.g. Chapter 8 of [75]) to deal with large datasets. A unifying framework in which most of these approximations can be formulated has been given by [71]. This framework includes the fully independent training conditional (FITC) approximation, which makes better use of all the data and can be combined with our approach to approximate the covariance at a higher cost. The work proposed in [82] considers sparsification approaches where the inducing points are latent variables their values are optimized within a consistent probabilistic latent variable model. However, none of these algorithms addresses the sparsification problem from a decision-theoretic perspective.

Our approach is analogous to the IVM in that we borrow ideas from active learning in order to carry out sparsification during approximate inference. For example, upper confidence bounds (UCB) are used in [84] for GP optimization within the bandit setting. We note that, unlike this latter experimental design scenario, in our sparsification framework we see the data beforehand and decide to include it in our approximation afterwards.

An information theoretic active learning algorithm for classification and preference learning is proposed in [61]. In the preference learning case, this method exploits the reduction of the preference learning problem to a classification setting [62]. This work is complementary to ours in that we can use it along with the FITC approximation in order to devise more effective decision-theoretic sparsification methods for multi-user preference learning. We leave the study of such an approach for future work.

The most relevant work to ours has been proposed in [47] where the use of loss functions in Bayesian methods is considered by formulating an EM algorithm that alternates between variational inference and risk minimization. We take the idea of bridging the gap between decision theory and approximate Bayesian inference [62] in a direction that leverages the efficiency of the IVM approach for GP sparsification, while overcoming its loss-insensitive approximation.

4.4 Conclusion

We proposed a decision-theoretic sparsification method for Gaussian process preference learning. We referred to our method as the valuable vector machine (VVM) to emphasize the importance of considering a loss-sensitive sparsification approach. We showed that the IVM's differential entropy criterion, a value of information criterion, and an upper confidence bound (UCB) criterion can be recovered in a generalized decision-theoretic framework by specifying the appropriate loss. Our approach provides a basis for inference in Gaussian process in the context of loss-sensitive sparsification. Empirical evaluations further showed, that this is an effective approach in approximating the posterior. The approximate posterior obtained from VVM can be used for modeling the users in each community in Chapter 3 to increase performance further.

VVM uses a simple greedy approach for approximation that even though empirically shown to perform well needs further theoretical guarantees. If the regret function considered for item selection is submodular, then we have guarantees on the approximation bounds and convergence for free. In addition, VVM is a post inference procedure in the sense that the full posterior has to be found before sparsification. An ideal extension of our approach is to use the decision theoretic framework of VVM within EP so that the full posterior is not required to be learned.

Once the approximate belief over the utility function is learned using VVM, we can use EEU framework for decision making. For each user, the expectation of each action (selection of an item) with respect to this belief is the expected expected utility. The item with highest expected expected value is the optimal item for recommendation.

In the EEU framework, we need to be able to find the optimal action by computing the expectations efficiently. These expectations are taken with respect to the distribution of the utility that is learned using one of the approaches discussed in this or previous chapter. In the subsequent chapter, we will present an efficient Monte Carlo method that computes the expectations using samples of a distribution (such as the belief over utilities). This efficient sampling approach uses a fraction of the samples that conventional sampling algorithms require to compute expected utilities for finding the optimal action.

Chapter 5

Loss-calibrated Monte Carlo Action Selection

So far we have covered two approaches to modeling and learning the distribution of the utilities. We have discussed using a structure in the data, i.e. user communities, and sparsification of the posterior for efficient learning of this belief over the utility function. In the EEU framework, we need to be able to compute the expectations with respect to this distribution to find the optimal action. In this chapter, we concentrate on computing this expectation efficiently.

In both conventional Bayesian decision theory and the expected expected utility framework, an important aspect of the selection of the optimal action is computing the expectations. Having known the distribution of the state or utility, these expectations can be computed by sampling. In this chapter, we focus on sampling for efficient selection of the optimal action. In particular, we focus on the expectation in either of the following cases:

1. **State uncertainty**: in conventional decision theory, we compute the expected utility with respect to the belief over the state value using the given utility function for each action. Hence, when the distribution of the state is given and we are computing the expected utility of action a (as in Equation (1.1)):

$$\mathrm{EU}_u(a) = \mathbb{E}[u(\boldsymbol{\theta}, a)] = \int u(\boldsymbol{\theta}, a) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}, \tag{5.1}$$

2. **Utility uncertainty:** in EEU framework where we consider the uncertainty in the utility and we compute the expectation with respect to the belief over the utility function. Thus, when a distribution of the utility function is given, e.g. it is found in a model similar to the ones discussed in Chapter 3 and 4, the expectation is computed with respect to the uncertain functions drawn from the posterior of the utilities in the expected expected utility framework (as in Equation (1.3)):

$$\mathbb{E}[\mathrm{EU}_u(a)] = \int \mathrm{EU}_u(a) p(u|\mathcal{D}) du.$$

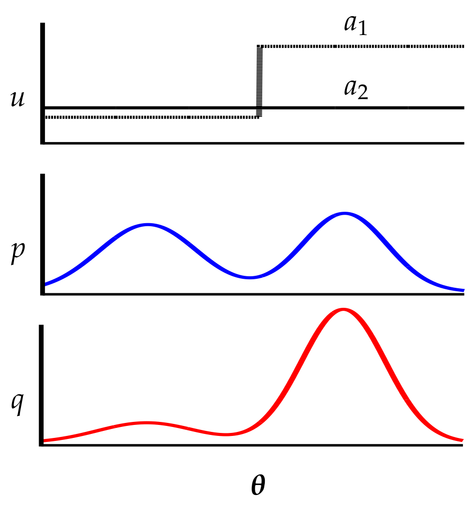


Figure 5.1: Motivation for loss-calibration in Monte Carlo action selection. (Top) Utility $u(\boldsymbol{\theta})$ for actions a_1 and a_2 as a function of state $\boldsymbol{\theta}$. (Middle) A belief state distribution $p(\theta)$ for which the optimal action $\arg\max_{a\in\{a_1,a_2\}} \mathbb{E}_p[u(\boldsymbol{\theta},a)]$ should be computed. (Bottom) A potential proposal distribution $q(\boldsymbol{\theta})$ for importance sampling to determine the optimal action to take in $p(\boldsymbol{\theta})$.

Without loss of generality, we consider the distribution of the state $p(\boldsymbol{\theta})$ or utility $p(u)$ regardless of whether it is a posterior or prior (it is conditioned on data or not). The second case is the same, except the samples are drawn from a function space such as samples from the Gaussian process posterior (e.g. [59, 91]). For notational convenience we denote $\mathrm{EU}_u(a)$ simply by $\mathrm{EU}(a)$.

In real-world settings such as robotics [87], the belief distribution $p(\boldsymbol{\theta})$ may be complex (e.g., highly multimodal) and/or high-dimensional thus prohibiting the application of analytical methods to evaluate the EU integral of (5.1). Practitioners often resort to the use of Monte Carlo methods to compute an approximate (but unbiased) expectation using n samples from $p(\boldsymbol{\theta})$:

$$\frac{1}{N}\sum_{i=1}^{N}[u(\boldsymbol{\theta}_i,a)], \quad \boldsymbol{\theta}_i \sim p. \tag{5.2}$$

Unfortunately, naïve application of Monte Carlo methods for optimal action selection often proves to be inefficient as we illustrate in Figure 5.1. At the top we show two utility functions for actions a_1 and a_2 as a function of univariate state θ on the x-axis. Below this, in blue, we show the known state belief distribution $p(\theta)$. Here, it turns out that $\mathrm{EU}(a_1) > \mathrm{EU}(a_2)$. Unfortunately, if we sample from $p(\theta)$ to compute a Monte Carlo expected utility for each of a_1 and a_2, we find ourselves sampling frequently in the region where $u(\boldsymbol{\theta},a_1)$ and $u(\boldsymbol{\theta},a_2)$ are very close, but where suboptimal a_2 is marginally better than a_1.

An intuitive remedy to this problem is provided by an importance sampling approach (see e.g. [33]) where we sample more heavily in regions as indicated by distribution $q(\theta)$ and then reweight the Monte Carlo expectation to provide an unbiased estimate of $\mathrm{EU}(a)$. Formally, the theory of importance sampling tells us that since

$$\mathrm{EU}(a) = \int \left[\frac{u(\boldsymbol{\theta}, a)p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right] q(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{5.3}$$

we can draw samples from q to compute an (unbiased) estimate of $\mathrm{EU}(a)$ as

$$\hat{\mathrm{EU}}_N(a) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{u(\boldsymbol{\theta}_i, a)p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}\right], \quad \boldsymbol{\theta}_i \sim q. \tag{5.4}$$

This leaves us with one key question to answer in this chapter: *How can we automatically derive a $q(\boldsymbol{\theta})$ to increase the probability that the optimal action a^* is selected for a finite set of n samples ?* Answering this question is important for real-time applications of Bayesian decision-theoretic approaches where efficiency and optimality are two key operating criteria.

To this end, in the subsequent section we derive a proposal distribution for a loss-calibrated Monte Carlo importance sampler that tightens a bound on the *regret*, i.e., the difference of the expected utility of the true optimal action and its estimate. We first establish an intuitive result that minimizing the probability of suboptimal action selection tightens this bound. For two actions, this bound is tight and minimizing this probability is in fact equivalent to maximizing the expected utility of the optimal action.

We evaluate our loss-calibrated Monte Carlo method in two domains. We first examine a synthetic plant control examples building on those of [47], who were also motivated by loss-calibration in Bayesian decision theory, albeit not in the case of Monte Carlo methods as we focus on in this work. We also demonstrate results in a Bayesian decision-theoretic robotics setting with uncertain localization motivated by the work of [87].

In both empirical settings and in states with up to 100 dimensions, we demonstrate that using our loss-calibrated Monte Carlo method find the optimal action with fewer samples than conventional loss-insensitive samplers. This suggests a new class of loss-calibrated Monte Carlo samplers for efficient online Bayesian decision-theoretic action selection.

5.1 Loss-calibrated Monte Carlo Importance Sampling

In many applications of decision theory, sampling is the most time-consuming step. Since we know these samples are ultimately used to estimate high-utility actions, we are interested in guiding the sampler to be more efficient for this task.

