Bayesian User Behavioral Modeling for Self-Learning Comfort Systems in Vehicles

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN
(Dr.-Ing.)

von der KIT-Fakultät für
Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Dipl.-Ing. María Guinea Márquez
geb. in Madrid, Spanien

Tag der mündlichen Prüfung:	5. Juli 2024
Hauptreferent:	Prof. Dr.-Ing. Eric Sax
Korreferent:	Prof. Dr. rer. nat. Cornelius Neumann

Abstract

The growing popularity of personalized services, such as Netflix or Spotify, has pushed the automotive industry to rethink the interior space in vehicles to meet customer demands for more individual and predictive comfort systems. To satisfy user's needs, in-vehicle comfort functions play a fundamental role. Automating the in-car comfort functions according to the particular needs and routines of each individual user would not only make customers feel their car is personalized, but also increase their level of comfort and create a more enjoyable and less distracting driving experience.

The main goal of this dissertation is, therefore, to apply machine learning techniques to transform the in-vehicle comfort functionalities into self-learning comfort systems, capable of learning the individual user behavioral patterns – such as turning on the seat heater when it is cold – in order to seamlessly control the desired vehicle functionality as the user would do.

To accomplish this objective, the general requirements for a self-learning comfort system in vehicles are identified through a comprehensive literature review. Furthermore, taking those system requirements as a foundation, a generic system architecture describing the main system components is introduced. Afterwards, a novel Bayesian nonparametric model for understanding user behavior with in-vehicle comfort functions is presented. The user behavioral model is the fundamental element of a self-learning comfort system. To infer the model's parameters, a variational inference algorithm is introduced. The algorithm extends the current state-of-the-art in variational inference techniques by permitting a truncation-free inference approach with minimal resource usage, enhancing the model's flexibility and adaptability.

To prove the feasibility of a self-learning comfort system in real-world settings, a preliminary prototype of a self-learning window system is developed as an initial concept demonstration. Based on the knowledge gained from the prototypical implementation, the initial inference algorithm is optimized. The optimized variational inference algorithm expands the previous one by enabling the incorporation of expert knowledge into the learning procedure.

Then, the user behavioral model and the inference methods are examined using synthetically generated data that represents challenging scenarios. Finally, a comprehensive evaluation of the model's performance is presented, for which real-world data collected over a six-month experimental study is employed.

Overall, self-learning comfort systems offer a promising approach to satisfy the user needs for personalized and intelligent interior spaces in vehicles. The user behavioral model presented in this work stands as a fundamental element for their development.

Kurzfassung

Die zunehmende Beliebtheit von personalisierten Anwendungen wie Netflix oder Spotify hat die Automobilindustrie dazu gezwungen personalisierte Systeme auch für den Innenraum von Fahrzeugen zu entwickeln, um auch dort den Kundenwunsch nach individualisierbaren und prädiktiven Komfortsystemen zu erfüllen. Die automatisierte Anpassung der Komfortfunktionen im Fahrzeug an die individuellen Bedürfnisse und Gewohnheiten jedes einzelnen Nutzers erfüllt nicht nur den Wunsch nach Individualität, sondern erhöht zudem den Komfort und ermöglicht ein angenehmeres und fokussierteres Fahrerlebnis.

Das Ziel dieser Dissertation ist daher die Anwendung von maschinellen Lernverfahren, um Fahrzeug-Komfortfunktionen in selbstlernende Komfortsysteme umzuwandeln. Diese Systeme sollen in der Lage sein die individuellen Verhaltensmuster des Nutzers – wie zum Beispiel die Aktivierung der Sitzheizung im Winter – zu erlernen und anschließend die entsprechenden Komfortfunktionen selbständig zu steuern, genauso wie der Kunde sie bedienen würde.

Um dieses Ziel zu erreichen, werden die allgemeinen Anforderungen an ein selbstlernendes Komfortsystem im Fahrzeug durch eine Literaturrecherche identifiziert. Unter Berücksichtigung dieser Anforderungen werden eine generische Systemarchitektur und die Beschreibung der wichtigsten Systemkomponenten vorgestellt.

Anschließend wird ein neuartiges Bayesian Nonparametric Modell vorgestellt, welches das Nutzerverhalten bei der Bedienung von Komfortfunktionen im Fahrzeug erlernt. Dieses Modell ist das grundlegende Element eines selbstlernenden Komfortsystems. Zur Bestimmung der Modellparameter wird einen Algorithmus der Variationsinferenz eingeführt. Der Algorithmus erweitert den

Stand der Technik bezüglich Variationsinferenztechniken, indem die Flexibilität und Anpassungsfähigkeit des Modells durch einen sogenannten "truncation-free" Inferenzansatz mit minimaler Ressourcennutzung verbessert wird. "Truncation-free" bedeutet hierbei, dass die Anzahl der Parameter nicht vordefiniert ist.

Um die Umsetzbarkeit des selbstlernenden Komfortsystems unter realen Bedingungen nachzuweisen, wird ein vorläufiger Prototyp eines selbstlernenden Fenstersystems als erste Konzeptdemonstration entwickelt. Die gewonnen Erkenntnisse aus der prototypischen Implementierung werden zur Weiterentwicklung des ursprünglichen Inferenzalgorithmus verwendet. Die Erweiterung des Algorithmus besteht in der Möglichkeit der Einbeziehung von Expertenwissen in das Lernverfahren.

Darüber hinaus werden das Nutzerverhaltensmodell und die Inferenzmethoden untersucht mit synthetisch generierten Daten, die herausfordernde Szenarien abbilden. Abschließend wird auf Basis von Messdaten einer sechsmonatigen Studie eine umfassende Bewertung der Modellgüte durchgeführt.

Zusammenfassend bieten selbstlernende Komfortsysteme einen vielversprechenden Ansatz, um die Kundenwünsche nach personalisierten und intelligenten Fahrzeuginnenräumen zu erfüllen. Das in dieser Arbeit vorgestellte Nutzerverhaltensmodell ist das grundlegende Element für ihre Entwicklung.

Acknowledgment

This thesis is the outcome of a challenging and inspiring journey as a PhD student at Mercedes-Benz AG in collaboration with the Institut für Technik der Informationsverarbeitung (ITIV) of the Karlsruhe Institute of Technology. I could not have reached this goal without the guidance and help of many people.

Firstly, I would like to thank my supervisor Prof. Dr.-Ing. Eric Sax, whose insights and feedback was invaluable to help me refine my thoughts and broaden my perspective. My gratitude extends to Prof. Dr. rer. nat. Cornelius Neumann for being the co-advisor of my work and for the stimulating discussions. Furthermore, I would like to thank the chairman of the examination Prof. Dr.-Ing. Michael Heizmann and the examiners Prof. Dr.-Ing. Maria Francesca Spadea and Prof. Dr.-Ing. Peter Rost for their thoughtful questions during the oral defense.

I would also like to express my gratitude to my supervisor at Mercedes-Benz AG, Dr. Irma Nitsche, for her support and guidance throughout these years, which has made me grow both professionally and personally.

A special thanks goes to my colleagues at Mercedes-Benz and to the group of PhD students of Prof. Sax for the engaging and valuable discussions.

Lastly, I cannot forget my family and friends for their unconditional love and support. I profoundly thank my husband Giovanni for always being there when I needed a word of encouragement.

Stuttgart, July 2024

María

Contents

1 Introduction

1.1 Motivation

Since Gottlieb Daimler and Karl Benz developed the world's first automobile more than 120 years ago, vehicles have greatly improved in performance, safety and comfort. While the early efforts of the automotive industry were concentrated on the search for reliable and portable engines, today's research is focused on the electrification and integration of cutting-edge machine learning technologies into vehicles ([1], [2]). As vehicles transform from traditional modes of transportation into intelligent environments, the potential to revolutionize the driving experience becomes increasingly evident.

The significance of the interior space in vehicles has grown recently, since people are inside their vehicles for a considerable portion of their days [3]. Personalized content in social media platforms and other web applications, such as Netflix ([4]) or Spotify ([5]), are setting the expectation for an individual experiences in vehicles [6]. The desire for personalization is also motivated by the growing popularity of car sharing solutions, which increases the demand for vehicles capable of adapting their interior space to the individual needs of each different customer. Offering passengers a tailor-made in-vehicle experience will not only contribute to increase their satisfaction, but will be a crucial factor to brand differentiation [7]. Furthermore, as higher-level autonomous driving solutions become commonplace, the necessity for intelligent systems within vehicle interiors gains further prominence, as they will be crucial to foster trust and acceptance of autonomous vehicles [8].

In the interior space of vehicles, the systems designed to enhance the comfort and well-being of passengers during travel are known as comfort systems. Traditional automotive comfort systems have been designed with a one-size-fits-all approach, failing to account for the diverse preferences and behaviors of drivers and passengers. Even if some vehicle comfort systems offer different configuration options, they are designed to fit the average user, and are often determined by user studies and expert knowledge, incorporating little to no user-specific information.

Occupants' individual information, like their frequent routes, preferred seat heating stage, or locations at which they typically open the window, can serve as a foundation for automated and personalized comfort features. Although comfort is inherently personal and subject to change over time, humans are creatures of habits and tend to frequently perform the same actions under similar circumstances [9]. For instance, a driver may frequently turn on the seat heater in cold, rainy days. The identification of such individual behavioral patterns is possible taking advantage of the recent advancements in machine learning techniques, which have demonstrated remarkable success in uncovering patterns in data [10]. Automating the in-car comfort functions according to the particular needs and routines of each individual user would not only make customers feel their car is personalized, but also increase their level of comfort and create a more enjoyable and less distracting driving experience.

The main goal of this dissertation is, therefore, to investigate how machine learning techniques can contribute to transform in-vehicle comfort functionalities into intelligent and adaptable systems, capable of learning users' frequent actions in order to seamlessly control the desired vehicle functionality as the users would do. A system with such capabilities is referred to as a self-learning comfort system throughout this dissertation.

1.2 Research Questions

The primary motivation of this dissertation is to evaluate whether machine learning techniques can be used to learn and predict individual user behavioral patterns with vehicle comfort functionalities, in order to develop personalized vehicle comfort systems. To achieve this objective, the following central research questions (RQ) are presented.

RQ 1: *What are the requirements for a self-learning comfort system?*

Once the requirements for a self-learning comfort system have been identified, the next step is to present a conceptual framework or architecture of the system that fulfills these requirements, leading to the next research question.

RQ 2: *How can the architecture and the components of a self-learning comfort system be designed to satisfy the specified requirements?*

At the core of any self-learning system lies a machine learning algorithm capable of identifying individual user behavioral patterns and predicting future user actions. Hence, the third research question that this work aims to answer is:

RQ 3: *How can individual user behavioral patterns with vehicle comfort functionalities be learned and predicted using machine learning algorithms?*

1.3 Thesis Overview

The content roadmap of this dissertation is illustrated in Figure 1.1.

Chapter 1 introduces the concept of self-learning comfort systems in vehicular scenarios. It motivates the research in this area and presents the research questions that this thesis aims to answer.

Chapter 2 provides a review of the theoretical background and related work on self-learning systems and Bayesian learning. Furthermore, the norms, design

guidelines and constraints imposed by the vehicle environment are presented, together with applications of intelligent systems in vehicles.

In Chapter 3, the iterative methodology followed throughout this dissertation and the requirements on self-learning comfort systems in vehicles are introduced. Moreover, the main components needed to fulfil these requirements are defined.

Based on the system's requirements, Chapter 4 presents a Bayesian nonparametric user behavioral model to learn and predict individual behavioral patterns with vehicle comfort functionalities. The design of this user behavioral model constitutes the central element of this dissertation, as it is the fundamental pillar upon which a self-learning comfort system is built. Also, a novel truncation-free variational inference algorithm is presented, which allows the model to dynamically adapt its complexity to each individual user.

Subsequently, Chapter 5 presents a prototypical implementation of a self-learning comfort system, exemplified with a self-learning window. This preliminary study aims to prove the feasibility of the system and provide valuable insights for further development. Based on the results and knowledge gained from the prototype, a second model inference method is introduced. This second algorithm extends the first one by enabling to incorporate expert knowledge into the learning procedure.

The properties and performance of the user behavioral model are further analyzed in Chapter 6, leveraging for this purpose synthetically generated data sets. Firstly, the model's ability to cope with different context circumstances are examined with four datasets, each of which is designed to represent a single challenging situation. Secondly, the influence of the number of user actions and user's feedback on the model's uncertainty of future user actions is examined.

Complementary to the evaluation performed in the previous section with synthetic data, Chapter 7 presents an exploratory demonstration of the user behavioral model. To this purpose, the user behavioral model is employed to extract the behavioral patterns with the power window and seat heater of four different users. The data utilized in this exploratory demonstration was collected from the user's vehicles over a span of 6 months.

Finally, Chapter 8 draws a conclusion and gives an outlook on possible fields of research in the area of self-learning comfort systems and Bayesian nonparametric techniques for modeling user behavior.

1. **Introduction**	• Motivation and research questions
2. **Literature Review**	• Intelligent and self-learning systems • Vehicle environment and applications • Bayesian learning
3. **Derivation of a Self-Learning Comfort System**	• Methodology • System requirements and architecture
4. **A Novel User Behavioral Model**	• Model description • First inference algorithm
5. **Feasibility Prototype of a Self-Learning Comfort System**	• Prototype description and results • Optimized inference algorithm
6. **Analysis of the User Behavioral Model**	• Model analysis with synthetic datasets
7. **Experimental Evaluation of the User Behavioral Model**	• Model evaluation with real data
8. **Conclusion and Outlook**	• Answer to research questions • Future work

Figure 1.1: Content roadmap of the thesis. The thesis original contribution (chapters 3-7) is indicated in light grey.

2 Literature Review

Research in self-learning comfort systems in vehicles is interdisciplinary, and benefits from the integration of ideas, concepts, and methods from various research areas.

2.1 Intelligent and Self-Learning Systems

Intelligent systems are designed to exhibit characteristics associated with human intelligence. They have, at least, three capabilities: perception, reasoning, and action [11]. Perception is achieved thorough sensors or any other input device that enables the system to gather relevant data from the environment, such as cameras, microphones, or sensors of physical quantities. Reasoning involves processing the input data and generating suitable actions to achieve the system's goals. Finally, intelligent systems must be able to execute the actions they decided to take using actuators, which can be physical devices or software-based functions.

According to [11], intelligent systems or agents, are entities that can perceive its environment through sensors and act upon that environment through effectors. They can be classified into five groups based on their level of intelligence and capabilities, as showed in Figure 2.1. The five categories, from lowest to highest degree of intelligence, are: simple reflex, model-based reflex, goal-based, utility-based, and learning agents.

Simple reflex agents act only on the basis of the current percept, ignoring the rest of the percept history. Their function is based on the rule: "if condition, then action".

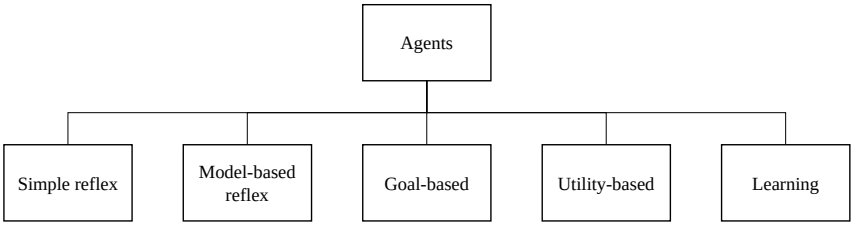


Figure 2.1: Types of intelligent systems, or agents, based on their level of intelligence and capabilities according to [11].

Model-based agents, on the other hand, can handle partially observable environments. Their current state is stored inside the agent maintaining a structure which describes the part of the world which cannot be seen. This knowledge about "how the world works" is called a model of the world.

Taking a step further, goal-based agents build upon the capabilities of model-based agents, by incorporating "goal" information. Goal information describes situations that are desirable. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.

Utility-based agents introduce the concept of assigning desirability measures to states, distinguishing between goal states and non-goal states. They achieve this by employing a utility function that maps states to utility values. Rational utility-based agents select actions that maximize the expected utility of their outcomes, taking into consideration the probabilities and utilities associated with each possible outcome. To do this effectively, these agents must maintain a comprehensive model of their environment, a task that involves research in perception, representation, reasoning, and learning.

Learning agents, on the other hand, possess the remarkable ability to adapt to unknown environments and enhance their performance beyond their initial knowledge. A learning agent comprises several key components: the "critic" describes how well the agent is doing with respect to a fixed performance standard, the "learning element" is responsible for making improvements, the "performance

element" selects external actions, and the "problem generator" suggests actions that lead to new and informative experiences.

Self-learning systems are learning agents capable of learning even when there is no external indication concerning the correctness of the response of the system to the presented data [12]. For an intelligent system to be able to “program itself”, i.e., acquire the knowledge it needs to achieve its goal, the use of the application has to involve a substantial amount of repetitive behavior [13]. The terms “self-learning system” and “autonomous system” are often used interchangeably. The decision to use one term over the other depends on which aspect of the system’s capabilities is being emphasized: the term “self-learning” highlights the system’s capability to acquire knowledge and improve performance over time, while the term “autonomous” stresses the system’s ability to operate independently.

Self-learning systems have been employed in various domains due to their ability to autonomously improve and adapt over time. In the field of network telecommunications, self-learning systems have been employed for misuse and attack detection ([14], [15]). The energy and smart home sectors benefit from self-learning systems in optimizing energy consumption patterns to minimize waste and costs ([16], [17], [18]). In healthcare, self-learning systems aid medical diagnostics by detecting fault-diagnosis [19].

2.1.1 Automation

A fundamental characteristic of self-learning comfort systems is their capacity to control the vehicle comfort systems automatically, based on the user’s individual preferences. Automation does not exist in an all-or-none fashion, but can be implemented at various levels, described in Table 2.1 [20]. A task may be accomplished: (1) manually, with no assistance from the system, (2) by the user with input in the form of recommendations provided by the system, (3) by the system, with the consent of the user required to carry out the action, (4) by the system, to be automatically implemented unless vetoed by the user, or (5) fully automatically, with no user interaction. These levels reflect the extent to which

an intelligent system can operate independently, make decisions, and perform tasks without human intervention. When selecting the level of automation for an intelligent system, several aspects must be considered, such as the difficulty level of the task, the associated risks, and the user acceptance.

		Roles	
	Level of Automation	User	System
1	None	Decide, Act	-
2	Decision Support	Decide, Act	Suggest
3	Consensual	Approve	Decide, Act
4	Monitored	Veto	Decide, Act
5	Full	-	Decide, Act

Table 2.1: Levels of automation. Adapted from [21].

The design of a system with automation capabilities is challenging [22]. The system must be able to answer following questions [23]: when to act, what actions to take, how to perform the actions, and how to learn to improve its behavior. For this purpose, [24] defines nine principles that should guide the system's automatic behavior, which have significally influenced this work:

- **Valuable**: advances the user's interests and tasks, in the user's opinion.
- **Pertinent**: attentive to the current situation.
- **Competent**: within the scope of the agent's abilities and knowledge.
- **Unobtrusive**: not inferring with the user's own activities or attention, without warrant.
- **Transparent**: understandable to the user.
- **Controllable**: exposed to the scrutiny and according to the mandate of the user.

- **Deferent**: gracefully unimposing.
- **Anticipatory**: aware of current and future needs and opportunities.
- **Safe**: minimizes negative consequences, in the user's opinion.