Here, we will pick a distribution q to draw samples from, which are in turn used to select the action that maximizes the EU. The estimated optimal action $\hat{a}_N$ defined as

$$\hat{a}_N = \arg\max_a \quad \hat{\text{EU}}_N(a). \tag{5.5}$$

Since the samples are drawn randomly from q, $\hat{a}_N$ is a random variable and so is its expected utility $\text{EU}(\hat{a}_N)$. As such, we use $\mathbb{E}, \mathbb{P}$ and $\mathbb{V}$ henceforth to denote the expectation, probability and variance operators. We emphasize that all random variables are determined by q.

In principle, we would like to select the distribution q to minimize regret, i.e. maximize the true EU of the estimated action $\hat{a}_N$. As this is challenging to do directly, proceed in three steps:

1. We establish a connection between regret and the probability of suboptimal action selection in Theorem 1.
2. Since calculating the probability of selecting the suboptimal action is intractable to be directly minimized, we derive an upper bound in Theorem 15, based on the variance of the difference of estimated utilities.
3. Theorem 17 shows how to calculate the distribution q to minimize this bound.

5.1.1 Minimizing regret

To find the optimal estimated action $\hat{a}_N$ with fewer samples, we wish to select q that minimizes the regret. Formally we define this as

$$\begin{aligned} \min_q \qquad & \ell(\hat{a}_N) \\ \text{where} \qquad & \ell(\hat{a}_N) = \mathbb{E}\left[\text{EU}(a^*) - \text{EU}(\hat{a}_N)\right]. \end{aligned} \tag{5.6}$$

Direct minimization of Equation 5.6 is difficult, hence we bound it with the probability of selecting a suboptimal action instead. Tightening this bound with respect to q will lead to a practical strategy. It is detailed in the following theorem.

Theorem 1 (Regret bounds). *For the optimal action a^* and its estimate $\hat{a}_N$ the regret as defined in Equation 5.6, is bounded as*

$$\Delta\, \mathbb{P}\left[\hat{a}_N \neq a^*\right] \leq \ell(\hat{a}_N) \leq \Gamma\, \mathbb{P}\left[\hat{a}_N \neq a^*\right], \tag{5.7}$$

where $\Delta = EU(a^*) - \max_{a' \in \mathcal{A}\setminus\{a^*\}} EU(a')$ and $\Gamma = EU(a^*) - \min_{a' \in \mathcal{A}} EU(a')$.

Proof. We know $\mathbb{E}\left[\text{EU}(\hat{a}_N)\right]$ is equal to

$$\begin{aligned} &\mathbb{P}\left[\hat{a}_N = a^*\right]\text{EU}(a^*) + \sum_{a \in \mathcal{A}\setminus\{a^*\}} \mathbb{P}\left[a = \hat{a}_N\right]\text{EU}(a) \\ \geq &\mathbb{P}\left[\hat{a}_N = a^*\right]\text{EU}(a^*) + \sum_{a \in \mathcal{A}\setminus\{a^*\}} \mathbb{P}\left[a = \hat{a}_N\right]\min_{a' \in \mathcal{A}}\text{EU}(a') \\ = &\mathbb{P}\left[\hat{a}_N = a^*\right]\text{EU}(a^*) + \mathbb{P}\left[a^* \neq \hat{a}_N\right]\min_{a' \in \mathcal{A}}\text{EU}(a'). \end{aligned}$$

This is equivalent to stating that $\ell(\hat{a}_N) \leq \Gamma \; \mathbb{P}\left[\hat{a}_N \neq a^*\right]$ after some manipulation. Similarly, we have that

$$\mathbb{E}\left[\mathrm{EU}(\hat{a}_N)\right] \leq \mathbb{P}\left[\hat{a}_N = a^*\right] \mathrm{EU}(a^*) + \mathbb{P}\left[\hat{a}_N \neq a^*\right] \max_{a' \in \mathcal{A} \setminus \{a^*\}} \mathrm{EU}(a')$$

which leads to $\ell(\hat{a}_N) \geq \Delta \mathbb{P}\left[a^* \neq \hat{a}_N\right]$. □

The bound is very intuitive: minimizing the probability of the estimated optimal action $\hat{a}_N$ being suboptimal will lead to a bound on the regret. Clearly, for two actions we have $\Delta = \Gamma$. Thus, in the two-action case, minimizing the probability of selecting a suboptimal action is *equivalent* to maximizing the expected utility of the selected action. With more actions, these objectives are not equivalent, but we can see that the difference is controlled in terms of Δ and Γ.

5.1.2 Minimizing the probability of suboptimal action

We now turn to the problem of minimizing $\mathbb{P}\left[a^* \neq \hat{a}_N\right]$. Before doing so though, we first mention few lemmas that are used in proving the main theorem. In particular, Lemma 2 to 5, provide the necessary bounds on the terms involving max that are hard to handle. In Lemma 6 we start providing bounds on the probability of suboptimal actions. Further details of the proofs are available in the supplement.

Lemma 2. *Assuming utility values are non-negative everywhere, for a given action $a \neq a^*$ we have*

$$EU(a^*) \leq \mathbb{E}\Big[\max_{a' \in \mathcal{A} \setminus \{a\}} E\hat{U}_N(a')\Big] \leq \sum_{a' \in \mathcal{A} \setminus \{a\}} EU(a').$$

Proof. Considering Jensen's inequality we have

$$\max_{a' \in \mathcal{A} \setminus \{a\}} \mathbb{E}[\mathrm{E\hat{U}}_N(a')] \leq \mathbb{E}[\max_{a' \in \mathcal{A} \setminus \{a\}} \mathrm{E\hat{U}}_N(a')],$$

and from the definition $\mathrm{EU}(a^*) = \max_{a' \in \mathcal{A} \setminus \{a\}} \mathrm{E\hat{U}}_N(a')$ where expectation is taken with respect to q. Also, since for non-negative utilities we have $\max_{a' \in \mathcal{A} \setminus \{a\}} \mathrm{E\hat{U}}_N(a') \leq \sum \mathrm{E\hat{U}}_N(a')$ the lemma is proved. □

Lemma 3. *Assuming utility values are non-negative everywhere, for a given action $a \neq a^*$ we have*

$$\mathbb{E}\left[E\hat{U}_N(a) - \max_{a' \in \mathcal{A} \setminus \{a\}} E\hat{U}_N(a')\right] \;\leq\; EU(a) - EU(a^*).$$

Proof. Applying the expectation to each term and considering Lemma 2 we conclude the proof. □

Lemma 4. *The following bound for a given action $a \neq a^*$ and $u(a, \boldsymbol{\theta}) \geq 0$ holds:*

$$\left(\mathbb{E}\left[\hat{\mathrm{EU}}_N(a) - \max_{a'\mathcal{A}\backslash\{a\}} \hat{\mathrm{EU}}_N(a')\right]\right)^2 \leq \Big(\sum_{a'\in\mathcal{A}} \mathrm{EU}(a')\Big)^2. \tag{5.8}$$

Proof. From Lemma 2 we can expand the first term as

$$\begin{aligned}&\left(\mathbb{E}\left[\hat{\mathrm{EU}}_N(a) - \max_{a'\mathcal{A}\backslash\{a\}} \hat{\mathrm{EU}}_N(a')\right]\right)^2 \\ &\qquad \leq \mathrm{EU}(a)^2 - 2\mathrm{EU}(a)\mathrm{EU}(a^*) + \Big(\sum_{a'\in\mathcal{A}\backslash\{a\}} \mathrm{EU}(a')\Big)^2 \end{aligned} \tag{5.9}$$

and we know $\Big(\sum_{a\in\mathcal{A}} \mathrm{EU}(a')\Big)^2 = \Big(\mathrm{EU}(a) + \sum_{a'\in\mathcal{A}\backslash\{a\}} \mathrm{EU}(a')\Big)^2$ which means

$$\Big(\sum_{a'\in\mathcal{A}\backslash\{a\}} \mathrm{EU}(a')\Big)^2 = \Big(\sum_{a'\in\mathcal{A}} \mathrm{EU}(a')\Big)^2 - \mathrm{EU}(a)^2 - 2\mathrm{EU}(a)\big(\sum_{a'\in\mathcal{A}\backslash\{a\}} \mathrm{EU}(a')\big). \tag{5.10}$$

Substituting Equation 5.9 in Equation 5.10 indicates that the difference of the first and second expressions in Equation 5.8 is always non-negative. □

Lemma 5. *The following bound for a given action $a \neq a^*$ and $u(a, \boldsymbol{\theta}) \geq 0$ holds:*

$$\left(\mathbb{E}\Big[\max_{a'\mathcal{A}\backslash\{a\}} \hat{\mathrm{EU}}_N(a') - \hat{\mathrm{EU}}_N(a)\Big]\right)^2 \geq \Big(\mathrm{EU}(a^*) - \mathrm{EU}(a)\Big)^2.$$

Proof. Considering both sides in Lemma 3 are positive when multiplied by -1, we can square them. □

In the subsequent lemma, we upper bound the indicator function with a smooth and convex upper bound that will be easier to minimize. The use of *surrogate* function for minimizing indicator has also been used in similar problems (see e.g. [6]).

Lemma 6. *For an optimal action a^* and its estimate $\hat{a}_N$ obtained from sampling, we have $\forall t > 0$,*

$$\mathbb{P}\left[\hat{a}_N \neq a^*\right] \leq \sum_{a\neq a^*} \mathbb{E}\left[\left(t\left(\hat{EU}_N(a) - \max_{a'\in\mathcal{A}\backslash\{a\}} \hat{EU}_N(a')\right) + 1\right)^2\right].$$

Proof. Since we know $\mathbb{I}[v > 0] \leq (tv+1)^2$, we have

$$\begin{aligned}\mathbb{P}\left[\hat{a}_N \neq a^*\right] &= \sum_{a \neq a^*} \mathbb{P}[a = \hat{a}_N] \\ &= \sum_{a \neq a^*} \mathbb{E}\left[\mathbb{I}\left[\hat{\mathrm{EU}}_N(a) > \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{\mathrm{EU}}_N(a')\right]\right] \\ &\leq \sum_{a \neq a^*} \mathbb{E}\left[\left(t\left(\hat{\mathrm{EU}}_N(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{\mathrm{EU}}_N(a')\right) + 1\right)^2\right].\end{aligned}$$

□

Lemma 7. *Assuming utility values are non-negative everywhere, the following bound holds for some $t > 0$:*

$$\begin{aligned}\mathbb{P}\left[\hat{a}_N \neq a^*\right] \leq (k-1) + 2t \sum_{a \in \mathcal{A}\setminus\{a^*\}} \Big(EU(a) - EU(a^*)\Big) \\ + t^2 \sum_{a \in \mathcal{A}\setminus\{a^*\}} \mathbb{V}\left[\left(\hat{EU}_N(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{EU}_N(a')\right)\right] + t^2 \Big(\sum_{a' \in \mathcal{A}} EU(a')\Big)^2.\end{aligned}$$

Proof. Expanding the RHS of Lemma 6 we have

$$\begin{aligned}&\mathbb{E}\left[\left(t\left(\hat{\mathrm{EU}}_N(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{\mathrm{EU}}_N(a')\right) + 1\right)^2\right] \\ &= 1 + 2t\mathbb{E}\left[\left(\hat{\mathrm{EU}}_N(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{\mathrm{EU}}_N(a')\right)\right] + t^2\mathbb{E}\left[\left(\hat{\mathrm{EU}}_N(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \hat{\mathrm{EU}}_N(a')\right)^2\right].\end{aligned}$$

From Lemma 3 we can expand the first expectation and then considering $\mathbb{E}[X^2] = \mathbb{V}[X] + \mathbb{E}[X]^2$ and Lemma 4 we get the bounds in the Lemma 7. □

Lemma 8. *For the bounds detailed in Lemma 7, the value of t that minimizes the upper bound of RHS is*

$$t = \frac{\Delta}{\left(\sum_{a \in \mathcal{A}} EU(a)\right)^2},$$

where $\Delta = EU(a^*) - \max_{a' \in \mathcal{A}\setminus\{a^*\}} EU(a')$.

Proof. We know

$$\Delta = \mathrm{EU}(a^*) - \max_{a' \in \mathcal{A}\setminus\{a^*\}} \mathrm{EU}(a') \leq \mathrm{EU}(a^*) - \mathrm{EU}(a)$$

considering $\max_{a' \in \mathcal{A}\setminus\{a^*\}} \mathrm{EU}(a') \geq \mathrm{EU}(a)$ for $a \neq a^*$, then

$$\mathrm{EU}(a) - \max_{a' \in \mathcal{A}\setminus\{a\}} \mathrm{EU}(a') = \mathrm{EU}(a) - \mathrm{EU}(a^*) \leq -\Delta$$

and we can rewrite Lemma 7 by replacing the second term with its upper bound (because the variance decreases with the number of samples n to ultimately approach zero we disregard it here) as:

$$\mathbb{P}\left[\hat{a}_N \neq a^*\right] \ \leq\ (k-1)\left(1 - 2t\Delta + t^2\Big(\sum_{a\in\mathcal{A}} \mathrm{EU}(a)\Big)^2\right), \tag{5.11}$$

taking the derivative of RHS. with respect to t and equating to zero we will get the solution. $\square$

Lemma 9. *The following bounds on $\mathbb{P}[a^* \neq \hat{a}_N]$ holds as $N \to +\infty$:*

$$\mathbb{P}\left[\hat{a}_N \neq a^*\right] \quad \leq \quad (k-1)\left(1 - \left(\frac{\Delta}{\sum_{a\in\mathcal{A}} EU(a)}\right)^2\right). \tag{5.12}$$

Proof. Replacing the value of t in the bounds in Equation 5.11 yields the proof. $\square$

In words, the probability of estimating the optimal action increases in proportion with the gap between the expected utility of the best and second best actions. Also if, without loss of generality, we assume that the minimum value of the expected utility is zero, then for two action case the RHS of Equation 5.12 is also zero that indicates the bound in this lemma is tight. These bounds are most didactic in two action case.

Putting everything together, the following theorem bounds the probability of suboptimal action selection:

Theorem 10 (Upper bound on the probability of suboptimal actions)**.** *We have the following upper bound probability of suboptimal action selection for k actions in set $\mathcal{A}$, true expected utility $EU(a)$ and its estimation $\hat{EU}_N(a)$ obtained from finite samples:*

$$\begin{aligned}\mathbb{P}\left[\hat{a}_N \neq a^*\right] \leq (k-1) + \sum_{a\in\mathcal{A}\setminus\{a^*\}} t\Bigg(&\Delta + 2\Big(EU(a) - EU(a^*)\Big) \\ &+ t\mathbb{V}\left[\hat{EU}_N(a) - \max_{a'\in\mathcal{A}\setminus a} \hat{EU}_N(a')\right]\Bigg),\end{aligned} \tag{5.13}$$

where t is given in Lemma 8.

Proof. Replacing the value of t obtained in Lemma 8 in Lemma 7, we have this theorem. $\square$

The critical feature of Equation 5.13 is that all terms on the RHS other than the variance are constant with respect to the sampling distribution q. Thus, this theorem suggests that a reasonable surrogate to minimize the regret in Equation 5.6 and consequently maximize the expected utility of the estimated optimal action is to minimize the variance of the difference of the estimated utilities. This result is quite intuitive – if we have a low-variance estimate of the differences of utilities, we will tend to select the best action.

This is aligned with the importance sampling literature where it is well known that the optimal distribution to sample from is the one that minimizes the variance [34, 79] with a closed form solution as summarized in the following:

Corollary 11. *[79] Define*

$$\mathbb{E}[H(\mathbf{s})] = \int H(\mathbf{s}) f(\mathbf{s}) d\mathbf{s} = \int H(\mathbf{s}) \frac{f(\mathbf{s})}{g(\mathbf{s})} g(\mathbf{s}) d\mathbf{s}.$$

Then, the solution to the variance minimization problem

$$\min_{g} \quad \mathbb{V}\left[H(\mathbf{s})\frac{f(\mathbf{s})}{g(\mathbf{s})}\right]$$

is given by

$$g^*(\mathbf{s}) = \frac{|H(\mathbf{s})| f(\mathbf{s})}{\int |H(\mathbf{s})| f(\mathbf{s}) d\mathbf{s}}.$$

Our analysis shows the variance of the function that has to be minimized is of a particular form that depends on the difference of the utilities (rather than each utility independently).

Lemma 12. *We have the following bound on the sum of variances*

$$\sum_{a\in\mathcal{A}\setminus\{a^*\}} \mathbb{V}\left[\max_{a'\in\mathcal{A}\setminus a} \hat{EU}_N(a') - \hat{EU}_N(a)\right]\Bigg) \le \sum_{a\in\mathcal{A}\setminus\{a^*\}} \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} \Upsilon(\boldsymbol{\theta}_i, a)\right)^2\right] - C, \tag{5.14}$$

where $\Upsilon(\boldsymbol{\theta}_i, a) = \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}\left(\max_{a'\in\mathcal{A}\setminus\{a\}} u(\boldsymbol{\theta}_i, a') - u(\boldsymbol{\theta}_i, a)\right)$ *and* $C = \Big(EU(a^*) - EU(a)\Big)^2$.

Proof. We know $\mathbb{V}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ and the upper bound on the first term is proved using the Jensen's inequality (the weights are normalized as will be discussed in Equation 5.16). Also, from Lemma 5 we have the second term. □

5.1.3 Optimal *q*

We established that to find the optimal proposal distribution q^* (i.e. optimal q), we minimize the sum of variances obtained from Theorem 10. Since a^* is unknown, we consider actions in $\mathcal{A}$, rather than just $\mathcal{A}\setminus\{a^*\}$. Since C is independent of q in Equation 5.14, the objective is to minimize the RHS subject to $\int q(\boldsymbol{\theta}) d\boldsymbol{\theta} = 1$ so that the resulting solution is a proper distribution.

The following theorem provides the solution to the optimization problem in Equation 5.14 that we are interested in:

Theorem 13. *Let* $\mathcal{A} = \{a_1, \ldots, a_k\}$ *with non-negative utilities. The optimal distribution*

$q^*(\boldsymbol{\theta})$ is the solution to problem in Equation 5.14 and has the following form:

$$q^*(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta})\sqrt{\sum_{a\in\mathcal{A}}\left(\max_{a'\in\mathcal{A}\setminus\{a\}} u(\boldsymbol{\theta},a') - u(\boldsymbol{\theta},a)\right)^2}. \tag{5.15}$$

Proof. From the objective in Equation 5.14 we have the following value to minimize:

$$\begin{aligned}&\int q(\boldsymbol{\theta}_{1,\dots,N})\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\Upsilon(\boldsymbol{\theta}_i,a)p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}\right)^2 d\boldsymbol{\theta}_{1,\dots,N}\\ &=\frac{1}{N^2}\int\sum_{i=1}^{N}\sum_{j=1}^{N}\Upsilon(\boldsymbol{\theta}_i,a)\Upsilon(\boldsymbol{\theta}_j,a)q(\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_N)d\boldsymbol{\theta}_1\dots\boldsymbol{\theta}_N.\end{aligned}$$

Since all the samples are independent, the joint distribution factorizes as follows: $q(\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_N) = q(\boldsymbol{\theta}_1)\dots q(\boldsymbol{\theta}_N)$. Now if $i \neq j$, it is easy to see that q vanishes and those terms become independent of q. If $i = j$ however, we have one of the terms in the denominator canceled out with the joint. Also because the sum is over similar terms, we have N times the same expression that lead to the Lagrangian of the optimization to become:

$$\mathcal{L}(q,\lambda) = \frac{1}{N}\sum_{a\in\mathcal{A}}\int\frac{\Upsilon(\boldsymbol{\theta},a)^2p(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})}d\boldsymbol{\theta} + \lambda\left(\int q(\boldsymbol{\theta})d\boldsymbol{\theta} - 1\right).$$

Taking the derivative with respect to a fixed $q(\boldsymbol{\theta})$, we have

$$-\frac{1}{N}\sum_{a\in\mathcal{A}}\frac{\Upsilon(\boldsymbol{\theta},a)^2p(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})^2} + \lambda = 0 \Rightarrow \sum_{a\in\mathcal{A}}\frac{p(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})^2}\Upsilon(\boldsymbol{\theta},a)^2 = \lambda N$$

which concludes the theorem since λN only induces a proportionality constant. □

This is quite intuitive – the samples $\boldsymbol{\theta}$ will be concentrated on regions where $p(\boldsymbol{\theta})$ is large, and the difference of utilities between the actions is large, which is precisely the intuition that motivated our work in Figure 5.1. This will tend to lead to the empirically optimal action being the true one, i.e. that $\hat{a}_N$ approaches a^*.