2.1.2 User Acceptance

The analysis and prediction of the user acceptance is crucial, since an intelligent system that aligns with user expectations and is well-accepted is more likely to be successful in achieving its goals and fulfilling the intended purpose. In this regard, several models can be used to assess and predict user acceptance of new technologies. One of the most influential and widely recognized models is the Technology Acceptance Model (TAM), which has undergone various refinements and extensions since it was first introduced [25]. TAM, described in Figure 2.2, postulates that among the many variables that influence system usage, two factors are especially important: the *perceived usefulness* and the *perceived ease of use*. Perceived usefulness refers to the degree to which a person perceives the technology as useful to her everyday life, and it is often the strongest positive predictor of an individual's behavioral intention to use new technology ([26], [27]). The second variable, the perceived ease of use, refers to a user's perception of how effortless a technological device would be to use.

The Automation Acceptance Model (AAM), as described in [28], is an extension of the TAM that aims to provide a more accurate representation of the acceptance of automated systems. AAM takes into account the interactive nature of many automated systems, represented in Figure 2.2 by the dashed-lines, since user experiences with the automated system influence the system usage. AAM remarks on the influence of *task-technology compatibility* and *trust* in the acceptance of automated systems. Compatibility refers to the degree to which the automated system aligns with the user's task requirements, work processes and past experiences. Trust reflects the user's belief in the system's ability to perform its primary functions.

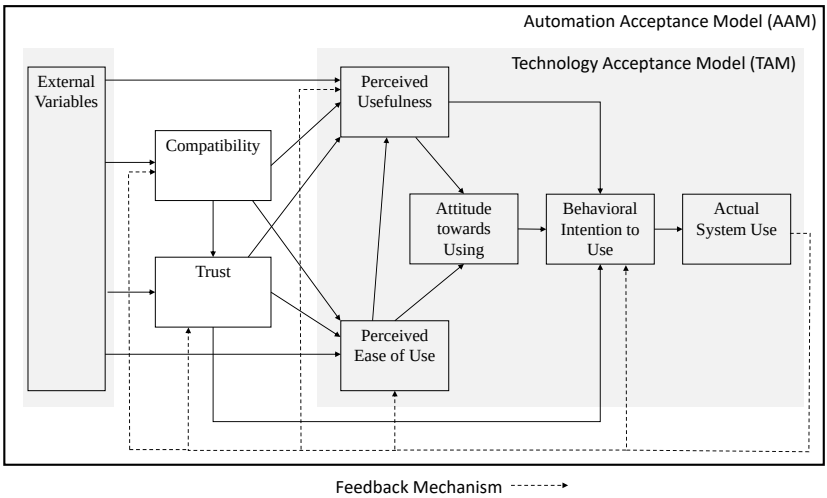


Figure 2.2: Factors that influence user acceptance, according to the TAM and AAM frameworks. Adapted from [28].

TAM, as well as AAM, have proven to be useful in evaluating the acceptance of technologies in several domains, for example health care [29], self-driving cars [30], and in-vehicle functionalities ([31], [32]).

2.1.3 User Modeling under Uncertainty

User modeling techniques play a crucial role in developing intelligent systems that provide personalized and adaptive experiences [33]. It describes the process of creating a user model, which captures the behavior (patterns, goals, interesting topics, etc.) the user shows when interacting with the system, for the purpose of customizing products and services to better suit the user [34]. User modeling involves inferring unobservable information about a user from observable information about her, such as her actions or utterances [35].

Recent research on user modeling has predominantly pursued to describe the user's behavioral patterns and preferences, motivated by the growing demand for

personalized services and information in electronic commerce, social media and news applications [36]. There are two primary approaches to user modeling: stereotype-based and individual-specific models. In stereotype-based user models, users are classified into predefined stereotypes or classes based on common characteristics. For example, users who have similar browsing or purchase histories might be grouped into a class of "frequent buyers". On the other hand, individual-specific user models focus on representing each user uniquely. These models aim to capture the specific preferences, interests, and behaviors of individual users, allowing intelligent systems to tailor their actions precisely to each user's needs. In the context of personalized systems, such a self-learning comfort system, the relevance of individual-specific models becomes evident as they facilitate a higher level of adaptivity.

Situations where the user repeatedly performs a task that involves selecting among several predefined options, are ideal for using machine learning techniques to form a model of the user [36]. In such situations, a model of a user's decision-making process can be created, and it can be used to emulate the user's decisions on future problems. However, it is rather difficult to describe unambiguously the knowledge and decision-making process of a human. The data acquired under these circumstances is often limited, imprecise and incomplete, and human behavior is inherently stochastic and can be observed only partially. Also, any model will be uncertain when predicting unobserved data. Hence, uncertainty plays a fundamental role in user modeling.

According to the machine learning literature, there are two types of uncertainty: epistemic and aleatoric [37], illustrated in Figure 2.3. Epistemic uncertainty, also known as model uncertainty, arises from a lack of knowledge about the true underlying relationship in the data. It is associated with the uncertainty about the model parameters or structure. Epistemic uncertainty can be reduced with more data or by building more complex models that better capture the underlying patterns in the data. On the other hand, aleatoric uncertainty, also known as data uncertainty, arises from the inherent variability or stochasticity in the data-generating process. It is irreducible even with an infinite amount of data and

can be thought of as the "noise" in the observed data. Even though it cannot be eliminated, it can be accounted for in the modeling process.

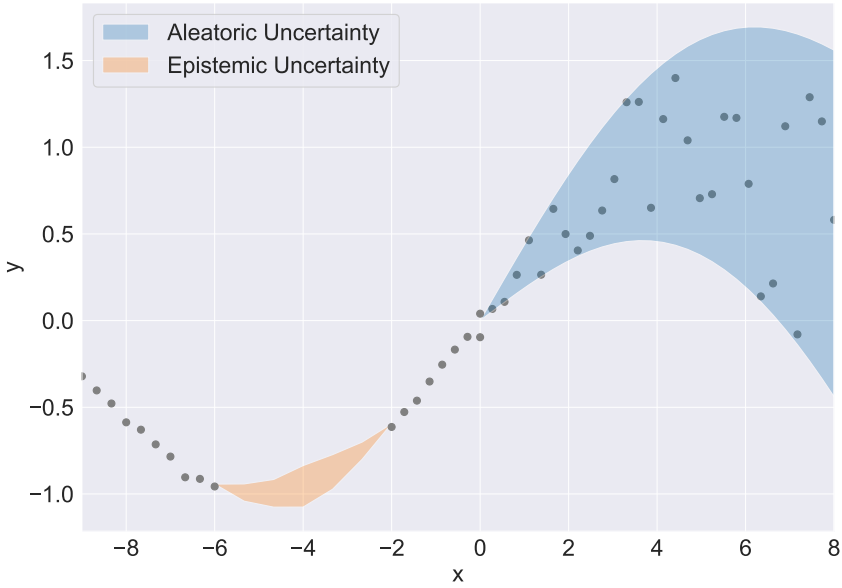


Figure 2.3: Illustration of the difference between aleatoric and epistemic uncertainties. The dots on the plot represent the available data points. Aleatoric uncertainty captures varying degrees of inherent noise in the data, while epistemic uncertainty indicates the ignorance gap due to a lack of data

In scenarios in which control is handed-over to automated systems, it is important to quantify the confidence about the models' predictions, because decisions based upon incorrect predictions can cause significant losses. Understanding if a model is underconfident or falsely over-confident (i.e., its uncertainty estimates are too small) can help preventing unintended behaviour [37].

Traditionally, probabilistic methods have been perceived as the ultimate tool for uncertainty handling [38]. They rely on probability theory to represent, propagate, and analyze all forms of uncertainty [39]. Probabilistic methods are commonly

used with Bayesian inference to model uncertainty in various parameters or variables ([40], [41]). Non-probabilistic methods, on the other hand, rely on modeling uncertainty by considering the variation across repeated trials. Instead of handling uncertainty using explicit probabilistic models, these methods use alternative mathematical frameworks or representations such as intervals ([42]), fuzzy sets ([43]), Credal partition ([44]), or distance-based evidence reasoning mechanisms ([45]) to quantify uncertainty. Classical non-probabilistic approaches often rely on large datasets to converge to a solution. However, when building individual-specific user models, there is usually a relatively small amount of data available for each user. This limitation makes non-probabilistic approaches less suitable than probabilistic methods for building user-specific models under uncertainty [46].

Probabilistic modelling has therefore emerged as one of the principal theoretical and practical approaches for designing machines that learn from data acquired through experience [46]. Probabilistic models have been successfully implemented in many scientific domains, such as transportation, medicine, economics, automated diagnosis, where uncertain factors are encountered and affect the knowledge of situations related to the systems' operations [47]. For example, [48] focused on Bayesian networks to understand driver music genre preferences to make personalized soundtrack recommendations. Also, a Bayesian decision model about driver distraction and aggression is presented in [49], which aims to identify pre-crash conditions. Therefore, probabilistic Bayesian methods are employed in this dissertation to develop a user-specific behavioral model for self-learning comfort applications in vehicles. Bayesian learning methods are introduced in Section 2.3.

2.1.4 Context-Awareness

While user models aim to capture user preferences, behaviors, and characteristics, context-awareness focuses on understanding and utilizing the contextual factors surrounding user interactions, such as location and time, to provide relevant and

timely services. The integration of context-awareness with user modeling enables the development of context-aware user models, which leverage the dynamic adaptation to changing contextual factors, enhancing the adaptability and accuracy of user models.

The most accepted definition of context is “any information that can be used to characterize the situation of an entity”, where an entity can be a person, place or object relevant to the interaction between a user and an application, including the user and the application themselves [50]. The nature of the context information analyzed depends on the specific application domain. Location, identity, time, and activity are the primary context types for characterizing the situation of any entity, as described in [51]. Nevertheless, social context, such as whether the user is alone, as well as the user’s emotional state can also be utilized [52].

In most real-world scenarios, context data are noisy, ambiguous or subject to change. For instance, in autonomous vehicles, unexpected road conditions or sudden obstacles can introduce uncertainty into the context that the vehicle must interpret and respond to. Hence, addressing uncertainty in context requires techniques that go beyond deterministic reasoning, and probabilistic methods offer mechanisms to model and manage uncertainty effectively [46].

Deciding if a context feature is relevant for an application involves evaluating its impact on the application’s performance and its ability to provide valuable information for the problem at hand. One approach is to consult domain experts, whose knowledge can help identify features that are known to have a significant impact on the application’s behavior or performance. The work in [53] suggests that if domain knowledge is at hand, it should be utilized. Another approach is to evaluate the correlation between each context feature and the target variable, so that features that have a strong connection and are likely to be influential are recognized. Lastly, the implementation of cross-validation techniques allows for a comprehensive assessment of how the model performs with different subsets of context features, providing insights into feature stability and generalizability to new data ([54], [55]).

2.1.5 Decision-Making

Decision theory provides a framework for rational decision-making by considering uncertainties, risks, and potential outcomes associated with different options [56]. Combined with probability theory, it enables to make optimal decisions in situations involving uncertainty, such as those encountered in real-work applications by self-learning comfort systems. Decision-making can be seen as an important final step in every user model. User models allow understanding user preferences, but, if the intention is to translate this understanding into an action, it is essential to also consider the potential rewards and risks associated with each estimation.

In decision-making problems, it is required to choose an action from a set of possible actions. The optimal action is determined by maximizing the expected utility, which measures how compatible the action is with the current situation [57]. Equivalently, other authors define the optimal action as the one that minimizes the expected loss. Maximizing the benefit, or minimizing the lost, is the essence of a rational behavior.

In situations where the potential risks or costs associated with making a wrong decision are high, the decision-maker may prefer not to commit to any option. This is known as reject option [10]. By opting not to act, the decision-maker acknowledges the limitations of available information and the potential negative consequences of committing to a decision without a strong basis. For example, on a hypothetical automatic X-ray image classification system, it may be appropriate to automatically classify those image for which there is little doubt as to the correct class, while leaving a human expert to classify the more ambiguous cases. This can be achieved by introducing a threshold T, and rejecting those inputs for which the largest of the posterior probabilities is less or equal to T.

Threshold-based decision-making strategies are particularly useful when dealing with uncertain situations, as they simplify the decision process and provide clear guidelines for action. For example, [58] presents an intelligent calendaring system that uses Bayesian reasoning to make informed decisions. Only if the inferred

probability of users willing a service is above a “do-it” threshold, the system autonomously schedules a meeting on behalf of the user.

2.2 Vehicle Environment

In the early stages of automotive development, vehicles were predominantly mechanical devices. Comfort systems included manual seat and window adjustments, basic ventilation systems and rudimentary heating mechanisms. While these mechanical comfort systems provided some level of convenience, they often lacked the sophistication and versatility needed to fulfill the diverse comfort preferences of passengers.

The advent of electrical innovations brought a significant revolution in vehicles, and electrical systems began to replace or augment various mechanical functions [59]. As presented in Figure 2.4, electronic systems now constitute a significantly larger portion of a car’s overall cost, increasing from approximately 1% of the global car value in 1950 to roughly 35% in 2020. This substantial increase is primarily driven by consumers’ persistent expectations for electronic devices, IT services, connectivity, and the continuous integration of automation in vehicles [60].

Consequently, electrical comfort systems have become commonplace. Some examples include the electrically adjustable seat comfort system, which provides ergonomic and adjustable seat options and may include seat heater, seat cooling or lumbar support; and the power sunroof system, which allows adjusting the sunroof easily. However, they typically rely on manual input and predefined settings, lacking the ability to adapt to individual preferences.

Self-learning comfort systems aim to change this interaction paradigm, by anticipating and responding to the needs and preferences of each user, offering a personalized, adaptive, and user-centric experience. The automotive industry has

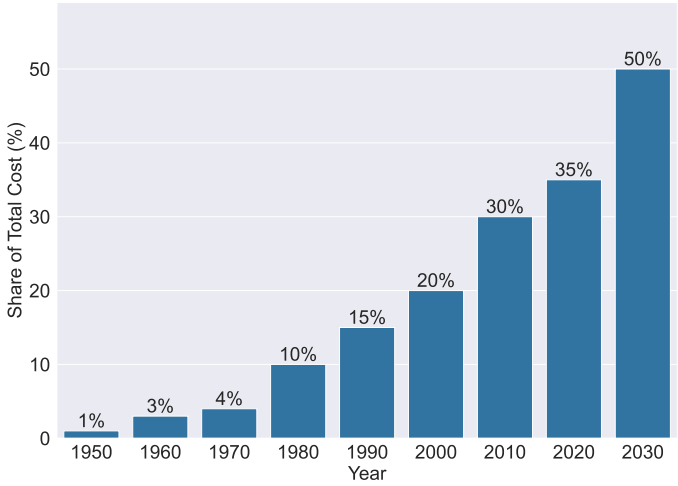


Figure 2.4: The cost of automotive electronics as a percentage of total car costs worldwide from 1950 to 2030. Adapted from [61].

already initiated a process of transformation towards more intelligent and personalized vehicles, even though there is still room for further advancements and improvements, especially in the vehicle comfort domain.

One critical factor that constraints the development of intelligent systems in vehicles compared to other environments, such as data centers or high-performance servers, is that vehicles have limited computational resources. Besides the strict safety and data privacy constraints, presented in the next section, the algorithms must operate efficiently within the resource limitations, encompassing processing power, memory capacity and energy efficiency.

Figure 2.5: Interior of the Mercedes-Maybach S-Class [62].

2.2.1 Norms and Guidelines

Data Privacy and Protection

The European Commission and European Data Protection Board have acknowledged the importance of the protection of privacy as a fundamental condition for a responsible use and exchange of in-vehicle data. In the context of data privacy and protection in vehicles, the General Data Protection Regulation and ePrivacy Directive are relevant ([63], [64]). The privacy principles comprise the three core points of transparency, autonomy, and data security. In order to protect the customers' right to privacy, car manufactures must provide them with clear, meaningful information about the types of information collected and how it is used. Moreover, customer's consent must be obtained before collecting, processing, using, and sharing in-vehicle data. Lastly, vehicle manufacturers must protect customers from misuse of the data required for vehicle communication systems thorough the application of data protection principles. When applying security measures, gateways and firewalls must seal off the security-relevant areas in the

networked vehicle, and data should be encrypted. The vehicles' software and hardware architectures must ensure a high level of technical security.

Safety

Multiple guidelines and norms have been developed by different organizations to ensure that vehicle systems are used safely. The most relevant International Organization for Standardization (ISO) norms when developing in-vehicle intelligent systems are ISO 26262 "Road vehicles — Functional safety" ([65]), ISO 21434 "Road Vehicles — Cybersecurity engineering" ([66]), ISO 9241 "Ergonomics of Human System Interaction" ([67]), and ISO 21448 "Road vehicles — Safety of the intended functionality" ([68]). Even though they are voluntary, many automobile manufacturers have committed to complying with them.

ISO 26262 provides guidelines for the development of safety-critical automotive systems. It focuses on the identification and management of safety risks associated with the system's functionality, and provides a framework for the development of safety-related hardware and software. Complementing this, ISO 21434 addresses the need for cybersecurity in automotive electrical systems. It integrates principles and requirements for cybersecurity throughout the development lifecycle, aiming to protect vehicles against potential cyber threats.

ISO 9241, on the other hand, is a series of standards that provide guidelines for the design and evaluation of user interfaces for interactive systems. They cover a wide range of topics related to human-computer interaction, including ergonomic principles, usability, accessibility, and user-centered design. According to these guidelines and norms, vehicle systems should prioritize simplicity, intuitiveness, and ease of use. User interfaces should be designed with minimal cognitive load, allowing drivers to access the features without excessive interaction. Also, they suggest reducing the need for visual and manual interactions.

The standard ISO 21448, also known as SOTIF, was published recently to address the new safety challenges faced by autonomous vehicles and advanced driver assistance systems [69]. ISO 21448 complements ISO 26262, as illustrated in Figure

2.6. While ISO 26262 covers the malfunctioning behavior of electric/electronic systems, which includes random hardware faults and systematic hardware and software faults, SOTIF activities cover functional insuficiencies of the intended functionality or by reasonable foreseeable misuse by persons. By identifying foreseeable scenarios and implementing mitigation strategies, this standard ensures safety beyond systematic failures, thereby enhancing the overall safety landscape in the automotive sector. For a self-learning comfort system in vehicles, this norm requires ensuring that the intended functions work reliably, and analyzing its behavior in various scenarios to anticipate and prevent any unforeseen risks or hazards that might arise due to the system's learning capabilities.

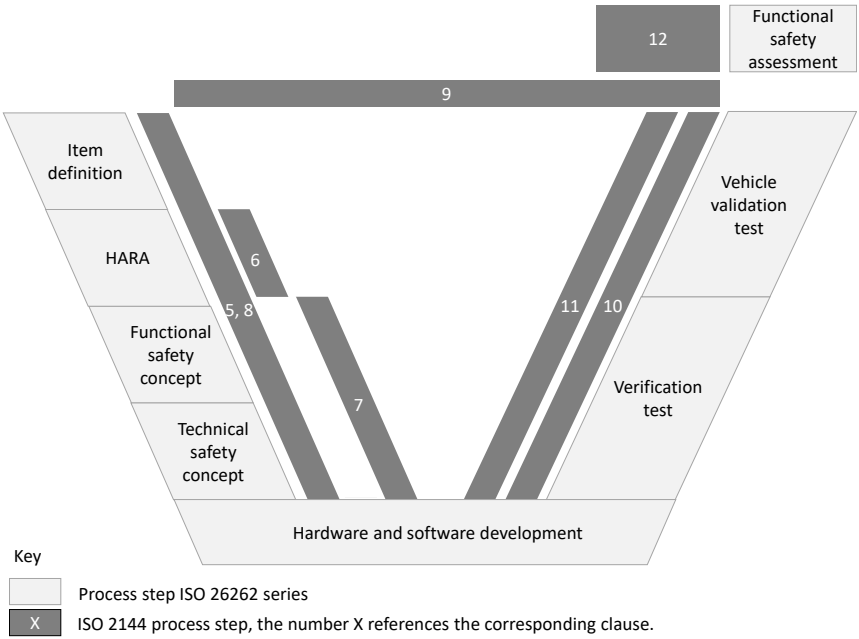


Figure 2.6: Overlap of ISO 26262 and ISO 21448 activities. Adapted from [68]. The title of the ISO 21448 clauses is indicated in Table 2.2.

Clause number	Title
5	Specification and design.
6	Identification and evaluation of hazards.
7	Identification and evaluation of potential functional insufficiencies and potential triggering conditions.
8	Functional modifications addresing SOTIF-related risks.
9	Definition of the verification and validation strategy.
10	Evaluation of known scenarios.
11	Evaluation of unknown scenarios.
12	Evaluation of the achievement of the SOTIF.

Table 2.2: ISO 21448 clauses [68].

2.2.2 Driver Distraction

The research on driver distraction has experienced significant growth, largely motivated by its close relationship with the field of adaptive infotainment systems. Driver distraction occurs "when a driver is delayed in the recognition of information needed to safely accomplish the driving task because some event, activity, object or person within or outside the vehicle compelled or tended to induce the driver's shifting attention away from the driving task" [70]. Distracted drivers are more likely to be involved in traffic accidents and exhibit poorer driving performance compared to focused drivers [71].

Distractions can be grouped into three main categories, based on the sources that causes them [72]. Firstly, visual distractions, such as looking at mobile devices, divert the driver's eyes from the road. Secondly, cognitive distractions, like engaging in a conversation, occupy the driver's attention. Finally, manual distractions, such as adjusting radio stations, require the driver's hands to be off the steering wheel. Even though each of these distractions can impair driver performance and increase the risk of accidents, they do not affect the driving

performance equally. Studies have shown that visual and manual distractions impact the driver performance more negatively than cognitive distractions [73].

The design of in-vehicles systems and their associated user interfaces plays an important role to mitigate driver distractions [74]. Unintuitive user interfaces, excessive information displays, or overly demanding interactions can draw the driver's attention away from the road and compromise her concentration. Therefore, their design must complain with the safety guidelines.

2.2.3 Adaptive Infotainment Systems

Systems that adapt their behavior or presentation to the environment, user or task are known as adaptive systems [75]. Adaptive systems share some similarities with self-learning systems, as they both can change or improve based on its experiences or interactions with the environment. While self-learning systems are a subset of adaptive systems, not all adaptive systems necessarily involve learning. For example, a wiper washer system that starts to function when it detects rain based on pre-programmed rules is adaptable, but not self-learning. Hence, for this work, only in-vehicle adaptive systems with learning capabilities are relevant. Specially, those which encode their understanding of the user in a user model that represents an estimate of the characteristics of the user relevant to the particular application.

In the vehicular context, adaptive systems have been mainly used for enhancing the user interfaces for infotainment functions. Adaptive interfaces have become key elements in vehicles because they give the driver quicker access to the information she needs or wants, in a more appropriate form, which reduces the chances for distraction [76]. According to [76], a critical property of in-vehicle adaptive systems is building the user models unobtrusively, requiring neither explicit setting of preferences nor rating of the system's performance. They suggest that user models must be built with implicit feedback through normal interaction, since drivers have no attention to spare.

Adaptive infotainment systems typically support the driver either by filtering information or by giving recommendations. To the first group of adaptive systems, those that filter information to reduce the workload of drivers, belongs the system described by [77], which filters information according to situational requirements. For example, phone calls are automatically redirected to the voice mailbox when the workload level of the driver exceeds a certain threshold, so the driver can direct full attention to the driving situation. The work in [78] describes an adaptive system which recognizes the driver's affective state, such as emotions or sleepiness, based on an individual user model. The system's goal is to provide feedback about driver state to the driver and take actions to influence the driver state positively, such as by changing the radio station, rolling down the window or splashing some water on her face.

Adaptive systems that support the driver by providing recommendations with relevant content at the right moment, simplifying the interaction, belong to the second group. In this context, recommender systems can be viewed as a subtype of adaptive interfaces. The system presented in [79], for instance, gives proactive recommendations for fuel stations depending on the context, such as when the fuel tank is nearly empty, which aims to reduce the interactions. The system further provided explanations about why a certain recommendation was given, to enhance the system's transparency and hence, its acceptance ([80], [81]). The research in [82] presents an adaptive radio station system, which uses knowledge about the recurrent preferences of the user to make user-specific recommendations. It evaluates two different levels of adaptivity or automation in the interactions with users. In the first approach, users can choose a radio station from a list of recommendations, whereas in the second approach, the system automatically changes the radio station, but users have the possibility to abort the execution within the first seconds over a dialog. Results of both approaches confirmed the positive effects on the users' workload, which suggests that higher level of automations in adaptive vehicle systems are promising in recurrent, routine situations.

Aligned with these results, studies indicate that higher levels of automation lead to reduced driver distraction levels and enhanced user experience. For instance, [83] studies the advantages of fully automated telematic systems in situations in which

only routine tasks are performed. It suggests that they are more advantageous than systems with intermediate levels of adaptivity. Also, the research work in [31] on vehicle recommender systems concludes that the information should be delivered without any request from the user, in an understandable and intuitive manner to minimize the cognitive load of drivers. Furthermore, [84] studied the performance of vehicle infotainment systems with no user interaction in terms of driver distraction and user experience, and concluded that they performed better than vehicle infotainment systems featuring lower levels of interaction. The study conducted by [32] focus on smart vehicle comfort functionalities. It evaluates their user acceptance in recurrent and routine scenarios. Even though results indicate a considerable level of acceptance for smart automated comfort functions in routine situations, smart automated comfort system that allowed users to abort the automation received significantly higher acceptance scores compared to those that directly executed the automation. Therefore, results indicate that safety-relevant vehicle functions shall be highly automated, rather than fully automated.

2.3 Bayesian Learning

Bayesian learning is a machine learning framework which applies probability theory to learn from data, enabling to handle uncertainty in models, make predictions, and adapt beliefs in light of new data [46]. In the context of Bayesian learning, a model refers to a mathematical representation of a system, process, or phenomenon in a given problem, that captures the assumptions and beliefs about how data is generated or how the different variables are related to each other. Unlike traditional frequentist approaches that rely on fixed parameters, Bayesian learning embraces uncertainty by treating model parameters as random variables, allowing to quantify the confidence in various hypotheses. This makes it particularly well-suited for addressing problems where data is limited, noisy, or incomplete. For example, Bayesian learning has been employed in areas like personalized medicine ([85], [86]) and recommendation systems ([87], [88]), where

there is a relatively small amount of data for each patient or client, and the predictions must be customized for each person, becoming necessary to build a model for each person, with its inherent uncertainties.

Learning is achieved through two key components: the prior distribution and the likelihood function. The prior distribution captures initial beliefs about the parameters of a model before observing any data, while the likelihood function expresses the probability of observing the data given the model parameters. By combining these components, Bayes' theorem yields the posterior distribution, which represents the updated beliefs about the parameters after incorporating the observed data, as showed in Figure 2.7.

Bayes' theorem, also known as Bayes' rule, is defined as follows:

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta) \cdot P(\theta)}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\theta)P(\theta)}{\int_\theta P(\mathcal{D}|\theta)P(\theta)\mathrm{d}\theta} \tag{2.1}$$

where $P(\theta|\mathcal{D})$ represents the posterior distribution over parameters θ given data $\mathcal{D}$, $P(\mathcal{D}|\theta)$ is the likelihood of observing the data given the parameters θ, $P(\theta)$ denotes the prior distribution representing initial believes about θ, and $P(\mathcal{D})$ is the marginal likelihood, also referred to as the model evidence, which acts as a normalizing constant.

In cases where the prior distribution and the likelihood function are chosen from the same family of probability distributions, the resulting posterior distribution will also belong to that same family. This family of prior and likelihood distributions is referred to as a conjugate family, and the prior distribution is referred to as a conjugate prior for the given likelihood function. The benefit of using conjugate distributions lies in the simplification of the calculations involved in updating probabilities. When the prior and likelihood are conjugate, the resulting posterior distribution can be obtained analytically, often leading to closed-form expressions, which can significantly speed up computations and facilitate more intuitive interpretation of the results. Otherwise, it may be necessary to approximate the value of the denominator on Bayes' rule, which typically involves marginalising all the variables in the model except for the variables of interest. Such high-dimensional

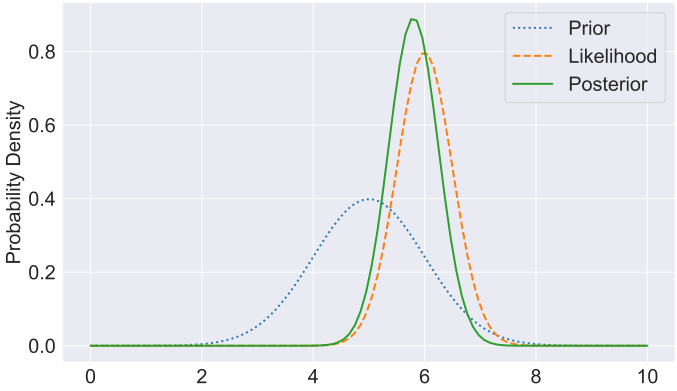
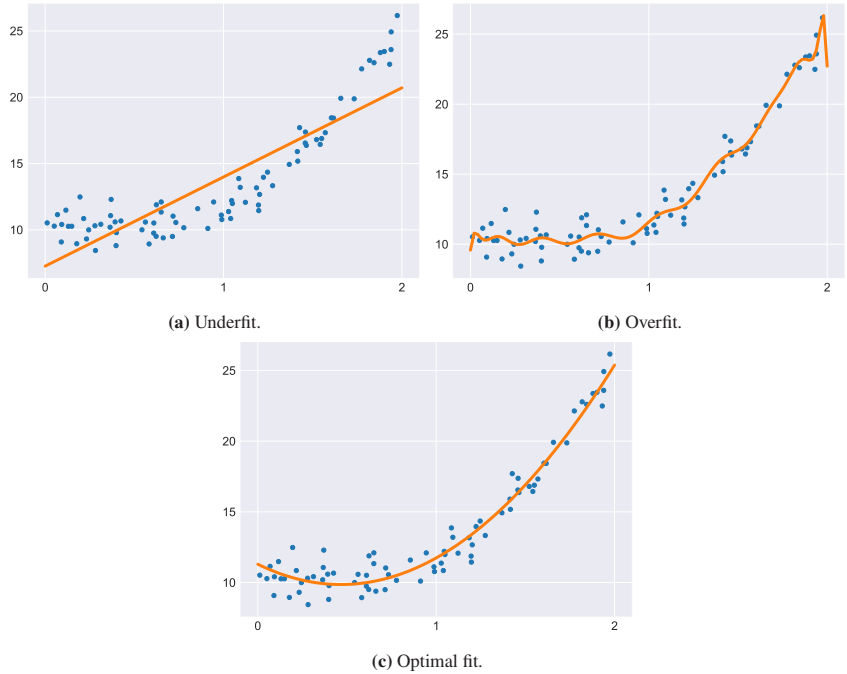


Figure 2.7: Bayesian updating of the prior distribution to posterior distribution. The posterior distribution is a compromise between the information brought by the prior and the information brought by the likelihood.

sums and integrals are generally computationally demanding, in the sense that for many models here is no known polynomial-time algorithm for performing them exactly. Therefore, it is often necessary to approximate the posterior distribution. Methods for approximate inference are introduced in Section 2.3.2.

Bayesian models that use a fixed and finite number of parameters to capture the relevant information in the data necessary to make predictions are known as parametric models. The process of determining the optimal number of parameters is generally time-consuming and may not be scalable to large-scale unfamiliar data. This is due to the fact that it typically involves human labor or restarting the algorithm several times to find the optimal settings. Moreover, parametric models are prone to suffering from overfitting or underfitting if there is a misfit between the complexity of the model and the amount of data available [89]. Hence, parametric models are often not well-suited for applications where the model must be flexible and grow in complexity as more data is collected, such as for capturing the diverse and evolving nature of user preferences in comfort systems. In such cases, nonparametric Bayesian models offer a better approach. The concepts of underfitting, optimal fitting, and overfitting, are shown in Figure 2.8.



(a) Underfit.

(b) Overfit.

(c) Optimal fit.

Figure 2.8: Illustration of the underfitting/overfitting dilemma on a simple regression case. Data points are shown as blue dots and model fits as orange lines. Underfitting occurs with a linear model (top-left panel), an optimal fit with a polynomial of degree 2 (botton panel), and overfitting with polynomial of degree 20 (top-right panel).

2.3.1 Bayesian Nonparametric Models

Bayesian nonparametric (BNP) models extend the Bayesian framework to scenarios where the number of underlying parameters is unknown or could potentially grow as more data is observed. They offer an alternative to traditional parametric models by assuming an infinite-dimensional parameter space, effectively having infinitely many parameters. This allows for greater flexibility in capturing the inherent complexity of real-world data without the need for pre-specified model structures. They are useful to find out the latent causes and structures behind data, and appear as an alternative to model selection. Model selection is one of

the main concerns within the machine learning community and is often related to issues such as overfitting and underfitting, as described in [90]. By using a model with unbounded complexity, underfitting can be mitigated, while the Bayesian approach of computing or approximating the full posterior over parameters can help to mitigate overfitting. BNP models have been used to solve problems such as face recognition [91], clustering gene expression patterns ([92], [93]), speech recognition [94], and modelling the topics of documents [95].

The central idea behind BNP methods is the replacement of the classical finite-dimensional prior distribution with a general stochastic process, allowing an open-ended number of degrees of freedom in a model [96]. A review of Bayesian nonparametric models is outside the scope of this dissertation, so only the key models for this work are mentioned. For further insights, references [90], [97] and [39] are recommended.

2.3.1.1 Dirichlet Processes

The Dirichlet process (DP), first formalized in [96], is a stochastic process whose samples are probability distributions. Hence, a Dirichlet process is often defined as a "distribution over distributions". It is a fundamental tool in Bayesian nonparametric modeling, particularly for clustering and mixture models [98].

Dirichlet processes are defined by a positive real number α, called the concentration parameter, and a base distribution G_0 over a measurable space Θ. According to [96], a random distribution G is distributed according to a DP if its marginal distributions are Dirichlet distributed. More formally, given a measurable set S, a base probability distribution H, and a positive real number α, the Dirichlet process $DP(H, \alpha)$ is a stochastic process whose realization is a probability distribution over S. For any measurable finite partition of S, denoted $\{A_i\}_{i=1}^{n}$, the following holds:

$$\begin{aligned} &\text{if } G \sim \text{DP}(G_0, \alpha) \\ &\text{then } (G(A_1), ..., G(A_K)) \sim \text{Dir}(\alpha G_0(A_i), ..., \alpha G_0(A_K)). \end{aligned} \tag{2.2}$$

In the above equation, *Dir(·)* denotes the Dirichlet distribution. The density function of a Dirichlet distribution is:

$$p(\pi_1, ..., \pi_k|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \tag{2.3}$$

where $\Gamma(\cdot)$ is the gamma function.

The parameters G_0 and α play intuitive roles in the definition of the Dirichlet process. The base distribution, G_0, is the expected value of the process: the DP draws distributions "around" the base distribution the way a Normal distribution draws real numbers around its mean. The concentration parameter, α, can be understood as an inverse variance: the larger α is, the smaller the variance and thus, the DP will concentrate more of its mass around the mean. Figure 2.9 shows different draws from the Dirichlet process DP($\mathcal{N}(0, 1), \alpha$) for different values of α. Draws from a DP are discrete with probability one [96].

Since the formal definition of Dirichlet process is very abstract, it is frequent to use metaphors to provide a more intuitive understanding of how to generate samples from a DP. Among all the equivalent views, the most relevant ones are the Pólya urn scheme, the chinese restaurant process and the stick-breaking process [99].

The chinese restaurant process (CRP) is illustrated in Figure 2.10. The metaphor describes a chinese restaurant with an infinite number of circular tables, each with infinite capacity. As new customers enter, they can decide weather to seat at an occupied table or at an empty one. Customers will sit at an occupied table with a probability proportional to the number of customers already seated there, or at an unoccupied table with a probability proportional to the concentration parameter α. After infinitely many customers entered, one obtains a probability distribution over infinitely many tables to be chosen. This probability distribution over the tables is a random sample of the probabilities drawn from a Dirichlet process with scaling parameter α.

An impotant characteristic of the DP is that it exhibits a self-reinforcing property. The more often a given value has been sampled in the past, the more likely it

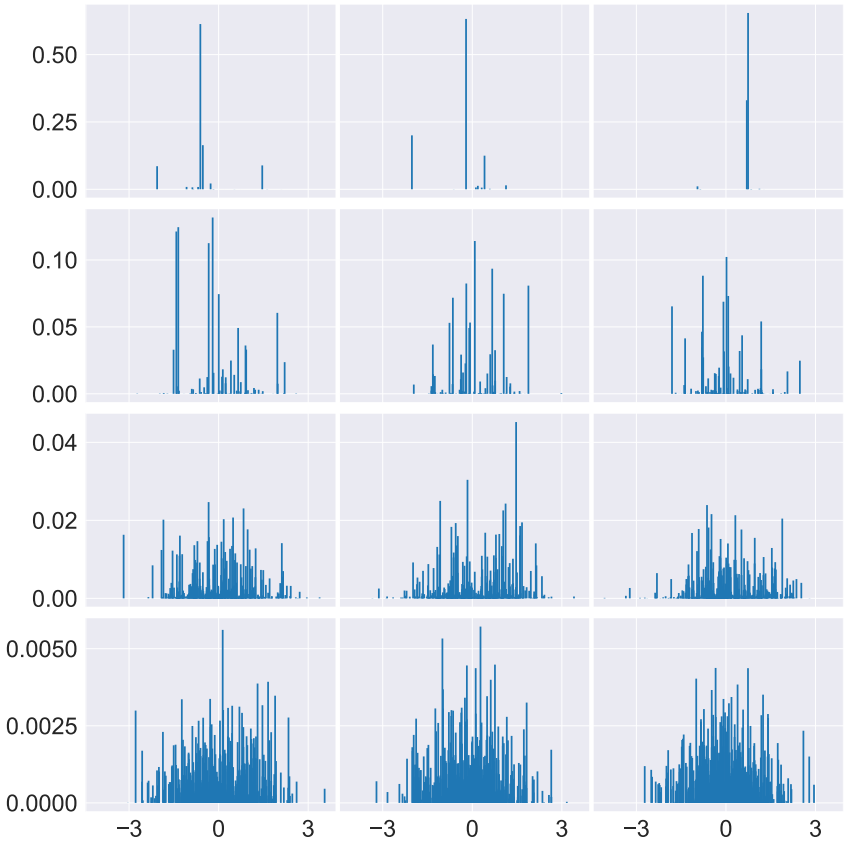


Figure 2.9: Draws from the Dirichlet process DP($N(0, 1), \alpha$). The four rows use different α. Top to bottom: $\alpha = 1, 10, 100, 1000$. Each row contains three repetitions of the same experiment. As seen from the graphs, draws from a Dirichlet process are discrete distributions. They become less concentrated (more spread out) with increasing α.

is to be sampled again, in a so-called "rich get richer" fashion. This property can be explained using the CRP metaphor: since customers sit at a table with a probability proportional to the number of customers already sitting at it, they are more likely to sit at a table with many customers than few.