In practice, the normalization constants for p and q are likely to be unknown, meaning that direct use of Equation 5.4 is impossible. However, there are well-known self-normalized variants that can be used in practice with $p(\boldsymbol{\theta}) \propto \tilde{p}(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta}) \propto \tilde{q}(\boldsymbol{\theta})$, namely

$$\hat{\mathrm{EU}}_N(a) = \frac{1}{N}\sum_{i=1}^{N}u(\boldsymbol{\theta}_i,a)\frac{\tilde{p}(\boldsymbol{\theta}_i)}{\tilde{q}(\boldsymbol{\theta}_i)}\bigg/\frac{1}{N}\sum_{i=1}^{N}\frac{\tilde{p}(\boldsymbol{\theta}_i)}{\tilde{q}(\boldsymbol{\theta}_i)} \quad \boldsymbol{\theta}_i \sim \tilde{q}. \tag{5.16}$$

This simply means that for the case of unnormalized $\tilde{p}$ and $\tilde{q}$, all the utility values have to now be reweighted by the slightly more complex value of $\left(\frac{\tilde{p}(\boldsymbol{\theta}_i)}{\tilde{q}(\boldsymbol{\theta}_i)}\Big/\sum_{j=1}^{n}\frac{\tilde{p}(\boldsymbol{\theta}_j)}{\tilde{q}(\boldsymbol{\theta}_j)}\right)$.

Furthermore, as it is hard to directly sample q, we must resort to Markov Chain

Monte Carlo (MCMC) methods [57], e.g. Metropolis-Hastings (MH). This disregards an important aspect, namely that the samples we obtain for q are not truly independent. Rather, the number of effective samples are affected by the mixing rate of the Markov chain. Our derivation above does not account for these mixing rates, which could be important in many applications. For this reason, our experiments will distinguish between two settings: First, one can run an extremely long chain, and subsample from this, approximating nearly independent samples as in the derivation above, which we call "subsampled MC". Secondly, one can run a single Markov chain, as would be typical in practice, which we call "Multiple MC".

5.2 Applications

As discussed earlier, many applications require optimal actions to be selected efficiently given known (but complex) p and u. In this section we provide applications and evaluate how well the samples drawn from p and q^* compare. In these simulations we are interested in finding the optimal action, that is, the one that maximizes the expected utility with the minimum number of samples. As such we generate samples from the true distribution p and the proposed optimal distribution q^* (obtained from Theorem 17 as per the application's specifications) and compute the expected utilities for each action. In case direct sampling is not possible we use Metropolis-Hastings MCMC by initializing the chain at a random point and using a Normal distribution centered at the current sample with isotropic covariance optimally tuned so that around 23% of samples are accepted [78]. In each experiment n samples are generated 200 times and the mean of the percentage of times the true optimal action is selected is reported.

We include two diagnostics for MCMC samplers: in the first one (`Subsampled MC`) we have generated a large chain of 100000 samples and selected random subsamples to compute the best action using the empirical expected utilities $\hat{\mathrm{EU}}_N(a)$. Since samples drawn from Markov chain is typically correlated, this diagnosis will help with ensuring all the samples are independent. In the second diagnostic (`Multiple MC`) we generate 200 chains with equal length each started at random point and run independently to calculate the expected utilities for selecting the best action.

5.2.1 Power-plant Control

We consider a power plant where the temperature of the generator has to be kept in a safe range or otherwise it has to be turned off so that the main generator is not melted as the first example from [47]. Suboptimal actions that either keep the generator on in high temperatures or turn it off in unnecessary cases when temperature is safe and no maintenance is required leads to financial loss for the power plant and should be avoided.

For this problem, we can model the distribution of the temperature at various points in the generator and use a high utility for cases where a safe action of turning the generator on or off is taken. Then we specify the distribution of this temperature and select the optimal action for the generator that maximizes the expected utility. Formally, we specify a simple utility function as,

$$u(\boldsymbol{\theta}, a = \texttt{on}) = \begin{cases} H_{\texttt{on}} & c_1^{(d)} < \boldsymbol{\theta}^{(d)} < c_2^{(d)} \\ L_{\texttt{on}} & \text{otherwise} \end{cases} \tag{5.17}$$

and,

$$u(\boldsymbol{\theta}, a = \texttt{off}) = \begin{cases} H_{\texttt{off}} & c_3^{(d)} < \boldsymbol{\theta}^{(d)} < c_4^{(d)} \\ L_{\texttt{off}} & \text{otherwise} \end{cases} \tag{5.18}$$

where $\boldsymbol{\theta}^{(d)}$ is the temperature at d-th point, H, L (for action `on`/`off`) is the value gained from the power plant for the desired action in the temperature limits. Values of $c_1^{(d)}, c_2^{(d)}, c_3^{(d)}, c_4^{(d)}$ are constants and define the utility functions. We use three distinct one dimensional utilities for simulations:

1. $c_1^{(1)} = 15, c_2^{(1)} = 20, c_3^{(1)} = 15, c_4^{(1)} = 21, H_{\texttt{on}} = 6.5, H_{\texttt{off}} = 5, L_{\texttt{on}} = 1.5, L_{\texttt{off}} = 4;$

2. $c_1^{(1)} = 45, c_2^{(1)} = 50, c_3^{(1)} = 35, c_4^{(1)} = 40, H_{\texttt{on}} = 6.5, H_{\texttt{off}} = 2.5, L_{\texttt{on}} = 1.5, L_{\texttt{off}} = 3;$

3. $c_1^{(1)} = 5, c_2^{(1)} = +\infty, c_3^{(1)} = -\infty, c_4^{(1)} = +\infty, H_{\texttt{on}} = 5, H_{\texttt{off}} = 3, L_{\texttt{on}} = 2.5, L_{\texttt{off}} = 3.$

We perform 3 experiments and for each we specify a distribution. Corresponding to each utility, the following three distributions are used to demonstrate how the samples drawn from p and q^* obtained from Theorem 17 perform in selecting the optimal action:

1. $p(\boldsymbol{\theta}) = 0.7 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 3, 7) + 0.3 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 12, 2)$ where $\mathcal{N}(\boldsymbol{\theta}^{(1)}; \mu, \sigma^2)$ is a normal distribution with mean μ and variance σ^2;

2. $p(\boldsymbol{\theta}) = 0.05 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 3, 1) + 0.2 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 6, 1) + 0.05 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 10, 3) + 0.3 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 15, 2) + 0.05 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 20, 7) + 0.1 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 25, 2) + 0.05 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 30, 3) + 0.2 * \mathcal{N}(\boldsymbol{\theta}^{(1)}; 40, 5)$

3. a log-normal distribution $p(\boldsymbol{\theta}) = \text{Log-}\mathcal{N}(\boldsymbol{\theta}^{(1)}; 0, 1)$.

These distributions represent various temperature behavior at different points in the power-plant. In Figure 5.2, the utility functions in the first column with black indicating the utility of action `on` as detailed in Equation 5.17 and purple specifying action `off` in Equation 5.18. We further illustrate the corresponding distributions in Figure 5.2 with p in blue from three experiments with varying distributions as detailed above and q^* in red in the second column are shown. In the third and fourth columns the result of performing `Subsampled MC` and `Multiple MC` of the Metropolis-Hastings sampler for selecting the best action is shown such that the x-axis represents the number of samples and the y-axis demonstrate the percentage of times the

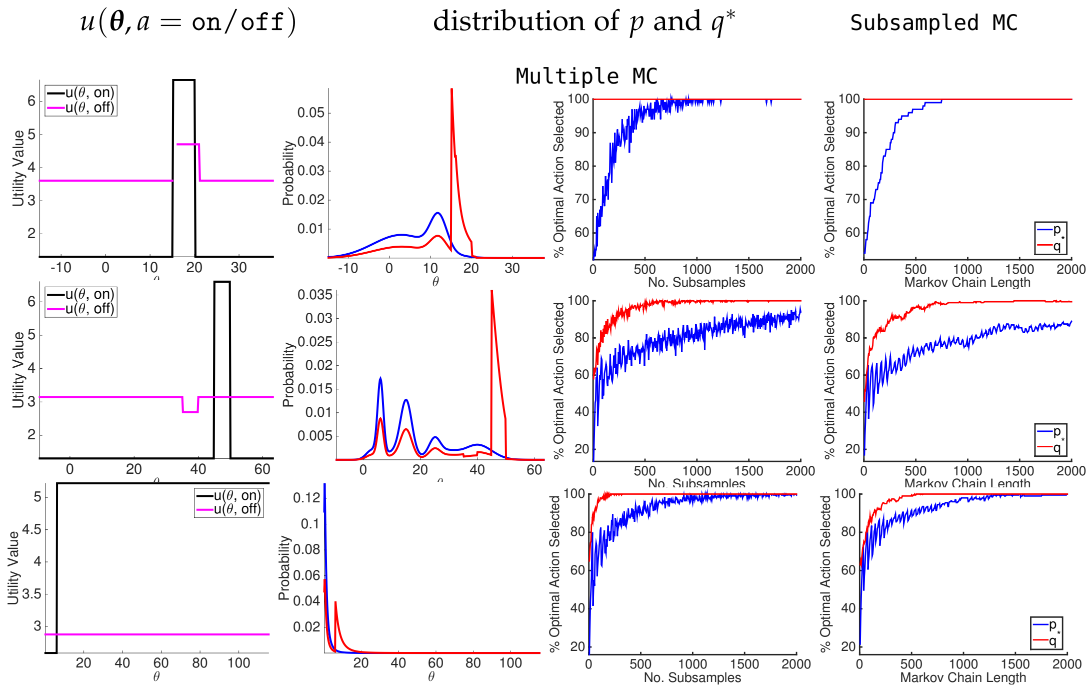
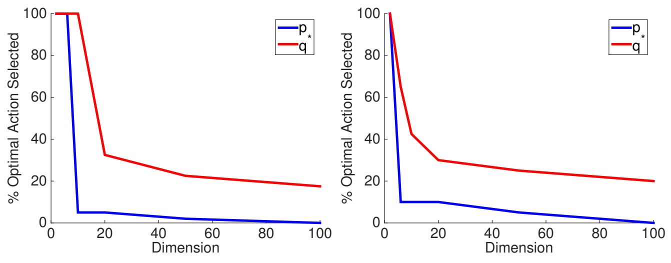


Figure 5.2: Power-plant simulations: the step-valued utility function (as in Equation 5.17 and 5.18) in the first column, the true distribution p (in blue) and q^* (in red) in the second column and in the third and forth columns the result of performing `Subsampled MC` and `Multiple MC` (as described in the text) are shown. In the two right-hand columns, note that q^* achieves the same percentage of optimal action selection performance as p in a mere fraction of the number of samples.