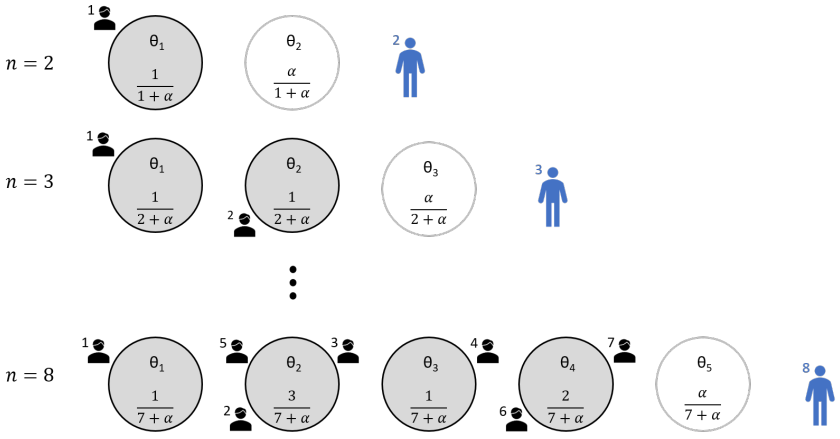


Figure 2.10: Illustration of the chinese restaurant process (CRP). Tables are represented as circles. Each table has a dish, represented as θ, which are draws from the base measure G_0. The probability of a new customer (in blue) sitting on each table is shown inside each table. New customers will sit at an occupied table (gray) with a probability proportional to the number of customers already seated there or at an unoccupied table (white) with a probability proportional to the concentration parameter α. The resulting distribution over tables is a random sample of a Dirichlet process $DP(\alpha, G_0)$.

2.3.1.2 Dirichlet Process Mixture Models

A Dirichlet process mixture (DPM) model assumes that, given a set of observations $\{x_1, x_2, ..., x_N\}$, each observation x_n is independently drawn from a distribution $F(\theta_n)$, such that:

$$\begin{aligned} x_n &\sim F(\theta_n) \\ \theta_n &\sim G \\ G &\sim \mathrm{DP}(\alpha, G_0), \end{aligned} \tag{2.4}$$

where $\mathrm{DP}(\alpha, G_0)$ is the Dirichlet process with concentration parameter α and base measure G_0. Since the function G, the DP, is discrete, different θ_n ($n \in$

$\{1, ..., N\}$) can take simultaneously the same value. Hence, the model above can be seen as a mixture model where the observations x_n with the same parameter θ_n belong to the same component or cluster. In this context, "components" represent different patterns or groups within the data.

The graphical representation of a generic DPM model is presented in Figure 2.11.

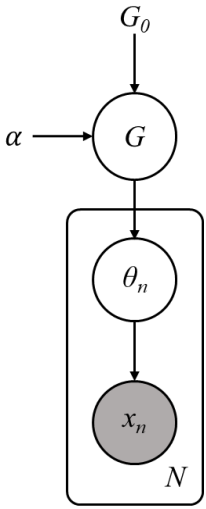


Figure 2.11: A generic Dirichlet process mixture (DPM) model. Shaded nodes represent observed variables, unshaded nodes represent latent variables, arrows denote dependence, and the rectangle is plate notation, denoting the number N copies of the outlined structure.

The Dirichlet process mixture is also referred to as an “infinite” mixture model because it can accomodate a potentially limitless number of components, as new data can introduce previously unseen components [100]. The fact that there is no limit to the number of distinct components which may be generated, bypasses the need to determine the “correct” number of components in a mixture model. In many context, a countably infinite mixture is often a more realistic model than a mixture with a small number of components. This makes Dirichlet process

mixtures the most popular approach for clustering data when the number of clusters is not well-defined in advance [98].

2.3.2 Variational Inference (VI)

Despite the simplicity of Bayes' rule, the high-dimensional integrals in its denominator are in most cases either impossible or very difficult to compute in closed form (see Equation 2.1). Thus, the main effort in Bayesian inference is concentrated on techniques that allow to bypass or approximately evaluate this term.

Bayesian inference methods can be classified into two broad categories: sampling methods and deterministic approximations. Sampling methods, such as Markov chain Monte Carlo (MCMC) techniques, involve a stochastic process where Markov chains are used to explore the parameter space by sampling successive points from the true posterior distribution. These samples are used to estimate the characteristics of the posterior distribution, such as the mean or variance. MCMC methods are known for their flexibility and ability to handle intricate and high-dimensional models. However, they can be computationally expensive and require careful tuning to ensure convergence and accurate results [101]. Due to the limitations of MCMC methods, the inference techniques employed throughout this thesis belong to the second category of inference techniques, the deterministic approaches, which formulates the problem as an optimization task rather than sampling.

The most widely-used deterministic method is variational inference (VI). VI methods seek to approximate the true posterior distribution with a simpler, parameterized distribution chosen from a predefined family, known as the "variational family" [10]. This process is guided by the principle of minimizing a divergence measure between the approximate distribution and the true posterior. VI methods are generally easier to implement and offer faster approximations with scalability advantages than MCMC, but they introduce approximation errors due to simplifying assumptions, as they truncate the model or the variational distribution to a

maximum model complexity. Figure 2.12 shows the intuition behind MCMC and variational inference methods.

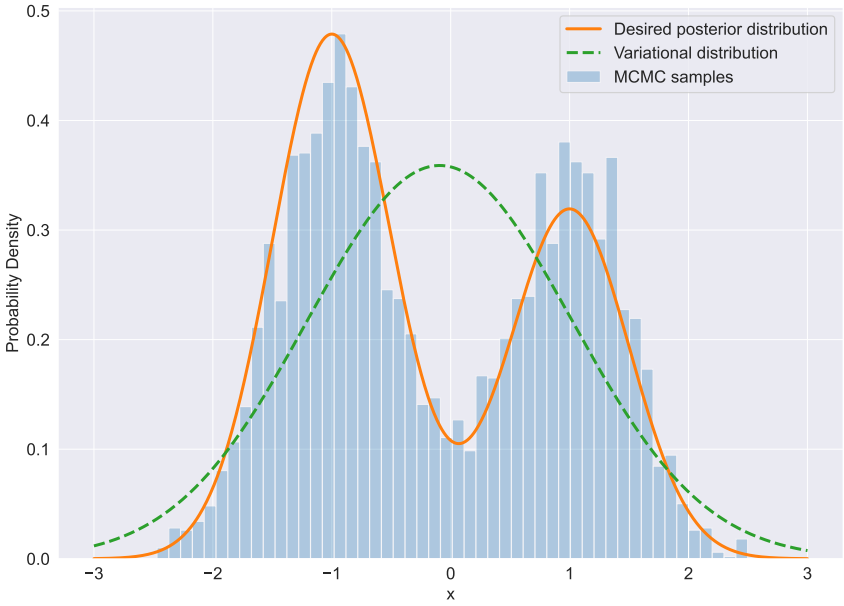


Figure 2.12: Illustration and comparison of MCMC and variational inference algorithms. While MCMC methods estimate the desired posterior distribution by generating samples, VI methods optimize a simpler, parameterized distribution, known as the variational distribution, to approximate the target distribution.

In a general problem of Bayesian inference in which all observed variables are denoted by X, and all hidden variables are indicated by Z, which includes the model parameters θ and latent variables, the goal of VI methods is to firstly, propose a family of distributions $\mathcal{Q}$ over the latent random variables, and secondly, find the family member $q^*(Z) \in \mathcal{Q}$ that is closer to the true but intractable posterior distribution $p(Z|X)$, where closeness is measured using the Kullback-Leibler (KL) divergence:

$$q^*(Z) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(Z)||p(Z|X)), \tag{2.5}$$

The KL divergence is defined as:

$$\mathrm{KL}(q(Z)||p(Z|X)) = \int q(Z) \log \frac{q(Z)}{p(Z|X)} \, dZ \tag{2.6}$$

Directly minimizing the KL divergence to find the best-fitting approximation $q^*(Z)$ requires computing the intractable true posterior, $p(Z|X)$. However, by leveraging the fact that $\log p(Z|X)$ does not depend on $q(Z)$ and using the definition of conditional probability, the terms can be rearranged as:

$$\mathrm{KL}(q(Z)||p(Z|X)) = -\mathcal{L}(q) + \log p(X) \tag{2.7}$$

where $\mathcal{L}(q)$ is the quantity known as the Evidence Lower Bound (ELBO), defined as:

$$\mathcal{L}(q) = \int q(Z) \log \frac{p(X, Z)}{q(Z)} \, dZ = \mathbb{E}_q[\log p(X, Z) - \log q(Z)] \tag{2.8}$$

Hence, maximizing the ELBO with respect to $q(Z)$ is equivalent to minimizing the KL divergence between the variational distribution and the true posterior. This decomposition is illustrated in Figure 2.13.

To complete the specification of the optimization problem of variational inference methods, it is necessary to indicate the family of distributions $\mathcal{Q}$ used to derive the best approximation of the posterior distribution. One of the most popular families of probability distributions is called mean-field variational family, which assumes that the latent variables are mutually independent and can be factorized into

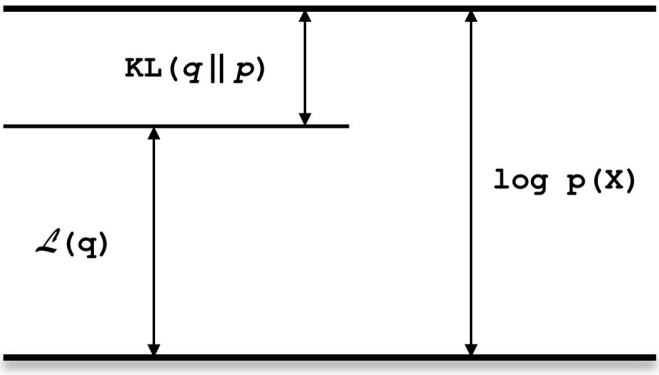


Figure 2.13: Illustration of the decomposition given by Equation 2.7, which holds for any choice of distribution $q(Z)$. Because the KL divergence satisfies $\mathrm{KL}(q(Z)||p(Z|X)) \geqslant 0$, the quantity $\mathcal{L}(q)$ is a lower bound on the log likelihood function $\log p(X)$. Adapted from [10].

independent components [101]. A generic member of the mean-field variational family is given by

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i). \tag{2.9}$$

Among all distributions $q(Z)$ having the mean-field form, the one for which the lower bound $\mathcal{L}$ is the largest must satisfy the following condition for each factor j [10]:

$$\log q_j^*(Z_j) = \mathbb{E}_{i \neq j}[\log p(X, Z)] + \text{const}, \tag{2.10}$$

where the notation $\mathbb{E}_{i \neq j}[...]$ denotes an expectation with respect to the q distributions over all variables z_i for $i \neq j$ so that:

$$\mathbb{E}_{i \neq j}[\log p(X, Z)] = \int \log p(X, Z) \prod_{i \neq j} q_i(Z_i) dZ_i. \tag{2.11}$$

Equivalently, taking the exponential of both sides in 2.10, it results that:

$$q_j^*(Z_j) \propto \exp \mathbb{E}_{i \neq j}[\log p(X, Z)]. \tag{2.12}$$

Equation 2.10 indicates that the logarithm of the optimal solution for factor q_j is obtained by considering the log of the joint distribution over all hidden and visible variables, and then taking the expectation with respect to the other factors q_i for $i \neq j$. It is therefore possible to find the optimal solution by first initializing all factors $q_i(Z_i)$ appropriately, and then cycle through them, replacing each in turn with a revised estimate given by the right-hand side of Equation 2.10, evaluated using the current estimates for all of the other factors. This iterative updating algorithm used to refine the variational parameters when the variational family is mean-field is known as Coordinate Ascent Variational Inference (CAVI), which is presented in Algorithm 1 [101].

The CAVI algorithm provides a powerful and flexible framework for performing approximate Bayesian inference [102]. Figure 2.14 illustrates the intuition behind CAVI, showing several iterations of CAVI for fitting five Gaussian components to two-dimensional data. At the beginning of the CAVI algorithm, the variational density of the mixture components is initialized to be a Gaussian distribution that is nearly centered and has a wide variance. As the algorithm progresses, the variational density is updated iteratively to better fit the observed data. With each iteration, the variational density "moves" to better align with the data, as the algorithm adjusts the parameters of the mixture components to better capture the underlying structure of the data. This process continues until convergence, at which point the variational density provides an approximation to the true posterior distribution over the model parameters and latent variables.

In Bayesian nonparametric models, such as the Dirichlet process mixture model presented in Section 2.3.1.2, the optimal approximation of the posterior distribution is obtained by iteratively usign the coordinate ascent update equations for the model parameters and latent variables defined in Equations 2.13 and 2.14, respectively. The expectations are taken with respect to the variational density

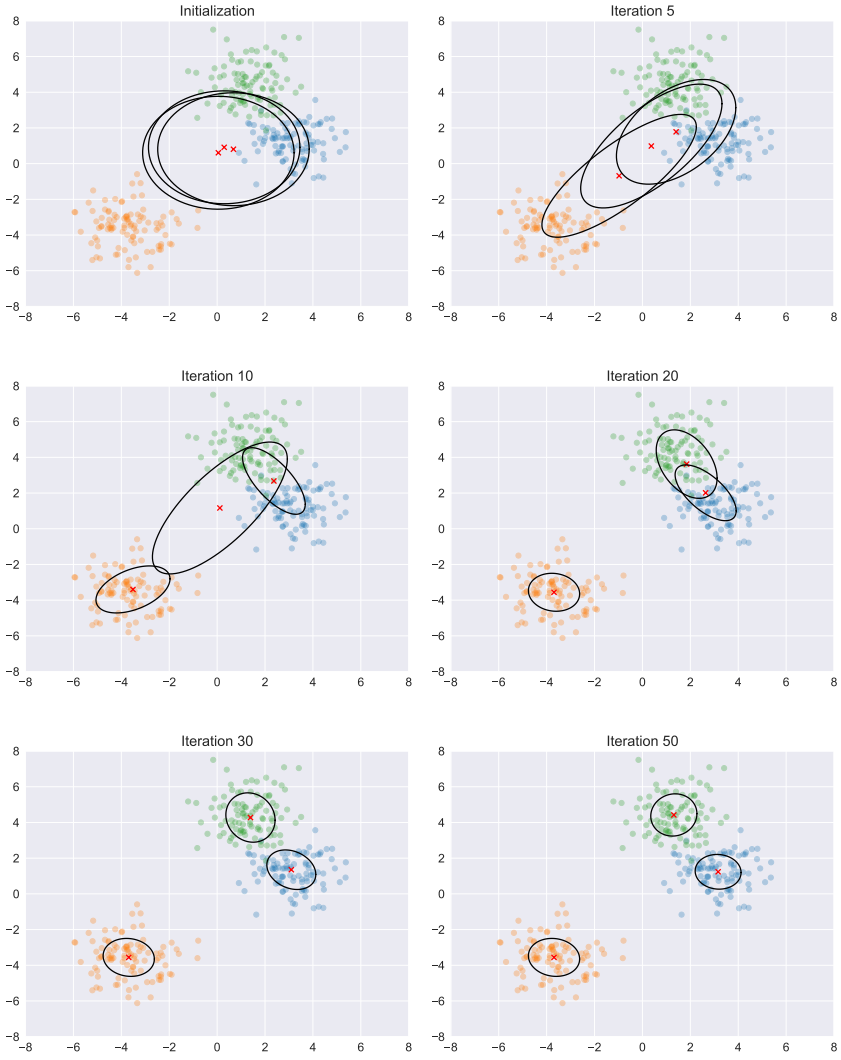


Figure 2.14: Example of the application of the CAVI algorithm on a two-dimensional Gaussian mixture model. The ellipses represent the 2-sigma contours of the variational approximating factors. The sub-plots illustrate the progression if the variational density of the mixture components as the CAVI algorithm iteratively refines the variational parameters.

Algorithm 1: Coordinate Ascent Variational Inference (CAVI)

Input: A model $p(X, Z)$, a data set X
Output: A variational density $q(Z) = \prod_{i=1}^{M} q_i(Z_i)$
Initialize: Variational factors $q_i(Z_i)$
while *ELBO has not converged* **do**
  for $i \in \{1, ..., M\}$ **do**
    Set $q_j(Z_j) \propto \exp\{\mathbb{E}_{i \neq j}[\log p(X, Z)]\}$;
  end
  Compute ELBO$(q) = \mathbb{E}[\log p(Z, X)] - \mathbb{E}[\log q(Z)]$;
end
Return: $q(Z)$

$q(\cdot)$ for all of the other variables, i.e., $q(Z, \Theta_{\not k})$ for 2.13 and $q(Z_{\not n}, \Theta)$ for 2.14, which permits the iterative calculation.

$$\log q^*(\theta_k) = \log p(\theta_k) + \sum_{n=1}^{N} \mathbb{E}[\log p(x_n|\theta_k)] + \text{const} \tag{2.13}$$

$$\log q^*(z_n) = \log p(z_n|Z_{\not n}) + \mathbb{E}[\log p(x_n|\theta_{z_n})] + \text{const} \tag{2.14}$$

Appendix C provides a more comprehensive and detailed mathematical derivation of these equations, as they are crucial for the inference of the user behavioral model presented in this work.

3 Derivation of a Self-Learning Comfort System

3.1 Methodology

The methodology utilized to derivate a self-learning comfort system is inspired by the principles of DevOps ([103]) and agile frameworks ([104]), which emphasize iterative development and continuous refinement. The agile methodology focuses on collaboration, rapid software releases and customer feedback, whereas DevOps is a collaborative approach that integrates software development (Dev) with operations (Ops) to improve the efficiency, quality and speed of software delivery. A typical DevOps lifecycle is illustrated in Figure 3.1.

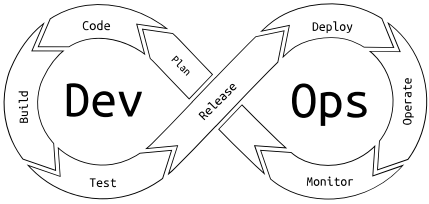


Figure 3.1: A typical DevOps lifecycle [105]. It represents a continuous and iterative process encompassing planning, coding, building, testing, deployment, operation, monitoring, and optimization stages.

The methodology followed in this dissertation focuses on the iterative execution of planning, coding, deployment, evaluation and optimization activities. In total, two iterative cycles are performed for the derivation a self-learnig comfort system. The first iteration focuses on the developing a flexible user behavioral model and a

truncation-free variational inference algorithm to infer the model's parameters. As presented in Section 3.3, the user behavioral model is the fundamental element of a self-learning system and the main contribution of this work. The user behavioral model and the inference algorithm are subsequently evaluated by means of a feasibility prototype (see Chapter 5).

Based on the feedback and results obtained during the prototype evaluation, the inference algorithm is refined and optimized by means of a heuristic thresholding mechanism, introduced in Section 5.4. Ultimately, this second approach is analyzed and evaluated in Chapter 6 and Chapter 7, respectively.

3.2 Requirements

The requirements on a self-learning comfort system are derived after a comprehensive literature review of the background information outlined in the preceding chapters and expert evaluation.

REQ1 Continuous Learning.

A self-learning comfort system should be able to identify user frequently performed actions with vehicle comfort systems. Since the users' preferences are likely to change over time, it should be capable of adapting its knowledge quickly, letting new information refine what has already been learned [106]. A self-learning comfort system should improve its performance over time by taking into account the user's direct and indirect feedback [13].

REQ2 Context-Awareness.

A self-learning comfort system should possess the ability to understand and utilize contextual information [107]. It should consider user preferences, past interactions, location, time and other relevant factors to provide personalized and appropriate assistance. Data should be retrieved from vehicle sensors, cameras, displays and any other source of information available.

REQ3 Decision-Making.

A self-learning comfort system should be capable of logical reasoning and decision-making. It should be able to handle the inherent uncertainty in real-word scenarios due to incomplete or noisy data, varying conditions and unpredictable events. Wrong decisions and unexpected actions might not only negatively influence the user's trust and acceptance of the system, but also cause driver distraction. Therefore, the system should decide to automatically control a vehicle functionality only in situations where it has very high confidence on its predictions. Moreover, techniques that facilitate interpreting the rationale for decision should be preferred, in order to be capable of identifying potential safety issues.

REQ4 Control and Actuation.

A self-learning system should have precise control mechanisms to actuate vehicle comfort systems. It needs to ensure smooth and pertinent control while adhering to safety requirements. Additionally, it should be subject to user scrutiny and aligned with user preferences [24]. Its actions should be understandable to the user [25].

REQ5 Safe Interaction.

The interaction between a self-learning comfort system and the driver must be performed safely, by avoiding distractions while driving [108]. Hence, a self-learning comfort system should behave unobtrusively, requiring neither explicit setting of preferences nor rating of the system's performance [76]. Since the user must not be constrained to perform additional work to provide explicit feedback to the system, the system should be able to handle data instances that are not provided with a ground truth value indicating the user's true preferences.

REQ6 Privacy and Security.

A self-learning comfort system shall ensure that user's information remains private and secure by fulfilling the data protection and privacy guidelines.

It should request customer's consent before collecting, processing and using the in-vehicle data. Any personal information stored in the vehicles shall be encrypted.

REQ7 Integration and Extensibility.

The system should be designed to seamlessly integrate with various platforms, applications and services [107]. It should be extensible and should have mechanisms for over-the-air updates to incorporate new features, bug fixes, and security patches.

3.3 Architecture

The main components of a self-learning comfort system are presented in Figure 3.2. These components work together to meet the system requirements for a self-learning vehicle comfort system presented in the previous section.

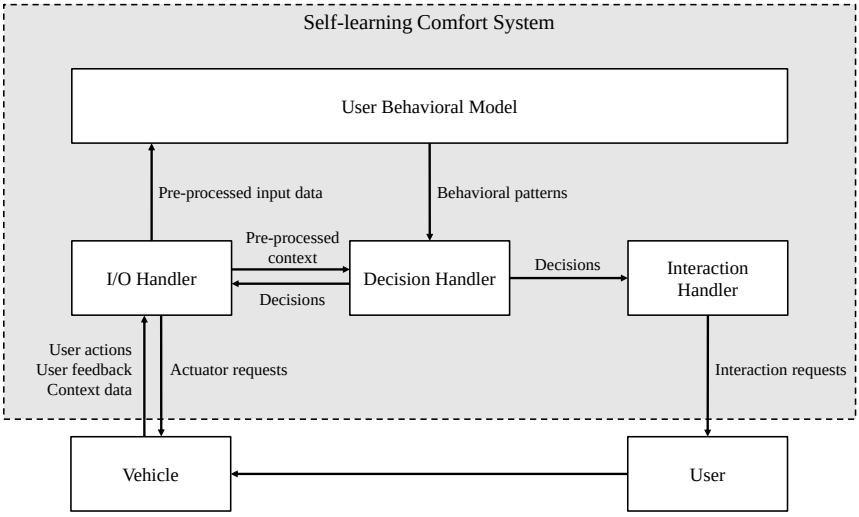


Figure 3.2: High-level overview of the components of a self-learning comfort system.

Input/Output Handler

The Input/Output handler (I/O handler) is responsible for managing the flow of data between the self-learning comfort system and its external environment, the rest of the vehicle. By separating how vehicle data is acquired from how it is used, the self-learning comfort system is extendable and flexible ([REQ7]).

The I/O handler has two main tasks. Firstly, it receives and processes the context information, the actions performed by the user and the user's direct and indirect feedback from the vehicle ([REQ2]). This involves data validation and transformation, to ensure compatibility with the user behavioral model and the decision handler. If necessary, it must perform data anonymization techniques to remove or modify personally identifiable information ([REQ6]). Secondly, it manages the delivery of requests generated by the decision handler towards the appropriate vehicle's actuators to control the comfort features ([REQ4]). This can involve formatting, encoding, or packaging the output signals to match the expectations of the receiving vehicle actuators.

User Behavioral Model

The user behavioral model is the central element of a self-learning comfort system. It is responsible for learning the user's individual behavioral patterns, including her preferences with vehicle comfort features based on historical data and contextual information ([REQ1], [REQ2]). The model learns by continuously "observing" the actions performed by the user, taking into account her reactions to the responses of the self-learning comfort system (the direct and indirect user feedback).

In addition to learning from user behavior, the model can also predict upcoming user actions, allowing the self-learning comfort system to anticipate the user's needs and react accordingly. The model also provides an indication of the level of uncertainty in the behavioral patterns it has identified, which can be useful for system optimization and performance evaluation.

Moreover, it shall enable over-the-air updates ([REQ7]). The update process depends on the specific model structure and algorithms utilized, but typically involves updating the model's weights or parameters while preserving the existing knowledge. Lastly, if historical user personal data is stored, it shall be protected using data encryption or data anonymization techniques ([REQ6]).

A user behavioral model for self-learning comfort systems in vehicles is presented in Chapter 4.

Decision Handler

The function of the decision handler is to determine the appropriate response of the self-learning comfort system based on the current contextual information and the set of behavioral patterns inferred by the user behavioral model ([REQ3]).

Every time the current context information changes, the decision handler evaluates if it matches with any of the behavioral patterns already learned by the user behavioral model. If so, it examines the level of uncertainty reported by the user behavioral model and decides what to do accordingly. Once a decision is made, it initiates the corresponding operations to accomplish it. Decisions are forwarded to I/O handler if an actuator must be requested, and to the interaction handler, in case the system must interact with the user.

The decision handler implements decision logic or algorithms that guide the decision-making process. Transparent and interpretable techniques that facilitate identifying potential safety issues shall be preferred ([REQ5]).

Interaction Handler

The interaction handler is the component of a self-learning comfort system that interacts with the user, providing information and controls. The interaction can happen via visual, acoustic and/or haptic channels, and must not interfere with

the driving task ([REQ5]). The space where the interactions between the user and the self-learning comfort system occurs is called user interface (UI).

The user's implicit and explicit feedback gathered by the UI (e.g. abortion of an automation suggestion) is collected by the I/O handler and directed to the user behavioral model, which updates and improves the user behavioral patterns accordingly.

4 A Novel Bayesian User Behavior Model

As mentioned in Section 3.3, the user behavioral model is a fundamental component of a self-learning comfort system. Probabilistic user modeling approaches are more appropriate than non-probabilistic ones in applications where uncertainty plays a fundamental role and where user-specific models are required, as discussed in Section 2.1.3. Hence, this section introduces a probabilistic nonparametric user model that addresses the system requirements outlined in Section 3.2.

Additional information about the probability distributions presented in this chapter can be found in the Appendix D.

4.1 Model Description

The behavior of a user u with a vehicle functionality f, is represented as a set of context-action pairs $B^{u,f} = \{(x_1, a_1), (x_2, a_2), \dots\}$, where a_n is a variable that represents the actions performed with the vehicle functionality f (e.g. activate, deactivate), x_n represents the context information retrieved from the vehicle when the action was performed, such as location, time, vehicle speed or outside temperature. The length of the dataset $B^{u,f}$ may increment as the user performs new actions.

The proposed behavioral model is represented in Figure 4.1, and the variables and parameters of the model are described in Table 4.1. The model assumes that

each context-action pair (x_n, a_n) can be explained by a latent variable, z_n, which represents the behavioral pattern responsible for that behavior.

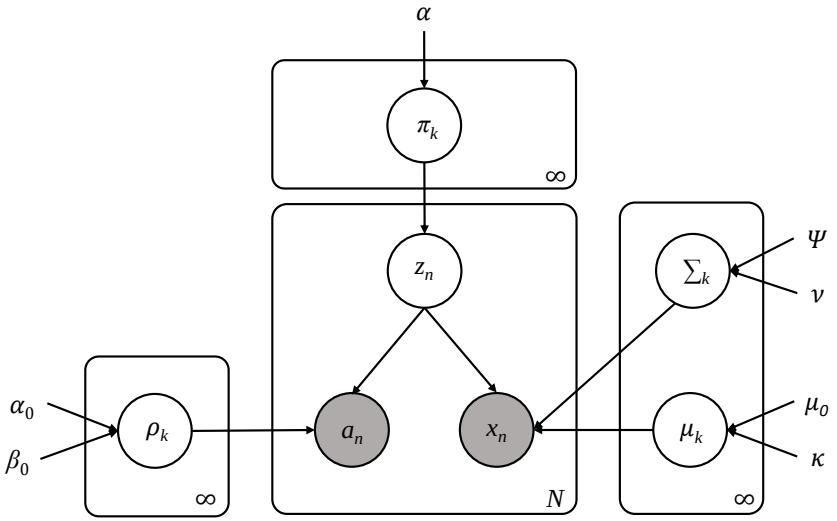


Figure 4.1: Graphical representation of the user behavioral model. Shaded nodes represent observed variables, unshaded nodes represent latent variables, arrows denote dependence, and rectangles denote the number of copies of the outlined structure.

A behavioral pattern represents the common cause or confounder between user actions a_n and context circumstances x_n. This kind of relationship is called a *fork* and can be represented graphically with the Bayesian network [109]:

$$x_n \leftarrow z_n \rightarrow a_n$$

Behavioral patterns make user actions and context circumstances statistically correlated even though there is no direct causal link between them. For example, in the following association:

$$\textit{Open Window} \leftarrow \overset{\text{confounder}}{\boxed{\text{pick-up order}}} \rightarrow \textit{Being at 20:00 at McDonald's}$$

Symbol	Description
$\boldsymbol{x}_n$	n-th context data point
$\boldsymbol{a}_n$	n-th action data point
$\boldsymbol{z}_n$	Behavioral pattern assignment of the n-th context-action pair
N	Number of behavioral patterns
α	DP weight concentration hyperparameter
$\boldsymbol{\pi}_k$	Weight of the k-th behavioral pattern
$\boldsymbol{\mu}_k$	Mean of the k-th context distribution
$\boldsymbol{\Sigma}_k$	Covariance matrix of the k-th context distribution
$\boldsymbol{\rho}_k$	Probability of success of the k-th action distribution
$\boldsymbol{\mu_0}, \kappa, \boldsymbol{\Psi}, \nu$	Normal-inverse-Wishart prior hyperparameters
α_0, β_0	Beta prior hyperparameters

Table 4.1: Variables and hyperparameters of the user behavioral model.

People who are at 20:00 at McDonald's tend to open their car's window, but the relationship is not one of cause and effect. Making a person being at 20:00 at McDonald's will not make her open the window of her car. Instead, both variables are explained by a third, the confounder, which is picking-up the order in the drive-thru (the behavioral pattern). This correlation can be eliminated by conditioning on the confounder: if only the actions performed by people who go to the McDonald's drive-thru is analyzed, no relationship between the time and location where vehicles' windows are opened is expected.

Since the number of behavioral patterns responsible for the behavior of a user is unknown, the user behavioral model places of nonparametric prior based on the Dirichlet process over z_n. This way, the model is able to automatically discover the number of behavioral patterns based on the previously observed action-context pairs. Each behavioral pattern z_n is explained by two distributions: the context distribution and the action distribution, described in sections 4.1.1 and 4.1.2, respectively.

4.1.1 Context Distribution

The context distribution associated with a behavioral pattern describes the region in the context space of similar circumstances that govern the user's actions. In the following, the terms "context distribution" and "context region" are used interchangeably, because context distributions describe regions in the context space. Each behavioral pattern assignment, z_n, denotes the unique context region associated with each context data point x_n.

Since user actions are affected by the sum of many independent random context variables, such as the weather, mood or location, a context region is represented by a multivariate Normal distribution (MVN). Accordingly, the context region of a behavioral pattern z_n is defined by $\mathcal{N}(\mu_k, \Sigma_k)$, which is the distribution likely to have generated the context data points x_n. Context vectors that are associated with the same context distribution are considered similar. Intuitively, one can think of it as measuring similarity by distance: each component defines a region in a multidimensional space in the same way one might think of a region in a physical location.

Throughout this work, context regions are represented graphically by the confidence regions of the associated MVN distribution. A confidence region is a multi-dimensional generalization of a confidence interval. It is defined as a set of points in an n-dimensional space, often represented as an ellipsoid around a point which is an estimated solution to a problem. For instance, a 3-sigma confidence region of a MVN distribution represents a region in the multivariate space around the mean vector of the distribution that contains approximately 99.73% of the data points. This region is defined by an ellipsoid, and it widens or narrows depending on the covariance structure of the multivariate normal distribution. The 1-sigma, 2-sigma, and 3-sigma confidence regions of a bivariate Normal distribution are shown in Figure 4.2.

The user behavioral model assumes that context data points are generated by a mixture of such multidimensional Normal distributions, where the number of mixture components, represented by the behavioral patterns, is unknown. Such

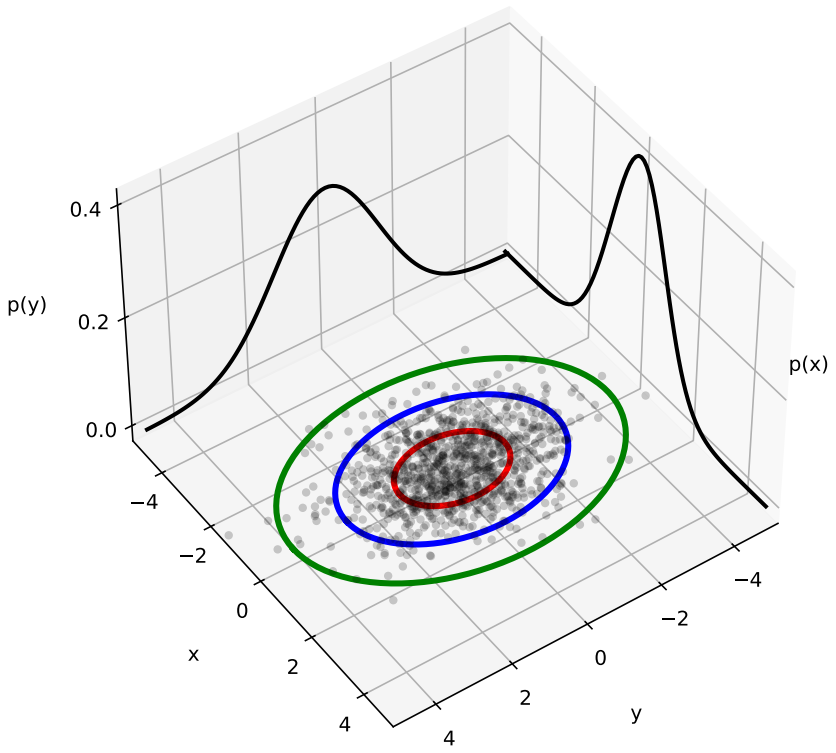


Figure 4.2: Illustration of a bivariate Normal distribution with mean $\mu = [0, 0]$ and covariance matrix $\Sigma = [\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}]$, shown along its two marginal distributions, $p(X)$ and $p(Y)$, and the 1-sigma (red), 2-sigma (blue), and 3-sigma (green) confidence regions, represented by ellipses in the corresponding colors.

nonparametric generalization of a Gaussian mixture model is called Dirichlet process Gaussian mixture model (DPGMM). The parameters of each context distribution are drawn from the DP so that $x_n \sim \mathcal{N}(\theta_{z_n})$, where $\theta_{z_n} \triangleq (\mu_k, \Sigma_k)$.

The generative process of context data points is described as follows:

$$
\begin{aligned}
\pi &\sim \text{GEM}(\alpha) \\
z_n &\sim \text{Categorical}(\pi) \\
\theta_k &\sim \text{DP}(\alpha, \text{NIW}(\mu_k, \Sigma_k)) \\
x_n &\sim \mathcal{N}(\theta_{z_n}),
\end{aligned}
\tag{4.1}
$$

where *GEM* represents the Griffiths-Engen-McCloskey distribution and *NIW* represents the Normal-inverse-Wishart distribution. The choice for the *NIW* distribution as the prior for the mean and covariance matrix of the Normal distributed context components is guided by mathematical and practical convenience. In this case, the natural choice of priors for the mean of a multivariate Normal is a Normal distribution, and an inverse-Wishart distribution for the covariance matrix, forming the joint distribution of $\boldsymbol{\mu}$ and Σ the Normal-inverse-Wishart distribution [110].

The multidimensional Normal distribution provides an adaptive and flexible representation for each individual region of similar context circumstances, while the Dirichlet process prior allows representing an unknown number of behavioral patterns. Both automatically adjust the complexity of the user behavioral model based on the available data. This is a key property of the user behavioral model.

4.1.2 Action Distribution

The action distribution associated with a behavioral pattern defines the probability of a future user action, given the set of actions that have been performed following that behavioral pattern and any form of user feedback.

In interactive scenarios, feedback can be direct or indirect. Direct feedback, also known as explicit feedback, is given to the model when users perform an action actively to indicate their acceptance or disapproval with the decisions made by the system. For instance, when users press the *Like* button to show their agreement

with the recommendations made by an intelligent system. Indirect or implicit feedback, on the other hand, is the one provided without an explicit user action. For example, a user, who every Saturday evening orders food, but one Saturday skips it, is indirectly informing the model about a change in her routines. Hence, the model's probability of her ordering food any particular Saturday must be lower than if she had not missed it. Indirect feedback is specially important when modeling user behavior in vehicular environments, because users are not always able to interact with an intelligent system directly, since they must pay attention to the road.

For simplicity and without loss of generality, it is considered that vehicle comfort functionalities have two different configurations, such as open and close (default) in the case of the power windows, or active and inactive (default) in the case of the seat heater. Hence, a user action is modelled using a binary variable $a \in \{0, 1\}$, where $a = 1$ means success (activation or positive feedback), and $a = 0$ means miss (inaction or negative feedback). The user behavioral model assumes that user actions made under similar circumstances, i.e., according to a behavioral pattern, are Bernoulli distributed: $a \sim \text{Bernoulli}(\rho)$. Accordingly, the probability function of any activity a given $\rho \in [0, 1]$ can be written as:

$$p(a|\rho) = \rho^a (1 - \rho)^{1-a} \tag{4.2}$$

In other words, user actions take the value 1 with probability ρ, and the value 0 with probability $1 - \rho$. For example, $\rho = 0.5$ indicates that the user is equally likely to activate or miss activating a comfort system, given a behavioral pattern. Therefore, to define the probability of a future user action it is essential to know the value of the parameter ρ.