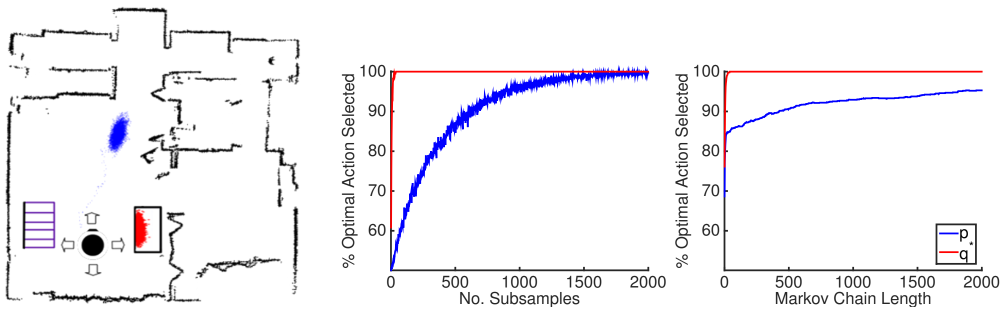
correct optimal action is selected. Here, in general we observe that a significantly smaller number of samples from q^* is needed to select the best action in comparison to the number of samples from p required to achieve the same performance.

To investigate the performance of sampling from p and q^* in higher dimensions, we use a d-dimensional Gaussian mixture corresponding to temperatures at each point in the plant as $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{10}, \boldsymbol{\Sigma}) + \mathcal{N}(\boldsymbol{\theta}; \mathbf{20}, \boldsymbol{\Gamma})$ where $\mathbf{10}$ and $\mathbf{20}$ are d-dimensional vectors with constant value 10 and 20 as the mean and $\boldsymbol{\Sigma}_{i,j} = 5 + \mathbb{I}[i = j]$ and $\boldsymbol{\Gamma}_{i,j} = 3 + 7\mathbb{I}[i = j]$ as $d \times d$ covariance matrix. In addition, the utility function in Equation 5.17 and 5.18 is specified with $c_1^{(d)} = 23, c_2^{(d)} = 25, c_3^{(d)} = 20, c_4^{(d)} = 22, H_{\text{on}} = 50d, H_{\texttt{off}} = 13, L_{\text{on}} = 1.1, L_{\texttt{off}} = 1.5\log(d)$. In Figure 5.3 for $d \in \{2, 4, 10, 20, 50, 80, 100\}$, we observe that in an average of 100 runs of the MCMC with 200 samples, as the dimensions increase using q^* is more efficient. In fact, for a 100-dimensional bimodal Gaussian we are unable to find the optimal action using only 200 samples from p, which should be contrasted with the significantly improved performance given by sampling from q^*.



(a) `Subsampled MC` (b) `Multiple MC`

Figure 5.3: Performance of the decision maker in selecting the best action as the dimension of the problem increases in the power-plant. Note that at 100 dimensions, p is unable to select the optimal action whereas q still manages to select it a fraction of the time (and would do better if more samples were taken).



(a) Environment's Map (b) `Subsampled MC` (c) `Multiple MC`

Figure 5.4: A robot's internal map showing the samples taken from its true belief distribution p (two modes are shown in blue, the second one is slightly obfuscated by the robot) and the optimal sampling distribution q^* derived by our loss calibrated Monte Carlo importance sampler in 5.4(a). In 5.4(b) and 5.4(c) we see the performance (in terms of percentage of optimal action selected) of our loss-calibrated sampling method using q^* leads to near immediate detection of the optimal action in only a few samples.

5.2.2 Robotic Navigation

Another application where sampling has been commonly used is localization in robotics [87]. It is a risk-sensitive-decision making problem where wrong actions may lead to catastrophic incidents like falling down the stairs or crashing into obstacles. Meanwhile, the robot has to navigate to the charger when it has the chance to maintain its power. Due to minimal resources on-board a robot and the nature of the real-time localization problem, it is crucial for the robot to be able to select the optimal action rapidly, yet safely.

The state of the robot is the combination of its coordinates on a map and its heading direction. In our example for these experiments, we use a three dimensional Gaussian belief state distribution with two locations in a room intended to model that a robot's belief update has been confused by symmetries in the environment:

one mode is at the robot's true location and the other at the opposite end of the room.

In this experiment, we consider a map as shown in Figure 5.4(a) where there is a flat in-door environment that the robot can move by selecting one of the four actions forward, backward, right or left. This action will lead to a movement step in robot from the current point on map with the heading direction towards the selected action. In doing so however, the robot has to avoid the stair (low utility region) and select the charging source (high utility region).

Assuming a deterministic transition dynamics model $\boldsymbol{\theta}' = T(\boldsymbol{\theta}, a)$ and denoting $(T(\boldsymbol{\theta}_x, a), T(\boldsymbol{\theta}_y, a))$ as the location of the robot after taking action a from state $\boldsymbol{\theta}$ (that is, moving from the current location in the heading's direction by the selected action) and $\mathcal{R}_{\mathtt{r}}$ the set induced by region r, we use the following utility function:

$$u(\boldsymbol{\theta}, a) = \begin{cases} H & (T(\boldsymbol{\theta}_x, a), T(\boldsymbol{\theta}_y, a)) \in \mathcal{R}_{\mathtt{charger}} \\ L & (T(\boldsymbol{\theta}_x, a), T(\boldsymbol{\theta}_y, a)) \in \mathcal{R}_{\mathtt{stair}} \\ M & \text{otherwise} \end{cases}, \tag{5.19}$$

where $L < M < H$ and $a \in \{\mathtt{forward}, \mathtt{backward}, \mathtt{right}, \mathtt{left}\}$. Using distribution q^* from Theorem 17 as illustrated in Figure 5.4(a), the samples from q^* (in red) concentrated on the charger's location which has higher utility value compared to the samples from p (in blue) that are from the mode of the distribution.

As shown in Figure 5.4(b) and 5.4(c), using distribution q^* and running the same diagnostics as the previous experiment we see significant improvement in selection of the optimal action, requiring only a fraction of the samples of p to achieve the same optimal action selection percentage.

5.3 Conclusion and Future Work

In this chapter we examined an efficient Monte Carlo method for computing the expected utilities. We argued the our loss-sensitive framework improves the efficiency of optimal Bayesian decision-theoretic action selection in comparison to conventional loss-insensitive Monte Carlo methods. In doing so, we derived an "optimal" proposal distribution for importance sampling that minimizes the regret bounds on the expected utility for multiple actions. This regret bound was also related to the probability of selecting suboptimal actions and was subsequently upper bounded by the variance of the utilities. We showed using an alternative distribution with the given form, MC samples more heavily from regions of significance as identified by their difference of utilities. Empirically, we showed that our loss-calibrated Monte Carlo method yields high-accuracy optimal action selections in a fraction of the number of samples required by loss-insensitive samplers in examples of up to 100 dimensions and actual robotics applications.

Although we have seen empirical evidence for efficiency of this Monte Carlo approach, the regret bounds derived in this chapter for multi-action problems are not very tight. Improving these bounds in the future can lead to better performance. In addition, even though the problem was formulated in terms of all the actions except the unknown optimal action, we had to consider all the actions which further approximates the upper bound. Alternatively, we could consider finding the optimal solution by iterating through the differences of each action with all the others. This could lead to a tighter bound as well.

Altogether, this work opens avenues of further research and enables us to suggest a new class of state-of-the-art loss-calibrated Monte Carlo samplers for efficient online Bayesian decision-theoretic action selection. Such a sampling technique is crucial for EEU framework where we need to compute integrals efficiently. This will have a great impact in efficient optimal action selection in the environments where the utility itself is unknown or imprecise. In addition, improving upon this framework, we can see applications where EEU becomes more widely used.

In the subsequent chapter, we will summarize this thesis and will discuss future works that can be built upon our contributions in this thesis.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this thesis, we investigated the problem of efficient Bayesian decision theoretic representation, learning and inference of unknown utilities and performing optimal action selection with respect to the state and utility uncertainty. In particular we were motivated by autonomous interactive systems, recommender systems and robotics scenarios. In these problems the environment is uncertain and the utility function has to be modeled and learned. To this end, we leveraged expected expected utility (EEU) as a general framework for decision making in uncertain environments and examined the problem of (1) learning the distribution of uncertain utilities and (2) efficient selection of the optimal action. In this thesis, we aimed to address both of these problems. For (1), we took a Bayesian non-parametric approach to utility function modeling and learning. We exploited community structure prevalent in collective user preferences using a Dirichlet Process mixture of Gaussian Processes (GPs) and showed how to sparsifying the GP model in order to preserve optimal decisions while ensuring tractable expected utility computations. We addressed (2) in a Monte Carlo framework by deriving an optimal loss-calibrated importance sampling distribution and showed how it can be extended to uncertain utility representations developed in the previous contributions. We briefly summarize our contributions as follows:

- The Von Neumann-Morgenstern theorem provides the necessary conditions for using utilities as a convenient way to model preferences. Utilities, in this case, can be learned from observed preferences. In Chapter 3, we exploited the observation that user populations often decompose into communities of shared preferences and modeled user preferences as an infinite Dirichlet Process mixture of communities. The preferences in each community were used to learn the belief model of utilities in a Gaussian process. The resulting inference algorithm scaled linearly with the number of users. This model was capable of efficiently finding the communities by grouping their preferences.

- As discussed in Chapter 4, we sparsified a Gaussian Process utility model for preference learning. By minimizing the loss incurred by removing a user or item from the posterior in the EEU framework, we proposed a principled method for sparsification. We referred to our method as the valuable vector machine (VVM) to emphasize the importance of considering a loss-sensitive sparsification approach. We showed several popular sparsification methods were recovered in a generalized decision-theoretic framework by specifying the appropriate loss. Overall, our approach contributes to the goal of bridging the gap between decision theory and approximate Bayesian inference by greedily minimizing the expected loss.