The value of the probability of success, ρ, is initially unknown and is determined by the behavioral pattern governing the user actions. As the model gains evidence of user's routines, it becomes capable of inferring the value of ρ. For mathematical convenience, it is assumed that the prior distribution for the probability parameter

ρ is the Beta distribution, as it is conjugate to the distribution that generates the data, the Bernoulli distribution.

The Beta distribution is a distribution on probabilities, and it is governed by two parameters: α, representing the number of successes, and β, the number of misses. The probability density function of a Beta distribution with shape parameters α and β is represented in Equation 4.3, where $B(\alpha, \beta)$ is the Beta function. The Beta function can be viewed as the normalizing constant of the distribution, and it is defined in Equation 4.4, where $\Gamma(\cdot)$ represents the Gamma function.

$$p(\rho|\alpha, \beta) = \text{Beta}(\rho|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} \rho^{\alpha-1}(1-\rho)^{\beta-1} \tag{4.3}$$

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{4.4}$$

Figure 4.3 shows the probability density function of the Beta distribution for several parameter values. It can be observed that the Beta distribution with parameters $\alpha = \beta = 10$ represents a strong initial assumption that the user is equally likely to activate or miss activating a comfort system ($\rho = 0.5$). Conversely, the Beta$(2, 2)$ distribution reflects greater uncertainty on the assumption $\rho = 0.5$ compared to Beta$(10, 10)$ due to its higher variance. In contrast, Beta$(1, 1)$ corresponds to a uniform (flat) prior, indicating complete uncertainty with no prior knowledge about the probability of success. In this case, it assigns equal probability to all values in the $[0, 1]$ range.

Considering a dataset of N user activity data points $\{a_1, a_2, ..., a_N\}$ made under similar contextual circumstances, all assigned to the same behavioral pattern z_k, and an initial belief of the user behavior under such circumstances, represented by a prior probability distribution Beta(α_0, β_0), the posterior distribution can be expressed as indicated in Equation 4.5, where $\sum_n a_n$ represents the number of times the user activated the comfort function or provided positive feedback, and $\sum_n (1 - a_n)$ indicates the number of times the user missed activating or gave negative feedback.

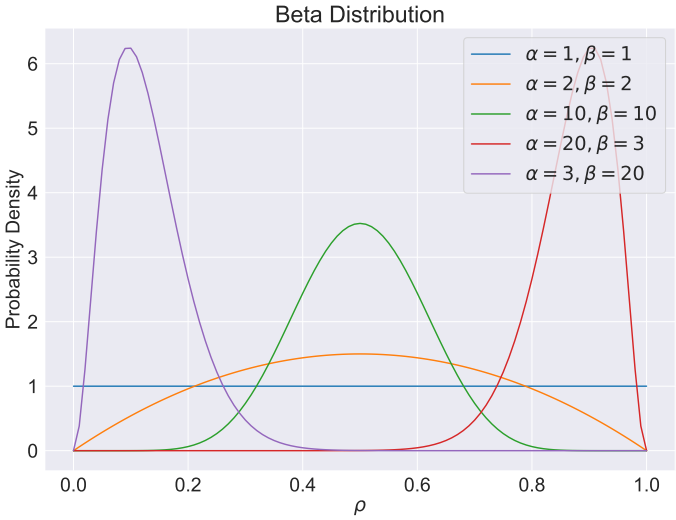


Figure 4.3: Probability density function of the Beta distribution with different parameters.

$$\begin{aligned} p(\rho_k | a_{1:N}, \alpha_0, \beta_0) &\propto \rho_k^{\alpha_0 + \sum_n a_n - 1} (1 - \rho_k)^{\beta_0 + \sum_n (1 - a_n) - 1} \\ &\propto \text{Beta}(\alpha_0 + \sum_n a_n, \beta_0 + \sum_n (1 - a_n)) \end{aligned} \tag{4.5}$$

Hence, Equation 4.5 can be expressed as follows, where # represents "count of":

$$p(\rho_k | a_{1:N}, \alpha_0, \beta_0) \propto \text{Beta}(\alpha_0 + \#\text{activations}, \beta_0 + \#\text{misses}) \tag{4.6}$$

Thus, the posterior distribution is a Beta distribution with parameters $\alpha' = \alpha_0 + \#\text{activations}$ and $\beta' = \beta_0 + \#\text{misses}$. Essentially, the Beta distribution acts as a counter, where α and β represent the counts of activating or miss activating a comfort function. Every newly observed activation is added to the existing prior belief to compute the posterior, which can become a prior for the next activity.

This framework can be extended to use different weights to control the rate at which α' and β' change in response to new evidence. For example, if the user provides direct feedback, a higher weight can be assigned to α' than when the user gives indirect feedback, since direct feedback provides more accurate information than indirect feedback. If the weight for direct positive feedback information is represented as w_{pos_d}, the weight for indirect positive feedback as w_{pos_i}, the weight for direct negative feedback as w_{neg_d}, and the weight for indirect negative feedback as w_{neg_i}, the updated parameters α' and β' can be calculated as:

$$\begin{aligned}\alpha' = \alpha_0 + w_{pos_d} \cdot \#\text{positive direct actions} + \\ w_{pos_i} \cdot \#\text{positive indirect actions}\end{aligned} \tag{4.7}$$

$$\begin{aligned}\beta' = \beta_0 + w_{neg_d} \cdot \#\text{negative direct actions} + \\ w_{neg_i} \cdot \#\text{negative indirect actions}\end{aligned} \tag{4.8}$$

The extended action distribution for a behavioral pattern z_n can thus be calculated as:

$$p(\rho_k | z_n, B^{u,f}) \propto Beta(\alpha', \beta') \tag{4.9}$$

where α' and β' are defined in Equation 4.7 and 4.8, correspondingly.

For instance, if the weights are set to $w_{pos_i} = 1$ and $w_{neg_i} = 0.5$ and a user activates a comfort system 7 times (positive indirect feedback) and misses activating it 3 times (negative indirect feedback) at nearly the same circumstances, the likelihood function of this observed data can be calculated using the Bernoulli distribution as follows:

$$P(data|\rho) = \rho^{7 \cdot w_{pos_i}} (1-\rho)^{3 \cdot w_{neg_i}} = \rho^7 (1-\rho)^{1.5}$$

Assuming that all 10 actions are made at nearly the same contextual circumstances (i.e., follow the same behavioral pattern), and that initially there is complete uncertainty about the user actions, so that $\alpha_0 = \beta_0 = 1$, the posterior distribution can be calculated as per Equation 4.9 as:

$$p(\rho_k|z_n, B^{u,f}) \propto \text{Beta}(\alpha_0 + 7, \beta_0 + 1.5) \propto Beta(8, 2.5)$$

Figure 4.4 shows how, initially, the model is completely uncertain about the value of ρ_k (prior distribution). After the user has performed 10 actions (activations and misses), the model's uncertainty diminishes, as it has gained understanding on the user's likely behavior. This improved understanding is captured by the posterior probability distribution over ρ_k.

Finally, there are several methods to estimate the value of ρ_k from the posterior distribution. One common method is to calculate the expected value of ρ_k, indicated by the mean of the posterior distribution. The mean of a Beta distribution is defined in Equation 4.10.

$$\mathbb{E}[\text{Beta}(\alpha, \beta)] = \frac{\alpha}{\alpha + \beta} \tag{4.10}$$

Accordingly, the estimated value of ρ_k in the previous example would be equal to $\rho_k \approx \mathbb{E}[\text{Beta}(\alpha, \beta)] = 8/10.5 = 0.762$. This means, that the probability of the user activating the comfort system in the future under such contextual circumstances, defined as the probability of success, would be 76.2%.

Similarly, other Bayesian point estimates can also be calculated. In especial, the mean and variance of the posterior distribution provide valuable information for decision-making, by quantifying central tendencies and variability. To make decisions, these statistics can be compared to predefined thresholds.

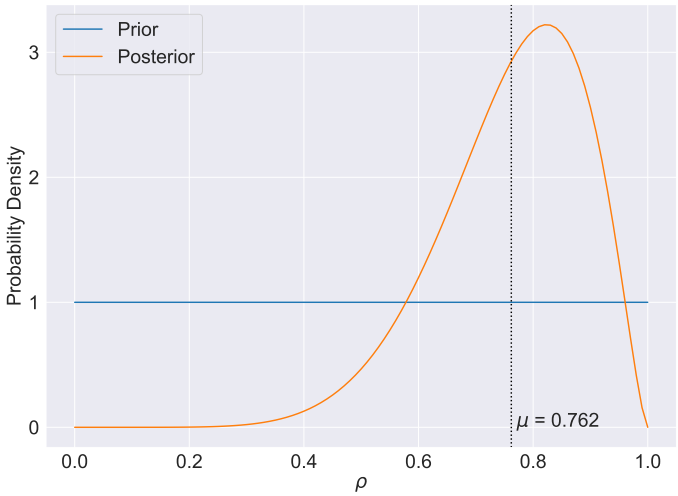


Figure 4.4: Prior (Beta(1, 1)) and posterior (Beta(8, 2.5)) distributions for ρ, indicating the probability of success (activation). The black dotted line denotes the mean of the posterior distribution at 0.762.

4.1.3 Model Hyperparameters

In Bayesian settings, a hyperparameter is a parameter of a prior distribution. The interpretation of the hyperparameters of the user behavioral model are discussed in the following:

- Mean hyperpriors of the context distribution ($\boldsymbol{\mu_0}$ and κ). The mean of each context distribution is estimated from $\kappa > 0$ observations with sample mean $\boldsymbol{\mu_0}$. Intuitively, $\boldsymbol{\mu_0}$ is a vector that controls the position of context regions in the feature space. Larger values of κ concentrate the context regions around $\boldsymbol{\mu_0}$.
- Covariance hyperpriors of the context distribution ($\boldsymbol{\Psi}$ and ν). The natural conjugate prior distribution of the covariance matrix of a multivariate Normal distribution is the Inverse-Wishart distribution $IW_p(\boldsymbol{\Psi}, \nu)$, because it has support over real-valued positive-definite matrices. It is parametrized

by a positive-definite scale matrix $\mathbf{\Psi} \in R^{p \times p}$ and degrees of freedom $\nu > p - 1$. $\mathbf{\Psi}$ is used to position the IW distribution in parameter space, and ν set the certainty about the prior information in the scale matrix. The larger the ν, the higher the certainty about the information in $\mathbf{\Psi}$, and the more informative is the distribution. The least informative specification then results when $\nu = p$, which is the lowest possible number of ν. Intuitively, ν is used to control the strength/informativeness of the prior. High values $\nu \gg p$ indicate a strong prior, while $\nu = p+1$ is the least informative setting. The size of the variance is partly determined by $\mathbf{\Psi}$: the smaller the elements of $\mathbf{\Psi}$, the smaller the variance of the IW distribution, and hence the more informative the prior will be. That is, the smaller values in the scale matrix $\mathbf{\Psi}$, the more confidence is deposited on the covariance prior.

- Weight concentration prior (α) of the Dirichlet process. Specifying a low value for the concentration prior makes the model put most of the weight on few behavioral patterns and set the weights of the remaining components very close to zero. In other words, unnecessary behavioral patterns are easily pruned from the model, but new patterns are less likely to be added during inference. High values of the concentration prior, on the other hand, allow a larger number of components to be active in the mixture. The value of the α must be greater than 0.

- Success prior (α_0, β_0) of the action distribution. Higher values of $\alpha_0 > \beta_0$ indicate that, a priori, it is more likely to observe positive user feedback than negative feedback (activations vs. deactivations or absence of actions). The success prior controls the initial belief on the "amount" of positive and negative user feedback on the vehicle functionality. For simplicity, throughout this work it is assumed that initially, there is no strong prior beliefs about the probability of success. This is achieved by setting the initial values of the success prior as $\alpha_0 = \beta_0 = 1$, which corresponds to a uniform prior, Beta$(1, 1)$.

Hyperparameters are not learned from the data, but rather set prior to model training. They can greatly impact the performance of the model. Hence, finding

the values that lead to best model performance is challenging. Throughout this work, the values of the hyperparameters of the user behavioral model are defined based on expert knowledge. Nevertheless, other strategies shall be explored in future work, such as learning their optimal value based on fleet data (see Section 8.2).

4.1.4 Generative Process

The generative story for any action-context pair (a_n, x_n) in the user behavioral model is as follows. A behavioral pattern assignment z_n is sampled from the set of all possible behavioral patterns, which are distributed according to the weights in π. Given the behavioral pattern assignment: firstly, a context data point x_n is drawn from the context distribution with parameters $\theta_{z_n} \triangleq (\mu_{z_n}, \Sigma_{z_n})$, which are distributed according to a Normal-inverse-Wishar distribution. Secondly, an action data point a_n is drawn from the action distribution with parameter ρ_{z_n}, which is Bernoulli distributed.

The generative process can therefore be summarized as follows:

1. Draw $\pi_k|\alpha \sim \text{GEM}(\alpha)$ for $k = 1, 2, ...$
2. Draw $(\mu_k, \Sigma_k) \sim \text{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{\mu}_0, \kappa, \boldsymbol{\Psi}, \nu)$ for $k = 1, 2, ...$
3. Draw $\rho_k \sim \text{Beta}(\boldsymbol{\rho}|\alpha_0, \beta_0)$ for $k = 1, 2, ...$
4. For each data point $n = 1, 2, ..., N$ do the following:
    a) Sample a behavioral pattern $z_n|(\pi_1, \pi_2, ...) \sim \text{Categorical}(\boldsymbol{\pi})$.
    b) Sample a context $x_n|(z_n, \mu_1, \Sigma_1, \mu_2, \Sigma_2, ...) \sim \mathcal{N}(x_n|\boldsymbol{\mu}_{z_n}, \boldsymbol{\Sigma}_{z_n})$.
    c) Sample an action $a_n|(z_n, \rho_1, \rho_2, ...) \sim \text{Bernoulli}(a_n|\rho_{z_n})$.

4.1.5 Predictive Probability

A fundamental characteristic of the user behavioral model of a self-learning comfort system is that it can be used to predict future user actions given a new vector of contextual information. The predictive distribution of the behavioral model represents the probability that the user performs an action at a new context, conditioned on the user's behavior in the past. The predictive distribution is given in Equation 4.11, where x^* represents a future new context vector, $B^{u,f}$ represents the previously observed action-context pairs (i.e., the past user behavior) and $\Theta \triangleq \{\mu, \Sigma, \rho, \pi\}$ represents the latent variables of the behavioral model.

$$p(a|\, x = x^*, B^{u,f}, Z, \Theta) \tag{4.11}$$

As already mentioned in Section 4.1, the variables a and x are conditionally independent given the confounding behavioral pattern z. Therefore, one can equivalently write the predictive probability as follows:

$$p(a|\, z = z^*, B^{u,f}, \Theta) \tag{4.12}$$

where z^* represents the behavioral pattern to which x^* most likely belongs to.

The behavioral pattern z^* to which x^* is assigned to, is the one whose associated context distribution most likely generated the data point x^*:

$$z^* = \operatorname{argmax} p(x^*|z) = \operatorname{argmax} \mathcal{N}(x^*|\mu_z, \Sigma_z), \tag{4.13}$$

where the probability density function of a multivariate Normal distribution is given by Equation 4.14.

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})] \tag{4.14}$$

The quantity $d_M(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$ is known as the Mahalanobis distance, and it represents how many standard deviations away the point x is from the center of the probability distribution parametrized as (μ, Σ). Thereby, making use of the Mahalanobis distance, the value of z^* can be equivalently determined by Equation 4.15.

$$z^* = \operatorname{argmin} d_M(x^*; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \tag{4.15}$$

Identifying the behavioral pattern z^* that is most likely responsible for generating the data point x^* from among all the model's behavioral patterns does not necessarily imply that z^* is a suitable match. It is essential to assess whether there is sufficient evidence to support that the context distribution associated with z^* indeed generated the data point x^*. To consider z^* a good match, the Mahalanobis distance between x^* and $\mathcal{N}_{z^*}(\mu, \Sigma)$ is examined. This Mahalanobis distance must be less than a predefined threshold, denoted as δ, measured in terms of standard deviations. In simpler terms, x^* is assigned to z^* only when x^* falls within the δ-sigma confidence region. Mathematically, this is represented by extending Equation 4.15 as:

$$z^* = \underset{d_M(x^*; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) < \delta}{\operatorname{argmin}} d_M(x^*; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \tag{4.16}$$

Finally, once the behavioral pattern z^* is identified, the calculation of the probability of a future user activation is determined by the action distribution associated to it, given in Equation 4.17.

$$p(\rho |\, z = z^*, B^{u,f}, \Theta) \propto \text{Beta}(\alpha', \beta') \; \forall a_n \in z^* \tag{4.17}$$

where α' and β' are defined by Equation 4.7 and 4.8, respectively.

4.2 First Truncation-free VI Algorithm

Section 2.3.2 briefly reviewed the coordinate ascent variational inference algorithm for posterior inference over the latent variables of a DPM model. However, the CAVI algorithm, like all variational inference methods, imposes a restrictive fixed truncation in the number of components K, which is hard to set a priori on user-based applications. K is often either too small and inexpressive, or too large and computationally inefficient [111]. Moreover, the computation of the evidence lower bound as criteria to increase the number of components induces excessive computational burden ([112], [113]). Therefore, this work proposes a different strategy to circumvent truncation with lightweight computational cost.

This dissertation introduces a truncation-free inference algorithm that automatically estimates the value of K during the inference process. This is achieved by leveraging the information provided by the responsibilities r_{nk}, which are calculated in each iteration of the inference algorithm. The responsibilities r_{nk} of a behavioral pattern z_k can be seen as the responsibility that the behavioral pattern z_k takes for "explaining" the observed context data point x_n. Mathematically, the responsibilities r_{nk} are defined as:

$$r_{nk} = p(z_n = k|x_n, \boldsymbol{\theta}) \tag{4.18}$$

Behavioral patterns that take essentially no responsibility for explaining the data points have $r_{nk} \approx 0$. In consequence, the effective number of points assigned to such behavioral pattern k, N_k, will be $N_k = \sum_{n=1}^{N} r_{nk} \approx 0$.

The proposed inference algorithm starts with a relatively small value of K (typically $K = 1$) and allows expressive behavioral patterns to be created based on the data. The algorithm computes $q^*(z_n)$ via Equation 2.14 for $K + 1$ components. The first K components are "true" components, because they are actually realized by the data. The remaining component is an unrealized component. If at the end of an iteration the effective number of points assigned to the unrealized component N_{K+1} is greater than 0, it means that there are data points that do not fit into one

of the K realized components. In such a case, the number of components K is incremented by one. At the next iteration, the parameters of the new component are estimated via Equation 2.13.

To ensure that the model can effectively represent the new component and adapt to any shifts or variations that may have occurred in the data distribution, $q(\Theta)$ is initialized via the k-means algorithm every time that K increases. k-means iteratively assigns data points to clusters and updates the cluster centroids to minimize the sum of squared distances within each cluster. The algorithm converges when the centroids no longer change significantly. A detailed description of the k-means algorithm is provided by [114].

The inference algorithm also checks if there are components that provide insufficient contribution in explaining the data. If there are empty components, $N_k = 0$ for any $k = 1, ..., K$, the empty ones are effectively removed from the model and K is accordingly decremented by the number of empty components. These steps are repeated until convergence.