- Finally in Chapter 5, we examined another aspect of the EEU framework, namely loss-calibrated Monte Carlo importance sampling methods to improve the efficiency of optimal Bayesian decision-theoretic action selection in comparison to conventional loss-insensitive Monte Carlo methods. We derived an optimal importance sampling distribution to minimize the regret bounds on the expected utility for multiple actions. This, to the best of our knowledge, is the first result linking the utility function and the optimal distribution for Monte Carlo importance sampling in Bayesian decision theory.

In short, in the EEU framework we can compute the posterior from the observations (preferences) by either considering their structure (communities) in Chapter 3 or the loss incurred for sparsification in Chapter 4. Having obtained the posterior, we can compute the expectations for finding the optimal action efficiently using the proposed approach in Chapter 5.

In the following, we discuss further extensions and avenues of research that can be investigated in the future.

6.2 Future Work

While this thesis addresses various aspects of EEU for learning from unknown utilities and computing the optimal action efficiently, there are many unexplored areas for future research:

1. **Loss Calibration in more general cases:**

   (a) **Divergence measures for loss-sensitive inference:** Inference in variational methods and EP can be thought of as minimizing the log likelihood of the desired posterior regardless of the task-at-hand. It seems natural to evaluate the approximated distribution with respect to the task loss. Using other divergence measures such as f-divergence as in ([42]), we can bridge this gap and obtain a task-specific measure to perform inference. This is because f-divergence is defined based on a convex function f that can be specified using the task-loss. Hence using a task-specific divergence instead of a general purpose KL. For instance for two actions (binary problems) and convex loss functions, [64] provides appropriate f functions and corresponding divergences. The relation between f-divergence, Bregman divergences and the binary experiment losses has been considered in [36, 52, 76].

In another class of divergence measures, namely α-divergences, [54] discussed the general problem of inference that includes variational inference and EP that corresponds to selection of α. It was argued that the α value that determines the divergence measure leads to selection of different regions of the true posterior. Different task losses correspond to divergences that focus on different regions of the posterior.

(b) **Loss-sensitive message passing in graphical models:** Approximate message passing ([92, 94]) is one of the main inference methods in graphical models. Intractable factors in the graph are approximated and exact message passing is performed on all the rest. In the factors that have to be approximated, task loss can be incorporated to tailor the approximation for a particular problem. In this case, the messages sent are calibrated with the task loss incurred by the approximations.

2. **Efficient Action Selection in more general cases:**

   (a) **Continuously parameterized actions for decision-theoretic action selection:** In many real-world problems, such as system control, the action is better modeled continuously and discretization makes the solution intractable [49]. In such problems efficient action selection, similar to the one discussed in Chapter 5, is crucial. However, since the action space is continuous, formulating a closed form solution for the proposal distribution that resembles the one found in this thesis is challenging because it involves an integral that is again hard to compute.

   (b) **Uncontrollable sampling:** When the utility partially depends on auxiliary variables that cannot be directly observed, optimal action selection can be challenging. In many real world applications there are aspects of the problem, that even though hard to sample, affect the utility. For instance in marketing when sampling individuals, age is an observable attribute that can be easily sampled, however, other aspects such as their interest in the product is not known in advance and only determined after the samples are all collected. These latent variables can be sampled using the efficient action selection algorithm discussed in this thesis however since these latent variables indirectly affect the expected utility, this might complicate the sampling.

   (c) **Applications in active learning:** In active learning, the learner is able to query the user or some other oracle for the label of a particular instance. A common practice in active learning is to determine a hypothesis model for sample selection based on the performance of that model. For example selective sampling ([26]) and its improved weighted version ([10]), performs active learning by iteratively selecting "regions" of the input space and refining the hypothesis models. In the light of the discussions in this thesis, the optimal hypothesis model is the optimal action sought using minimum samples drawn from a region.

(d) **Improving the bounds for optimal action selection:** Although the upper bounds in Chapter 5 led to a natural objective to loss-calibrated sampling, i.e. variance reduction, these bounds are not very tight. Improvements in these bounds may lead to other objectives that may exhibit better sampling performance.

3. **New Applications:**

(a) **Novel nonparametric approaches to community learning:** As we showed in Chapter 3, hierarchical Bayesian modeling of the users is useful in directly modeling and learning community-based preference structure. As such, extensions of the current nonparametric approaches to learning the communities from the preferences can be investigated. These nonparametric approaches, e.g. as an extension of Indian Buffet Process ([35]) and Mallows processes ([2]) that are specifically used for preference learning without considering user communities can take advantage of the social information available for each user.

(b) **Improving optimization methods:** Stochastic optimization methods, e.g. stochastic gradient descent, use a noisy gradient computed from a subset of training examples to efficiently update current solutions. These algorithms are increasingly popular due to their simplicity of use and speed. The ideas in this thesis can be used in the light of sampling the observations (i.e. as states) that are more important to finding the optimal parameters that optimize the objective loss function. In such cases, instead of sampling the observations uniformly, one can sample instances that help further reduce the expected loss. Although variance reduction is widely used for improving stochastic optimization, further research in using the loss function for efficient sampling may lead to further performance boost. It might be used in the Bayesian optimization solutions as well ([19]) where the function to be optimized is uncertain and only its values at particular points are known.

(c) **Extensions to sequential decision making:** In sequential problems (e.g. Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs)), a decision maker selects an action that affects the world and changes its state stochastically ([17]). The objective of the decision maker is to perform these series of actions in this uncertain environment to maximize some "value" function that may not be known in advance. This requires exploration in the space with exponentially large search space. The problem formulation is slightly different from the one considered in this thesis, namely that sequential problems are looking for a policy rather than a single action and the state distribution depends on the current action. However, sampling combined with dynamic programming is effective in finding the solution to these problems and is a natural extension of what is discussed in this thesis. For instance Monte Carlo tree search (MCTS) ([20]) as a framework for best action selection for current state in online planning is a straightforward extension of our approach where the samples from the tree paths are taken to maximize the value function similar to the approach discussed in Chapter 5.

6.3 Concluding Remarks

In this thesis, we discussed efficient and scalable computation of expected expected utility for optimal Bayesian decision-theoretic action selection. Applications in preference learning and robotics decision-making scenarios demonstrate the effectiveness of the theoretical foundations laid in this thesis. We hope the ideas in this thesis pave the way for future advances that address important practical problems at the intersection of Bayesian decision theory and scalable machine learning.

Appendix A

Alternative Formulation of the Loss-calibrated MC Action Selection

A.1 Minimizing the probability of suboptimal action

In this appendix, we provide an alternative formalization of the loss-calibrated Monte Carlo method discussed in Chapter 5.

Lemma 14. *For an optimal action a^* and its estimate $\hat{a}_N$ obtained from sampling, we have $\forall t > 0$,*

$$\mathbb{P}\left[a^* \neq \hat{a}_N\right] \leq \sum_{a \neq a^*} \sum_{a' \neq a} \mathbb{E}\left[\left(t\left(\hat{EU}_N(a) - \hat{EU}_N(a')\right) + 1\right)^2\right].$$

Proof. Firstly, decompose $\mathbb{P}\left[a^* \neq \hat{a}_N\right]$ as

$$\begin{aligned}\sum_{a \neq a^*} \mathbb{P}[a = \hat{a}_N] &\leq \sum_{a \neq a^*} \sum_{a' \neq a} \mathbb{P}[\mathrm{E\hat{U}}_N(a) > \mathrm{E\hat{U}}_N(a')] \\ &= \sum_{a \neq a^*} \sum_{a' \neq a} \mathbb{E}\left[\mathbb{I}\left[\mathrm{E\hat{U}}_N(a) > \mathrm{E\hat{U}}_N(a')\right]\right].\end{aligned}$$

Applying the inequality that $\mathbb{I}[v > 0] \leq (tv + 1)^2$ gives the result.

□

The next theorem then relates the value inside the above expectation to the variance, thus bonding the probability of incorrect action selection by the sum of variances.

Theorem 15 (Upper bound on the probability of suboptimal actions)**.** *We have the following upper bound on the probability of suboptimal action selection for k actions in set $\mathcal{A}$, true expected utility $EU(a)$ and its estimation $\hat{EU}_N(a)$ obtained from finite samples:*

$$\begin{aligned}\mathbb{P}\left[a^* \neq \hat{a}_N\right] \leq \sum_{a \neq a^*} \sum_{a' \neq a} \Big(1 + 2t\Big(EU(a) - EU(a')\Big) \\ + t^2 \mathbb{V}\left[\hat{EU}_N(a) - \hat{EU}_N(a')\right] + t^2\Big(EU(a) - EU(a')\Big)^2\Big),\end{aligned} \tag{A.1}$$

Proof. Expand the quadratic in Lemma 14 and use $\mathbb{E}[X^2] = \mathbb{V}[X] + \mathbb{E}[X]^2$ and $\mathbb{E}[\hat{\mathrm{EU}}_N(a) - \hat{\mathrm{EU}}_N(a')] = \mathrm{EU}(a) - \mathrm{EU}(a')$. □

In general, one would like to select the constant t to minimize this bound. As this depends on the variance, which is a function of q and the number of samples n, this is difficult to do analytically. However, we expect the variance to decrease

Lemma 16. *For the bounds detailed in Lemma 15, if the variance term is zero, the value of t that minimizes the upper bound is*

$$t = \frac{\sum_{a \neq a^*} \sum_{a' \neq a} EU(a') - EU(a)}{\sum_{a \neq a^*} \sum_{a' \neq a} \left(EU(a) - EU(a')\right)^2}.$$

Proof. In general, the value of t minimizing $2ta + t^2 b$ is $t = -a/b$. □

The critical feature of Equation A.1 is that all terms on the RHS other than the variance are constant with respect to the sampling distribution q. Thus, this theorem suggests that a reasonable surrogate to minimize the regret in Equation 5.6 and consequently maximize the expected utility of the estimated optimal action is to minimize the variance of the difference of the estimated utilities. This result is quite intuitive – if we have a low-variance estimate of the differences of utilities, we will tend to select the best action.

This is aligned with the importance sampling literature where it is well known that the optimal distribution to sample from is the one that minimizes the variance [34, 79]. Our analysis shows the variance of the function that has to be minimized is of a particular form that depends on the difference of the utilities (rather than each utility independently).