Similar to k-means, convergence is determined by proving that the means of the posterior distribution have stopped moving within some small tolerance $\epsilon > 0$ [10]. Monitoring changes in the means of the inferred components is a proxy to monitoring the ELBO of the full data and sidesteps its heavy computation [115]. Since components' means are computed at every iteration, no additional calculation is needed. The convergence check for the inference algorithm is given in Equation 4.19, where i represents the i-th iteration and $\boldsymbol{\mu}_k^i$ represents the value of the mean of the k-th context region at the i-th iteration.

$$\sum_{k=1}^{K} ||\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_k^{i-1}||^2 < \epsilon \tag{4.19}$$

Ultimately, once the behavioral pattern assignments, z_n, and the parameters of the context distributions, $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, have been identified, the inference algorithm determines the value of the parameter of the action distributions, ρ_k. For that, the algorithm determines the context-action pairs assigned to each behavioral pattern

according to the responsibilities, r_{nk}, calculated during inference of the context distributions. Context-action pairs are assigned to the behavioral pattern with the higher responsibility. Finally, the activation probability, $p(\rho_n)$, is calculated via Equation 4.9.

Since users' preferences and behavior are continuously changing, recent user actions are more informative than older ones. In other words, the importance or informativeness of a sample can be defined by its age. Hence, the user behavioral model must be able to adapt and update its learned patterns based on more recent data while gradually disregarding older data that may have become less relevant or accurate over time. This capacity to "forget" can be implemented by removing older data points from the dataset of context-action pairs, $B^{u,f}$.

One method to ensure that the dataset remains current and relevant is to impose a temporal constraint on the age of the data entries included in the dataset ([116], [117]). This technique is known as time-based sliding window, and it is widely used for concept drift detection [118]. The length of the sliding window, W, defines the upper limit on the age of the data entries, and it is a tuneable model parameter.

Mathematically, if the time-based sliding window technique is applied to the dataset $B^{u,f} = \{(a_t, x_t)\}$, $t = \{1, 2, ...\}$ to retain only recent data points, the resulting truncated dataset is defined by:

$$B^{u,f}_{\text{recent}} = \{(x_t, a_t) \,|\, t_{\text{current}} - t < W\} \tag{4.20}$$

where t_{current} denotes the current time.

The larger the window length, the more impact older samples have on the learned behavioral patterns. By employing this time-based truncation strategy on the dataset, the model maintains a focus on the most up-to-date observations, allowing for the consideration of evolving trends and patterns, while discarding older information that might have become less pertinent over time.

The pseudocode describing the proposed algorithm for the inference of the behavioral patterns and context regions is described in Appendix A.

4.3 Demonstration of the User Behavioral Model

This section demonstrates with a realistic example that the user behavioral model presented in Section 4.1 is useful to discover user behavioral patterns. This demonstration is not intended to be a thorough study of the user behavioral model, since a much deeper analysis is provided in Chapter 6.

Alice is the user in this example. Every Saturday morning she goes shopping, and parks her car in the parking garage of the supermarket, for what she opens the window of her car to get the ticket. Additionally, Alice opens the window once or twice a month when she goes to the drive-thru of a restaurant close to her house, to remove the ice that covers the window when she parks outside, or when she enters other underground parking garages in the city center and the surrounding area.

Figure 4.5 represents the data points where Alice has opened the window in the past month. In total, she has opened the window 21 times: 8 times at the parking garage of the supermarket, 3 times at the drive-thru, and 10 times at different, random places. For the sake of the example, it is assumed that the supermarket is located at coordinates $(10, 10)$ and the drive-thru is at $(12, 12)$, so that both places are distant enough from each other, to prevent the data points to overlap. Moreover, for the purpose of illustration, only two context dimensions are represented in the example: latitude and longitude.

The novel truncation-free variational inference algorithm is applied on the data set to discover the behavioral patterns that represent Alice's behavior. An example of the learned context regions is depicted in Figure 4.6.

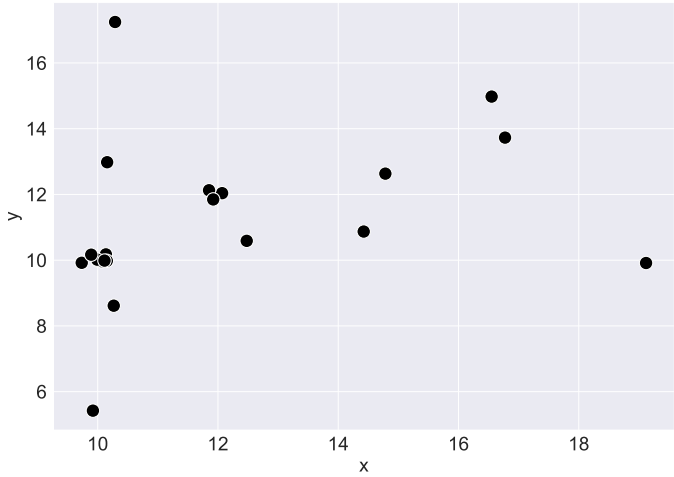


Figure 4.5: Data set representing the context circumstances where Alice has opened her car's window in the last month. It is assumed that the supermarket is located at coordinates $(10, 10)$ and the drive-thru is at $(12, 12)$

The context distribution of a behavioral pattern k is represented by the 3-sigma confidence ellipse of the multivariate Normal distribution $\mathcal{N}(\mu_k, \Sigma_k)$ representing the context region of the behavioral pattern. The use of 3-sigma confidence ellipses for representing the context distributions graphically helps to understand the range of possible values associated with the inferred behavioral pattern k, since it indicates the region in which data points are likely to fall with a high probability (approximately 99.7% of the data).

The training process is repeated for 100 Monte Carlo iterations, each presenting the data points in a random order. At each training time step t a new data point is presented. The maximum number of behavioral patterns, K, at each training time step averaged over the 100 iterations is illustrated in Figure 4.7. The figure shows how the value of K changes as new data arrives. Unlike the traditional CAVI algorithm, which relies on a predetermined value of K, the novel VI algorithm can adjust K based on the data, resulting in a more flexible and adaptive model.

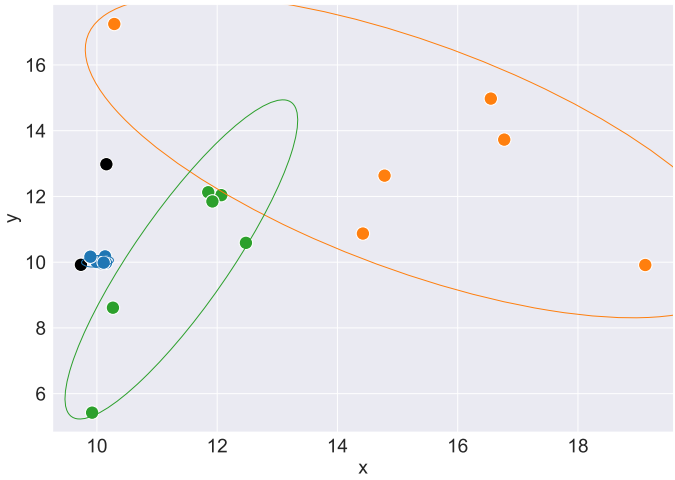


Figure 4.6: Context components and context components inferred by the truncation-free VI algorithm. The truncation-free algorithm determines automatically the value of K, the number of components based on the data.

Building on this example, insights on the predictive capabilities of the user behavioral model can be gained by inspecting the action distribution of the learned behavioral patterns. As introduced in Chapter 4.1.2, each instance of Alice opening her car's window is a valuable source of indirect positive feedback to the model, because it contributes to understanding her unique behavior. Hence, every time Alice opens the window, the user behavioral model is positively rewarded ($w_{pos_i} = 1$), encouraging it to learn such behavior.

To be able to calculate how probable is that Alice opens the window in the future at some location, it is first necessary to determine if there is any behavioral pattern associated with that context, as per Equation 4.16. For instance, to calculate the probability of a future window opening action at the parking garage of the supermarket, located at (10, 10), it is first necessary to calculate the behavioral pattern it belongs to. The data point $x^* = (10, 10)$ is associated with the behavioral pattern closest to that point, where closeness is measured in standard deviations, calculated using the Mahalanobis distance between the

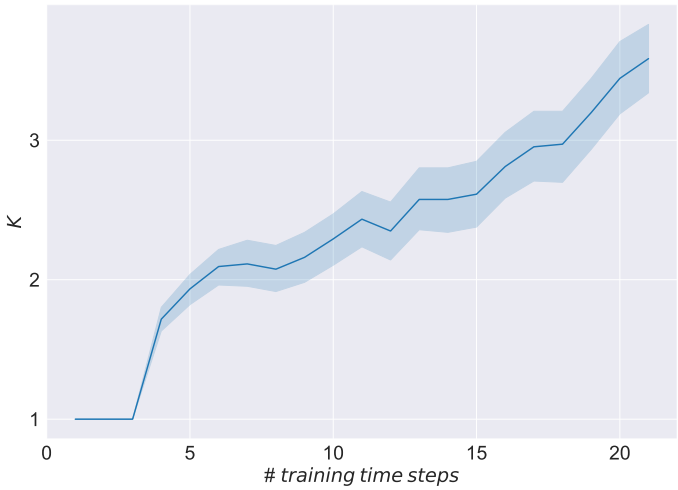


Figure 4.7: Number of behavioral patterns K inferred at every training step, averaged over 100 Monte Carlo iterations.

context distribution and the data point. As showed in Figure 4.6, the behavioral pattern assigned to $x^* = (10, 10)$ is the one colored in blue. Even though Alice had opened the window around the parking garage 8 times, the model has assigned 7 of them to the behavioral pattern (blue-marked points). Accordingly, the probability that Alice wants to open the window in the future at the parking garage can be calculated via Equation 4.9 as:

$$p(a = opening|z_n, B^{u,f}) \propto \text{Beta}(\alpha_0 + 7, \beta_0)$$

Assuming an uninformative prior distribution, for which $\alpha_0 = \beta_0 = 1$, the posterior distribution would be calculated as $p(a = opening|z_n, B^{u,f}) \propto \text{Beta}(8, 1)$, with mean $\mathbb{E}[\text{Beta}(8, 1)] = 0.889$ and variance $\text{Var}[\text{Beta}(8, 1)] = 0.01$. The resulting action distribution is depicted in Figure 4.8.

The probability that Alice opens the window at location $x^* = (16, 14)$ in the future can be calculated in a similar manner. In this case, the context region

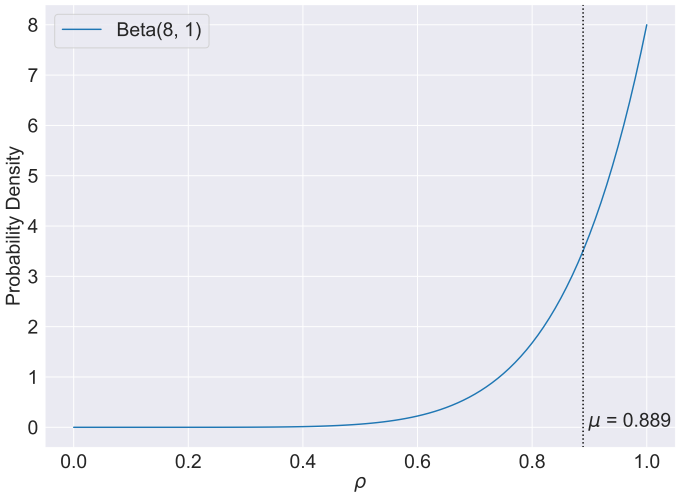


Figure 4.8: Action distribution representing the probability of Alice opening the window at the parking garage of the supermarket in the future. The action distribution follows a Beta distribution. The black dotted line indicates the mean.

associated with x^* is the one colored in orange. Since 6 data points belong to it, the probability of a future open window action is calculated as follows:

$$p(a = opening|z_n, B^{u,f}) \propto \text{Beta}(\alpha_0 + 6, \beta_0)$$

5 Feasibility Prototype of a Self-Learning Comfort System

Prototyping is an integral part of the design and development process of any system because it allows bringing conceptual or theoretical ideas to life and exploring their real-world impact before finally executing them [119]. By prototyping, assumptions, biases, and uncover insights about the users are revealed, which can be used to improve the system or create new solutions.

In the pursuit of validating the technical possibility of a self-learning comfort system, a feasibility prototype is built. The main objective of this preliminary study is to validate the conceptual design of a self-learning comfort system, examine the performance of the user behavioral model with real vehicle data, explore the design of the user interface, and asses the overall system acceptance.

The prototype is designed to automatically control the driver's window according to the driver's individual preferences. Hence, it is a prototypical implementation of a self-learning window system. The power window system is chosen for the prototype implementation due to its universal applicability and widespread use in modern vehicles. The choice allows participants to easily relate to routine scenarios, such as opening the window at a parking lot entrance or clearing the window from the frost and ice on cold winter days.

5.1 Functional Requirements

The following requirements for the prototype extend the general system requirements presented in Section 3.2 to encompass the specific goals and constraints of the prototypical implementation.

REQ-I The system shall learn user behavior within the limited evaluation duration.

In order to let participants experience the self-learning capabilities of the system during the evaluation period, it is crucial that the system quickly learns the user's preferred configurations.

REQ-II The system shall learn the driver's individual behavioral patterns involving the driver's power window.

The prototype self-learning window system shall be able to learn the participant's individual behavioral patterns involving opening and closing the window. During the prototype evaluation, the self-learning system shall assume that the only user is the driver. Consequently, the system shall focus solely on the information related to the driver's window for learning and automation.

REQ-III Location information shall be used to identify user behavioral patterns.

To facilitate ease of implementation, the prototype shall rely solely on location information for learning and identifying individual user routines. Specifically, it shall utilize the latitude and longitude coordinates obtained from the vehicle's GPS system. Collecting and processing location data simplifies the learning process, making it resource-efficient. Also, many real-world scenarios and user behaviors with the window system are location-dependent. Therefore, prioritizing location data allows the prototype to address common use cases effectively.

REQ-IV The system shall be eager to automate.

In the prototype setting, the primary goal is to demonstrate the core functionality of the self-learning system and allow users to understand how it works. Therefore, the prototype shall be encouraged to perform automated actions, even if the uncertainty level is higher than what would typically be encountered in real-life applications. Higher-uncertainty actions can still provide valuable feedback for system improvement. Users' reactions and responses to these actions can inform further refinements in the system's algorithms and decision-making processes.

REQ-V The system shall have full automation capabilities.

The goal of the prototype includes testing the hypothesis that in routine situations, highly automated systems are more useful than those with less automation capabilities. Hence, the prototype shall demonstrate full automation capabilities, meaning it shall control the window automatically without requiring user approval. By not requiring user approval for routine actions, the prototype shall also test how users adapt to and accept full automated systems. This information is crucial to align the final design of the user interface of a self-learning comfort system with the user expectations and preferences.

REQ-VI The system must not perform quick opposite automated actions.

The prototype must avoid opening and closing the window in quick succession because continuous and rapid window movements can pose safety risks and can lead to a poor user experience.

REQ-VII The user shall be informed about the system automated actions unobtrusively.

The design of the prototype's user interface shall allow the user to be aware of the system's capabilities and intentions. Transparency is a fundamental characteristic to improve trust and acceptance and to meet the safety standards. Therefore, the prototype shall inform the user in advance before performing automated actions, in order to allow the user to be aware of the system's capabilities and intentions.

REQ-VIII The system shall require the user's consent to initiate the learning process.

The prototype must guarantee privacy by ensuring that users are aware of and agree to the collection and analysis of their behavioral data. This not only aligns with data protection regulations but also promotes transparency and trust between users and the system.

5.2 Prototype Description

The functional architecture of the prototype follows the conceptual design presented in Chapter 3. The prototype of the self-learning power window system is designed to operate on a vehicle. The system's software is written in Python. It runs on a laptop connected with the car via an Ethernet-CAN interface, enabling the system to read from and write on the vehicle's main CAN buses. A class diagram of the software used for the self-learning window prototype is presented in Figure 5.1.

Participants directly interact with the self-learning system only once, at the beginning of the experiment ([REQ-V]). At vehicle initialization, the self-learning system asks users for permission to automatically control the windows based on their behavior, displaying for that a pop-up message on the laptop's screen ([REQ-VIII]). Figure 5.2 shows a snapshot of the prototype user interface when the system first asks the user for permission to start learning. Such interaction requests, and the user's answer, are sent out and processed in the interaction handler module, which in responsible for controlling the prototype's user interface. Moreover, users are notified about any upcoming automated action while driving via a similar unobtrusive, brief text notification displayed on the screen, which expires automatically after few seconds ([REQ-VII]).

The vehicle's current position is used as contextual information to infer user behavioral patterns. For this purpose, the I/O handler gathers the GPS longitude and latitude coordinates of the vehicle from the car's GPS system via the Ethernet-CAN

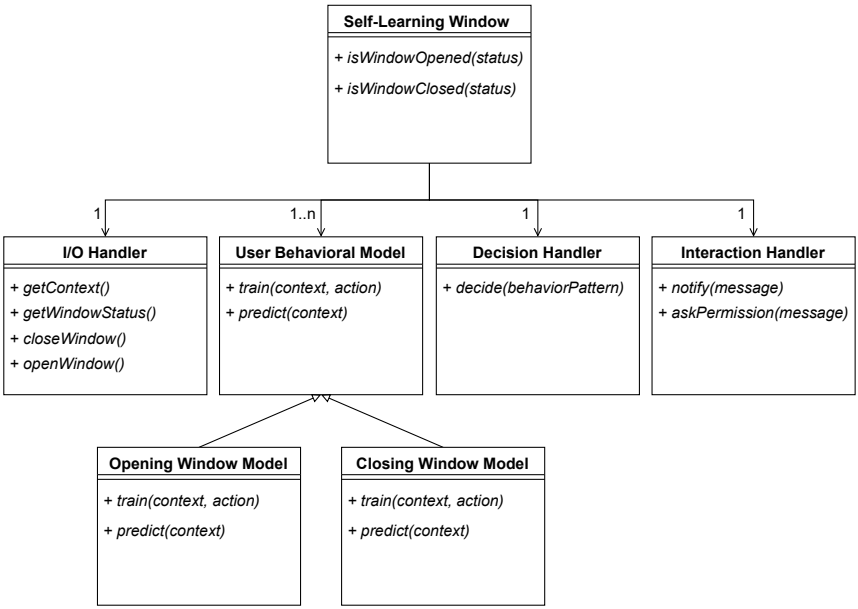


Figure 5.1: Class diagram of the self-learning window system implemented for the feasibility prototype.

interface ([REQ-III]). Furthermore, the I/O handler filters out any implausible or invalid CAN message, before sending the information further towards the user behavioral model or decision handler, by calling the function *preprocess(context)*.

A truncation-free version of the user behavioral model is implemented in the prototype self-learning window system to learn the user's individual behavioral patterns involving opening and closing the driver's window. The values of the model hyperparameters are listed in Table 5.1.