A.1.1 Optimal q

We established that to find the optimal proposal distribution q^* (i.e. optimal q), we minimize the sum of variances obtained from Theorem 15. Since a^* is unknown, we sum over all actions in $\mathcal{A}$, rather than just $\mathcal{A} \backslash \{a^*\}$. Since everything except variance in Equation A.1 is independent of q, we formulate the objective

$$\min_q \sum_{a \in \mathcal{A}} \sum_{a' \in \mathcal{A} \backslash \{a\}} \mathbb{V}\left[\hat{\mathrm{EU}}_N(a) - \hat{\mathrm{EU}}_N(a')\right] \text{ s.t.} \int q(\boldsymbol{\theta}) d\boldsymbol{\theta} = 1. \tag{A.2}$$

Here, the constraint on q is to ensure the resulting solution is a proper probability distribution.

The following theorem provides the solution to the optimization problem in Equation A.2 that we are interested in.

Theorem 17. *Let $\mathcal{A} = \{a_1, \dots, a_k\}$ with non-negative utilities. The optimal distribution $q^*(\boldsymbol{\theta})$ is the solution to problem in Equation A.2 and has the following form:*

$$q^*(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) \sqrt{\sum_{a \in \mathcal{A}} \sum_{a' \in \mathcal{A} \setminus \{a\}} (u(\boldsymbol{\theta}, a) - u(\boldsymbol{\theta}, a'))^2}. \tag{A.3}$$

Proof. Since we know $\mathbb{V}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, for computing the objective in Equation A.2 the second expectation becomes $(\mathrm{EU}(a) - \mathrm{EU}(a'))^2$ and is independent of q, then we only need to minimize $\mathbb{E}\left[(\hat{\mathrm{EU}}_N(a) - \hat{\mathrm{EU}}_N(a'))^2\right]$. Consider this value for a particular pair (a, a'). Denoting $\Upsilon(\boldsymbol{\theta}_i, a, a') = u(\boldsymbol{\theta}, a) - u(\boldsymbol{\theta}, a')$, this is equal to

$$\begin{aligned}\int q(\boldsymbol{\theta}_{1,\dots,n}) \left(\frac{1}{N} \sum_{i=1}^{N} \frac{\Upsilon(\boldsymbol{\theta}_i, a, a') p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)} \right)^2 d\boldsymbol{\theta}_{1,\dots,N} \\ = \frac{1}{N^2} \int \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\Upsilon(\boldsymbol{\theta}_i, a, a') \Upsilon(\boldsymbol{\theta}_j, a, a') p(\boldsymbol{\theta}_i) p(\boldsymbol{\theta}_j)}{q(\boldsymbol{\theta}_i) q(\boldsymbol{\theta}_j)} \\ \times q(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N) d\boldsymbol{\theta}_{1,\dots,N}.\end{aligned}$$

Since all the samples are independent, $q(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N) = q(\boldsymbol{\theta}_1) \dots q(\boldsymbol{\theta}_N)$. Now if $i \neq j$, it is easy to see that q vanishes and those terms become independent of q. If $i = j$ however, only one of the terms in the denominator cancels out with the joint. Also because the sum is over similar terms, we have n times the same expression, leading to the Lagrangian of

$$\frac{1}{N} \sum_{a \in \mathcal{A}} \sum_{a' \in \mathcal{A} \setminus \{a\}} \int \frac{\Upsilon(\boldsymbol{\theta}, a, a')^2 p(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} + \lambda \left(\int q(\boldsymbol{\theta}) d\boldsymbol{\theta} - 1 \right).$$

Taking the derivative with respect to $q(\boldsymbol{\theta})$, we have that

$$-\frac{1}{N} \sum_{a \in \mathcal{A}} \sum_{a' \in \mathcal{A} \setminus \{a\}} \frac{\Upsilon(\boldsymbol{\theta}, a, a')^2 p(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})^2} + \lambda = 0$$

which concludes the theorem since λN only induces a proportionality constant. □

This is quite intuitive – the samples $\boldsymbol{\theta}$ will be concentrated on regions where $p(\boldsymbol{\theta})$ is large, and the difference of utilities between the actions is large, which is precisely the intuition that motivated our work in Figure 5.1. This will tend to lead to the empirically optimal action being the true one, i.e. that $\hat{a}_N$ approaches a^*.

Bibliography

1. Airoldi, E. M.; Blei, D. M.; Fienberg, S. E.; and Xing, E. P., 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9 (2008). (cited on page 50)

2. Alnur Ali, M. M., Thomas Brendan Murphy and Chen, H., 2010. Preferences in college applications – a nonparametric bayesian analysis of top-10 rankings. In *Neural Information Processing Systems (NIPS) Workshop on Computational Social Science 2010*. (cited on page 82)

3. Auer, P.; Cesa-Bianchi, N.; and Fischer, P., 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning Journal*, 47, 2-3 (May 2002), 235–256. (cited on page 56)

4. Azari, H.; Parks, D.; and Xia, L., 2012. Random Utility Theory for Social Choice. In *Advances in Neural Information Processing Systems*, 126–134. (cited on page 32)

5. Bai, H.; Hsu, D.; Kochenderfer, M. J.; and Lee, W. S., 2011. Unmanned Aircraft Collision Avoidance using Continuous-State POMDPs. In *Robotics: Science and Systems VII, University of Southern California, Los Angeles, CA, USA, June 27-30, 2011*. (cited on page 1)

6. Bartlett, P.; Jordan, I. M.; and McAaliffe, J. D., 2006. Convexity, Classification, and Risk Bounds. *Journal of the American Statistical Association*, 101, 473 (Mar. 2006), 138–156. (cited on page 68)

7. Berger, J., 2010. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2nd edn. ISBN 9781441930743. (cited on pages 3, 7, and 10)

8. Berger, J. O., 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer New York, New York, NY. ISBN 978-1-4419-3074-3, 978-1-4757-4286-2. (cited on page 10)

9. Bernardo, J. M. and Smith, A. F. M., 1994. *Bayesian Theory*. John Wiley & Sons, Inc. ISBN 9780470316870. (cited on page 7)

10. Beygelzimer, A.; Dasgupta, S.; and Langford, J., 2009. Importance Weighted Active Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 49–56. ACM, Montreal, Quebec, Canada. (cited on page 81)

11. Birlutiu, A.; Groot, P.; and Heskes, T., 2010. Multi-task preference learning with an application to hearing aid personalization. In *Neurocomputing*, vol. 73. (cited on pages 32 and 50)

12. Bishop, C., 2006. *Pattern Recognition and Machine Learning \textbar Springer*. (cited on pages 9, 22, and 23)

13. Blei, D. M.; Ng, A. Y.; and Jordan, M. I., 2003. Latent Dirichlet Allocation. *Journal Machine Learning Research*, 3 (Mar. 2003), 993–1022. (cited on page 13)

14. Bonilla, E. V.; Guo, S.; and Sanner, S., 2010. Gaussian process preference elicitation. In *Advances in Neural Information Processing Systems*, 153–160. (cited on pages 32, 35, 36, 37, and 50)

15. Boutilier, C., 2003. On the Foundations of Expected Expected Utility. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, International Joint Conference on Artificial Intelligence'03, 285–290. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. (cited on pages 3 and 32)

16. Boutilier, C.; Das, R.; Kephart, J.; Tesauro, G.; and Walsh, W., 2003. Cooperative Negotiation in Autonomic Systems using Incremental Utility Elicitation. In *Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, 89–97. Morgan Kaufmann, San Francisco, CA. (cited on page 2)

17. Boutilier, C.; Dean, T.; and Hanks, S., 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 1 (1999), 1–93. (cited on page 82)

18. Braziunas, D. and Boutilier, C., 2006. Preference elicitation and generalized additive utility. In *proceedings of the 21st national conference on Artificial intelligence-Volume 2*, 1573–1576. AAAI Press. (cited on page 31)

19. Brochu, E.; Cora, V. M.; and de Freitas, N., 2010. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv:1012.2599 [cs]*, (Dec. 2010). ArXiv: 1012.2599. (cited on pages 32 and 82)

20. Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S.; et al., 2012. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4, 1 (2012), 1–43. (cited on page 83)

21. Chajewska, U. and Koller, D., 2000. Utilities as random variables: Density estimation and structure discovery. In *Uncertainty in Artificial Intelligence*, 63–71. Morgan Kaufmann Publishers Inc. (cited on page 50)

22. Chajewska, U.; Koller, D.; and Parr, R., 2000. Making Rational Decisions using Adaptive Utility Elicitation. In *In Proceedings of the Seventeenth National Conference on Artificial Intelligence*. (cited on pages 2 and 31)

23. Chu, W. and Ghahramani, Z., 2005. Extensions of gaussian processes for ranking: semi-supervised and active learning. In *Workshop on Learning to Rank at Advances in Neural Information Processing Systems*. (cited on pages 35 and 38)

24. Chu, W. and Ghahramani, Z., 2005. Preference learning with Gaussian processes. In *International Conference in Machine Learning*, 137–144. ACM, Bonn, Germany. (cited on page 32)

25. Chu, W. and Ghahramani, Z., 2005. Preference learning with Gaussian processes. In *International Conference in Machine Learning* (Bonn, Germany, 2005), 137–144. ACM, New York, NY, USA. (cited on page 50)

26. Cohn, D.; Ladner, R.; and Waibel, A., 1994. Improving Generalization with Active Learning. In *Machine Learning Journal*, 201–221. (cited on page 81)

27. Eric, B.; Freitas, N. D.; and Ghosh, A., 2008. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems*, 409–416. (cited on page 50)

28. Freed, M.; Harris, R.; and Shafto, M., 2004. Comparing methods for UAV-based autonomous surveillance. Technical report, NASA. http://ntrs.nasa.gov/search.jsp?R=20040068395. (cited on page 1)

29. Frigyik, B. A.; Kapila, A.; and Gupta, M. R., 2010. Introduction to the Dirichlet Distribution and Related Processes. Technical Report UWEETR-2010-0006, University of Washington. (cited on pages 16 and 22)

30. Fürnkranz, J. and Hüllermeier, E., 2010. *Preference Learning*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edn. (cited on page 35)

31. Gelman, A.; Carlin, J. B.; and Stern, H. S., 2014. *Bayesian data analysis*, vol. 2. (cited on pages 7, 12, and 24)

32. Gelman, A.; Christian, R.; Chopin, N.; and Rousseau, J., 1995. *Bayesian Data Analysis*. CRC press. (cited on pages 3, 7, 12, and 26)

33. Geweke, J., 1989. Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica*, 57, 6 (1989), 1317–1339. (cited on page 65)

34. Glasserman, P., 2004. *Monte Carlo Methods in Financial Engineering*. Applications of Mathematics. Springer, 1st edn. ISBN 0-387-00451-3, 978-0-387-00451-8. (cited on pages 71 and 86)

35. Griffiths, T. L. and Ghahramani, Z., 2011. The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research*, 12 (Jul. 2011), 1185–1224. (cited on pages 16 and 82)

36. Grunwald, P. D. and Dawid, A. P., 2004. Game Theory, Maximum Entropy, Minimum Discrepancy and Robust Bayesian Decision Theory. *The Annals of Statistics*, 32, 4 (Aug. 2004), 1367–1433. (cited on page 80)

37. Guiver, J. and Snelson, E., 2009. Bayesian inference for Plackett-Luce ranking models. In *proceedings of the 26th annual international conference on machine learning*, 377–384. ACM. (cited on page 32)

38. Guo, S. and Sanner, S., 2010. Real-time multiattribute Bayesian preference elicitation with pairwise comparison queries. In *Artificial Intelligence and Statistics*. (cited on pages 31 and 50)

39. Hjort, N. L.; Holmes, C.; Peter, M.; and G., W. S., 2010. *Bayesian Nonparametrics*, vol. 10. Cambridge University Press. (cited on pages 16, 20, 21, and 22)

40. Howard, R., 1966. Information Value Theory. *Systems Science and Cybernetics, IEEE Transactions on*, 2, 1 (Aug. 1966), 22 –26. (cited on page 55)

41. Hunter, D. R., 2004. Mm algorithms for generalized bradley-terry models. *Annal of Statistics*, 32, 1 (02 2004), 384–406. (cited on page 31)

42. John Lu, Z. Q., 2007. Statistical Inference Based on Divergence Measures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170, 3 (Jul. 2007), 857–858. (cited on page 80)

43. Kamishima, T., 2003. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 583–588. (cited on page 47)

44. Keerthi, S. S. and Chu, W., 2005. A Matching Pursuit Approach to Sparse Gaussian Process Regression. In *Advances in Neural Information Processing Systems*. (cited on page 51)

45. Koller, D. and Friedman, N., 2009. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press. (cited on pages 7, 9, 23, 25, and 30)

46. Krause, A. and Golovin, D., 2012. Submodular function maximization. Technical report, ETH Zurich. (cited on page 54)

47. Lacoste-Julien, S.; Huszar, F.; and Ghahramani, Z., 2011. Approximate Inference for the Loss-calibrated Bayesian. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, vol. 15, 416–424. (cited on pages 51, 52, 61, 65, and 73)

48. Lawrence, N.; Seeger, M.; and Herbrich, R., 2003. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*. (cited on pages 51 and 53)

49. Lazaric, A.; Marcello, R.; and Andrea, B., 2007. Reinforcement learning in continuous action spaces through sequential monte carlo methods. In *Advances in Neural Information Processing Systems*. (cited on page 81)

50. Luce, D., 1961. *Journal of the American Statistical Association*, 56, 293 (1961), 172–174. (cited on page 32)

51. MacKay, D., 2003. *Information Theory, Inference and Learning Algorithms*. (cited on pages 9, 17, and 19)

52. Masnadi-shirazi, H. and Vasconcelos, N., 2009. On the design of loss functions for classification: theory, robustness to outliers, and SavageBoost. In *Advances in Neural Information Processing Systems 21* (Eds. D. Koller; D. Schuurmans; Y. Bengio; and L. Bottou), 1049–1056. Curran Associates, Inc. (cited on page 80)

53. Meeds, E. and Osindero, S., 2005. An alternative infinite mixture of gaussian process experts. In *Advances in Neural Information Processing Systems*. (cited on page 50)

54. Minka, T., 2005. Divergence Measures and Message Passing. Technical report. (cited on pages 22 and 81)

55. Minka, T. P., 2001. *A Family of Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Massachusetts Institute of Technology. AAI0803033. (cited on pages 22 and 24)

56. Müller, P. and Quintana, F. A., 2004. Nonparametric Byesian data analysis. *Statistical Science*, 19 (2004), 95–110. (cited on pages 22 and 35)

57. Neal, R. M., 1993. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical report, University of Toronto, University of Toronto. (cited on pages 26 and 73)

58. Neal, R. M., 1998. Markov chain sampling methods for dirichlet process mixture models. Technical report, Deptartment of Statistics, University of Toronto. (cited on page 42)

59. Neal, R. M., 1998. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical report, University of Toronto. (cited on page 64)

60. Neal, R. M., 2011. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2 (2011). (cited on page 29)

61. Neil, H.; Ferenc, H.; Zoubin, G.; and Mate, L., 2011. Bayesian Active Learning for Classification and Preference Learning. In *Computing Research Repository*. (cited on page 61)

62. Neil, H.; Miguel, H.-L. J.; Ferenc, H.; and Zoubin, G., 2012. Collaborative Gaussian Processes for Preference Learning. In *Advances in Neural Information Processing Systems*. (cited on page 61)

63. Neumann, J. V. and Morgenstern, O., 1944. *Theory of Games and Economic Behavior*. Princeton University Press. ISBN 0691119937. (cited on page 36)

64. Nguyen, X.; Wainwright, M. J.; and Jordan, M. I., 2009. On surrogate loss functions and f-divergences. *The Annals of Statistics*, 37, 2 (2009), 876–904. (cited on page 80)

65. Nowicki, K. and Snijders, T. A. B., 2001. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96 (2001). (cited on page 50)

66. Patrascu, R.; Boutilier, C.; Das, R.; Kephart, J. O.; Tesauro, G.; and Walsh, W. E., 2005. New Approaches to Optimization and Utility Elicitation in Autonomic Computing. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, 140–145. AAAI Press, Pittsburgh, Pennsylvania. (cited on page 2)

67. Peter, O. and Whye, T. Y., 2010. Bayesian Nonparametric Models. In *Encyclopedia of Machine Learning*. Springer. (cited on pages 16 and 22)

68. Plackett, R. L., 1975. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24, 2 (1975), 193–202. (cited on page 32)

69. Platt, J. C.; Burges, C. J. C.; Swenson, S.; Weare, C.; and Zheng, A., 2002. Learning a Gaussian Process Prior for Automatically Generating Music Playlists. In *Advances in Neural Information Processing Systems*. (cited on page 35)

70. Postlewaite, A., 2011. Social norms and social assets. *Annual Review of Economics*, 3 (2011), 239–259. (cited on page 35)

71. Quiñonero-Candela, J. and Rasmussen, C., 2005. A unifying view of sparse approximate Gaussian process regression. In *Journal of Machine Learning Research*, 1939–1959. (cited on page 60)

72. Rasmussen, C. E., 2000. The infinite gaussian mixture model. In *Advances in Neural Information Processing Systems*, vol. 12. (cited on page 42)

73. Rasmussen, C. E. and Ghahramani, Z., 2002. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems*. (cited on page 50)

74. Rasmussen, C. E. and Williams, C., 2006. *Gaussian Processes for Machine Learning*. (cited on pages 4, 16, 17, and 20)

75. Rasmussen, C. E. and Williams, C., 2006. *Gaussian Processes for Machine Learning*. (cited on pages 35, 58, and 60)

76. Reid, M. D. and Williamson, R. C., 2011. Information, Divergence and Risk for Binary Experiments. *Journal of Machine Learning Research*, 12 (Mar. 2011), 731–817. (cited on page 80)

77. Robert, C., 2001. *The Bayesian Choice*. Springer Texts in Statistics. Springer, 2nd edn. (cited on pages 3, 7, and 30)

78. Roberts, G. O.; Gelman, A.; and Gilks, W. R., 1997. Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms. *The Annals of Applied Probability*, 7, 1 (Feb. 1997), 110–120. (cited on page 73)

79. Rubinstein, R. Y., 1981. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., 1st edn. ISBN 0471089176. (cited on pages 26, 71, and 86)

80. Seeger, M. W., 2003. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. Ph.D. thesis, University of Edinburgh. (cited on page 58)

81. Smola, A. J. and Bartlett, P., 2001. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems*. (cited on page 51)

82. Snelson, E. and Ghahramani, Z., 2006. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*. (cited on pages 51 and 60)

83. Snoek, J.; Larochelle, H.; and Adams, R. P., 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959. (cited on page 55)

84. Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M., 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Advances in Neural Information Processing Systems*. (cited on pages 52 and 61)

85. Teh, Y. W., 2010. Dirichlet process. In *Encyclopedia of machine learning*, 280–287. Springer. (cited on pages 4 and 16)

86. Teh, Y. W.; Jordan, M. I.; Beal, M. J.; and Blei, D. M., 2004. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101 (2004), 1566–1581. (cited on page 16)

87. Thrun, S., 2000. Probabilistic Algorithms in Robotics. *AI Magazine*, 21 (2000), 93–109. (cited on pages 64, 65, and 76)

88. Tzikas, D.; Likas, C.; and Galatsanos, N., 2008. The variational approximation for Bayesian inference. *IEEE Signal Processing Magazine*, 25, 6 (Nov. 2008), 131–146. (cited on page 23)

89. Von Neumann, J. and Morgenstern, O., 1944. *Theory of Games and Economic Behavior*. Princeton University Press. Princeton University Press. (cited on page 29)

90. Wainwright, M. J. and Jordan, M. I., 2008. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1, 1-2 (Jan. 2008), 1–305. (cited on pages 9, 23, and 24)

91. Wang, C. and Neal, R. M., 2013. MCMC methods for Gaussian process models using fast approximations for the likelihood. Technical report, University of Toronto. (cited on page 64)

92. Winn, J. and Bishop, C. M., 2005. Variational Message Passing. *The Journal of Machine Learning Research*, 6 (2005), 661–694. (cited on pages 23, 24, and 81)

93. Xu, Z.; Kersting, K.; and Joachims, T., 2010. Fast active exploration for link-based preference learning using Gaussian processes. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*. (cited on page 35)

94. Yedidia, J. S., 2011. Message-Passing Algorithms for Inference and Optimization. *Journal of Statistical Physics*, 145, 4 (2011), 860–890. (cited on page 81)

95. Yedidia, J. S.; Freeman, W. T.; and Weiss, Y., 2003. Exploring artificial intelligence in the new millennium. chap. Understanding Belief Propagation and Its Generalizations, 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1-55860-811-7. (cited on page 25)

96. Yellott, J. I., 1977. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15, 2 (1977), 109–144. (cited on page 32)