Two instances of the user behavioral model run in parallel on the prototype: the *opening window*-model and the *closing window*-model, because the prototype is required to learn the user's routines involving opening and closing the window ([REQ-II]). Hence, the opening-model is designed to continuously learn the user

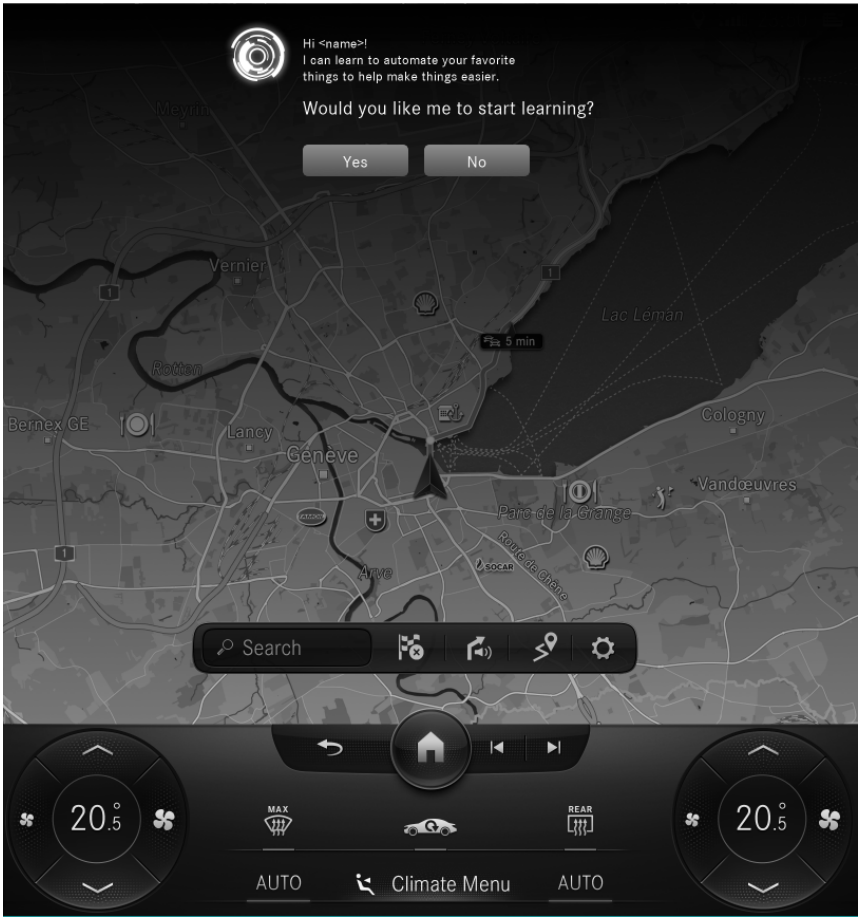


Figure 5.2: Snapshot of the system's initial UI design. The self-learning system requests user permission to start learning and automating comfort features.

behavioral patterns concerning opening actions, whereas the closing-model identifies the behavioral patterns related with closing activities. For this purpose, every time the driver's window is opened, the opening-model is positively rewarded ($w_{pos_i} = 1$), encouraging the model to learn such behavior. Similarly, the closing-model is rewarded when the driver's window is closed. Figure 5.3

Hyperparameter	Initial value
α	1
$\boldsymbol{\mu_0}$	$\mathbf{0}$
κ	10^{-6}
$\boldsymbol{\Psi}$	$\text{diag}(10^{-9}, 10^{-9})$
ν	$d + 1$
β_0	1
α_0	1

Table 5.1: Parameters and hyperparameters of the user behavioral model for the prototype evaluation (feature dimension $d = 2$). A vector of d zeros is denoted by $\mathbf{0}$.

shows the sequence diagram of how the prototype's components interact when learning the user's behavioral patterns by observing the user's behavior. The system continuously gathers the vehicle and user data, by calling the *getContext()* and *getWindowStatus()* methods of the I/O handler. Every time the user performs an action, the user behavioral model is re-trained by calling *train(context, action)*.

Having two different models to learn two opposite actions can originate conflicts. Due to the fact that the feature space is two-dimensional (latitude and longitude), both models can have very similar input data, so that the learned behavioral patterns may overlap. For example, if a user opens and closes the window several times within a small area, such as the entrance of a parking lot, the opening model may identify a behavioral pattern for opening the window, and the closing model may also find a behavioral pattern to close the window. Hence, next time the user is in that area again, the action of the system – opening or closing the window – is not straightforward, as both models may show high confidence levels. For example, the situation in Figure 5.4 represents how both models might show high confidence levels for opening and closing the window if the car is located at the black cross, leading to ambiguity in deciding the appropriate action.

To avoid such conflicts, every time the window is opened, besides rewarding the opening model, the closing model is slightly punished ($w_{neg_i} = 0.5$). Such

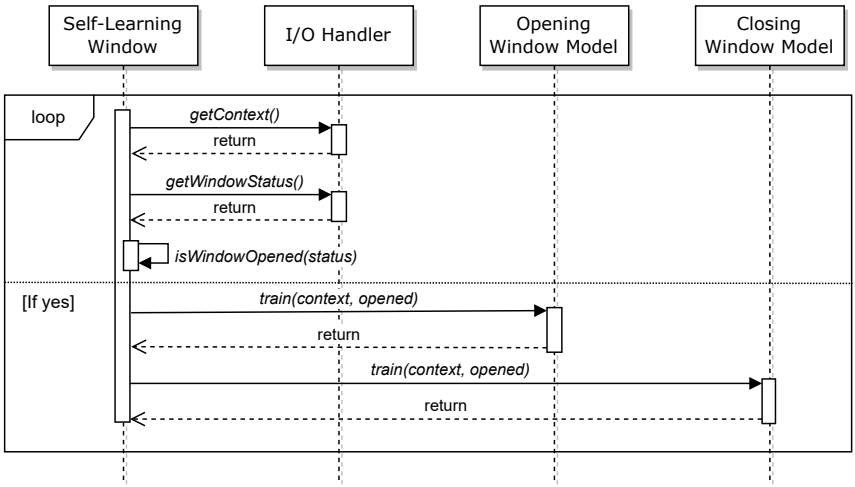


Figure 5.3: Sequence diagram of the process followed by the prototype self-learning window system to learn behavioral patterns involving opening the window. The self-learning window system continuously checks the vehicle's position and the window status using the functions *getContext()* and *getWindowStatus()*. When the window is opened, both the opening and closing models are updated to incorporate the new evidence of the user behavior (via *train(context, action=opened)*). The opening model is rewarded and the closing model is slightly punished to prevent the prototype from opening and closing the window immediately one after the other ([REQ-VI]).

negative feedback on the closing model ensures that both models agree on opening, and not closing, the window under similar circumstances. Similarly, when the window is closed, the opening model is also slightly punished. Furthermore, this mechanism aims to prevent the prototype from opening and closing the window immediately one after the other ([REQ-VI]), as it encodes the assumption that opening and closing are opposite, mutually exclusive actions.

Once both models infer the user behavioral patterns, the decision handler checks whether the current context information belongs to any behavioral pattern, from any of the two models via Equation 4.17. The sequence diagram representing the interaction of the system's components during the decision-making process is illustrated in Figure 5.5. If the current context information belongs to a context region, the decision handler decides which action to take based on the action

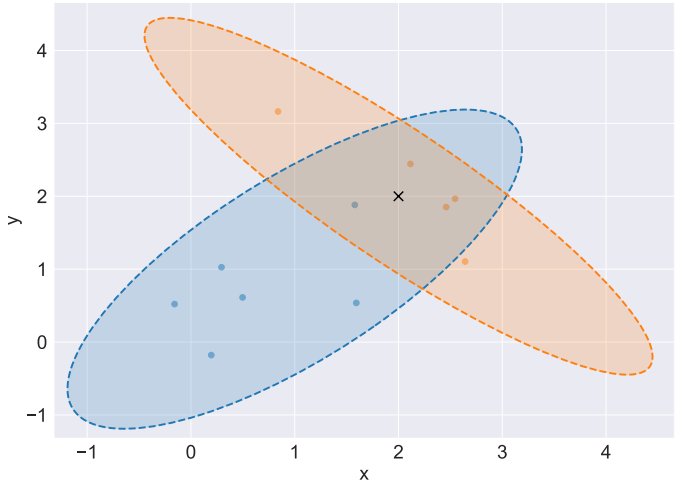


Figure 5.4: Example of overlapping behavioral patterns on a 2-dimensional context space. Blue (orange) points denote data instances where the user has opened (closed) the window. The blue (orange) ellipse denotes the context region of the opening (closing) window behavioral model. At the black cross, the action probability of the opening model would be Beta(7, 1) (mean = 0.88), and the action probability of the closing model would be Beta (6, 1) (mean 0.86), assuming a non-informative prior.

distribution of the corresponding behavioral pattern. The decision is based on the confidence value of the action distribution. If the mean of the action distribution of an *opening window*-model is equal to or greater than a threshold τ_{act}, the system automatically opens the window. Similarly, if the mean of the action distribution of an *closing window*-model is smaller than a threshold τ_{deact}, the system automatically closes the window.

The thresholds are chosen to make the decision handler eager to automate, in order to showcase the self-learning capabilities of the system within the limited evaluation duration ([REQ-I], [REQ-IV]). Therefore, the value of the thresholds is set to $\tau_{act} = 0.8$ and $\tau_{deact} = 0.2$, so that the prototype aims to open (or close) the window automatically when the same action is repeated at least three times under similar circumstances.

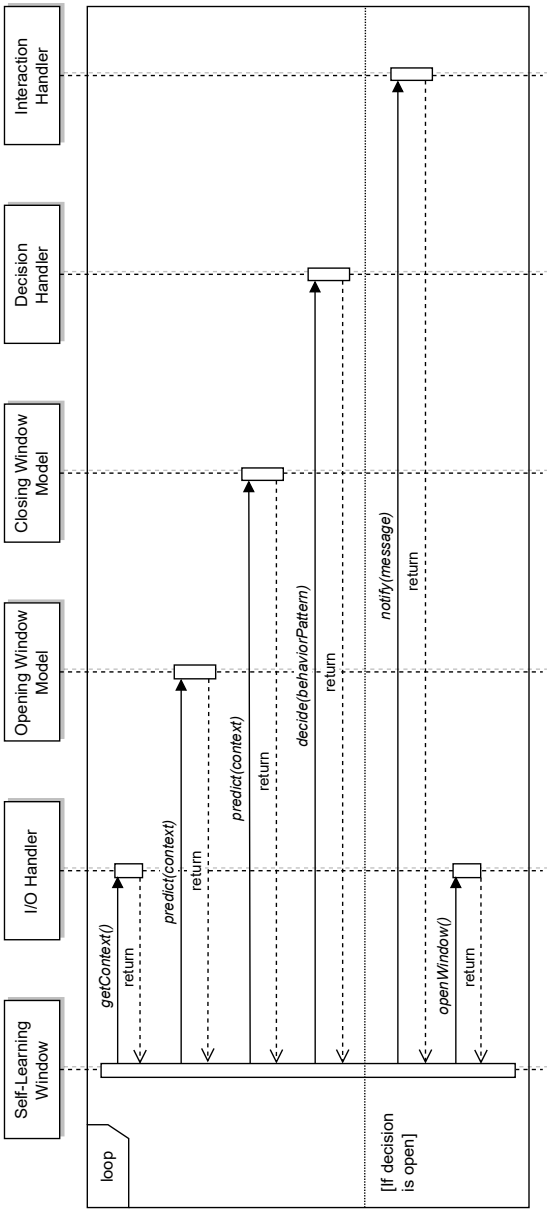


Figure 5.5: Process followed by the prototype self-learning window system to determine its actions. The system continuously collects the vehicle's current location using the function *getContext()*. Utilizing this information, it assesses if any behavioral pattern involving opening or closing the window matches the current context circumstances by calling *predict(context)* on both the opening and closing models, respectively. Then, the decision handler determines the appropriate action based on the reported behavioral patterns (*decide(behaviorPattern)*). If the probability of the user opening the window reaches the activation threshold, a pop-up message is displayed on the user interface to notify the user of the upcoming action (*notify(message)*), and the window is automatically opened (*openWindow()*).

5.3 Results and Discussion

In total, ten employees of Mercedes-Benz AG participated in the experimental evaluation of the self-learning window prototype. Participants were informed about the general goal of the investigation and were requested to drive the vehicle around the Mercedes-Benz Technology Center (MTC) in Sindelfingen. They were instructed to simulate a routine behavior with the power window, by driving the same route at least ten times, and opening or closing the window at least three times at almost the same location. The self-learning comfort system was reset before a new participant entered inside the car.

After the experiment, participants were asked to describe their experience with the self-learning window system. All participants agreed that the automated actions performed by the self-learning window system followed the routines they simulated during the experiment. Also, each participant could share at least one daily life routine where they believed a self-learning window system would be beneficial to them.

However, participants that simulated their behavior at a parking garage reported unexpected behavior of the prototype self-learning window in such conditions. To replicate their behavior at a parking garage, these participants opened and closed the window at nearly the same location, as if they were entering into a parking's garage and briefly opening the window to obtain an entry ticket. These seemingly opposite actions occurred under almost identical conditions, resulting in both behavioral patterns being equally rewarded and punished within practically identical contextual circumstances. This scenario led to uncertainty about the participant's preferences, as the two models suggested distinct actions.

This result indicates that having two different models to identify user's routines might not be the best conflict-solving strategy for a self-learning comfort system. Such configuration is based on the assumption that opening and closing are two mutually exclusive actions, so that when users open the window, they do not want to close it (and hence the closing model is punished), and vice versa. It was seen that such assumption does not hold in all circumstances. Hence, the next concept

of a self-learning comfort system may be based on the idea that "the contrary of opening is the absence of action", for which both opening and closing models should not be related, and the conflict-solving strategy shall be made based on the uncertainty level of the corresponding behavioral patterns. Furthermore, taking into account other context features, such as the time information, or the driving direction of the vehicle might also be beneficial.

Furthermore, four participants mentioned that they sometimes did not expect the automated actions performed by the self-learning window system, as they were not at exactly the same location where they had previously performed the actions, but rather some distance apart from there. A careful review of the behavioral patterns inferred by the self-learning comfort system after the experiments revealed that the user behavioral model tended to increase the covariance of an existing context component rather than create a new one, which occasioned underfitting. As already introduced in section 5.4, this model's reaction is due to the "rich-get-richer" property of Dirichlet processes. For this reason, the user behavioral model was not detailed enough and therefore, generalized user's behavior to circumstances users did not expect. A solution to this phenomenon is presented in Section 5.4.

5.4 Optimized Truncation-free VI Algorithm

Despite all the favorable properties of the nonparametric methods based on the Dirichlet distribution, the Dirichlet process places prior assumptions on the structure of the context components: partitions will typically be dominated by a few very large components, with overall "rich-get-richer" usage. A new observation belongs to a context region with a probability proportional to the number of observations already present in the component, as described in Section 2.3.1.2. Therefore, larger regions have a greater chance to get associated with new observations.

Since the inference algorithm presented in Section 4.2 is initialized with one single component (typically $K = 1$), the "rich-get-richer" effect of the DP can dominate over the geometry of the context regions, causing the DP to become trapped in the undesirable configuration where no new components can be generated [120]. If K is too small, the algorithm may increase the covariance of an existing component rather than creating a new one, and the model complexity would be less than the apparent optimum. This event is known as underfitting. The more underfitted a model is, the less accurately it captures the relationship between the input and output variables and consequently, the more it deviates from the observed data. The opposite of underfitting is overfitting. When a model is overfitted, it fits too closely against the observed data, rendering it unable to generalize well to new data points.

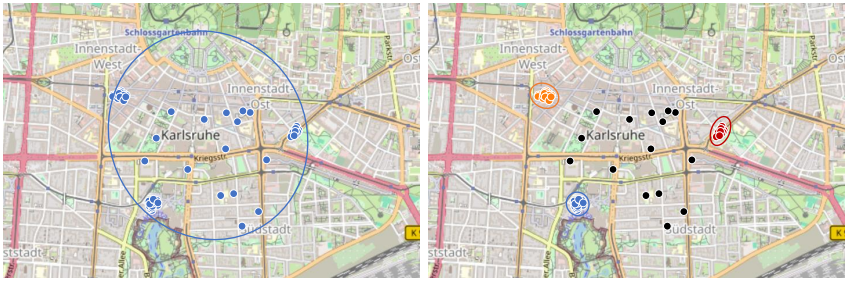
For modeling geographical data, the "rich-get-richer" effect of the DP and the resulting underfitting may cause behavioral patterns that take part in a tiny context region (i.e., at the level of parking lots) go unnoticed if there are data points widespread at larger scales (at the level of cities or countries, for instance), because the model may assign all data points to the same component. This phenomenon is illustrated in Figure 5.6.

If the model deviates too much from the observed data, it may predict future user actions at potentially incorrect locations. Based on such inaccurate predictions, a self-learning window system can, for instance, automatically control the window unexpectedly, which will probably surprise and confuse the user, and will therefore compromise the trust and acceptance of the system [32]. Therefore, for safety relevant comfort systems such as the power windows, it can be necessary to weaken the "rich-get-richer" assumption to favor the accuracy of the model over its generalizing and predicting capabilities, resulting in more detailed context regions. On the other hand, such strategy may not be advisable for other comfort systems whose user acceptance constraints are not so strict, such as the seat heater.

Hence, this dissertation proposes to use a heuristic criterion to adjust the level of detail of the model, that is, the model complexity, in an understandable manner based on the specific requirements of each comfort system. The heuristic criterion

defines the conditions under which the model may have converged to a suboptimal representation of the user behavior, where no new context components can be generated.

Intuitively, the heuristic criterion defines how to detect context component that are "too large" or "too dispersed" with regard to the observed data. If during inference the algorithm identifies a region that is too large, then the model complexity is increased by incrementing the number of components that can be realized, i.e., the value of K. If at the next iteration the newly created component is not assigned to any data point, it will be removed from the model and the value of K will be decreased accordingly. To avoid components to be created in one iteration and removed in the next one when K is increased, $q(\Theta)$ is initialized via the k-means algorithm. The heuristic criterion can be understood as a plug-in criterion to encourage the algorithm to detect more detailed context regions.



(a) **(b)**

Figure 5.6: Effect of the "rich-get-richer" property of the DP on geographical data. Context distributions are represented using 3-sigma confidence ellipses. On the left, the inferred context region is more "spread out", indicating greater variability among the data points assigned to them, or equivalently, a larger covariance matrix. On the right, the context regions have smaller covariance matrices, so that the data points assigned to them are more concentrated and have less variation. Hence, these regions are considered more "detailed" because they exhibit more consistent characteristics.

There are many options to describe how to detect a suboptimal convergence during the inference of the behavioral model. In this dissertation, a context component is considered to be suboptimal, that is, too dispersed, if its generalized variance

(GV) is greater than a threshold ξ (Equation 5.2). The GV is a scalar value which generalizes variance for multivariate random variables, and it is defined as the determinant of the covariance matrix, as indicated in Equation 5.1.

$$\text{GV}(\mathcal{N}(\mu, \Sigma)) = det(\Sigma) \tag{5.1}$$

The generalized variance is related to the multidimensional scatter of points around their mean, as demonstrated in [121]. As the GV increases, the data points exhibit greater dispersion around the mean. For instance, in a 2-dimensional (or 3-dimensional) space, when $GV = 0$, the data points lie on a line (or plane). Consequently, the GV serves as a measure that quantifies the area (or volume) occupied by the data points, as noted by [122]. Similar heuristic threshold tests have been applied in other research works to determine cluster creation and pruning ([123], [124]).

$$\text{GV}(\mathcal{N}(\mu, \Sigma)) > \xi \tag{5.2}$$

The pseudocode describing the algorithm for the inference of the behavioral patterns and context regions with the heuristic criterion is described in Appendix B.

5.4.1 Algorithm Demonstration

This demonstration of the user behavioral model is built on the realistic example presented in Section 4.3. However, this section employs the truncation-free VI algorithm with heuristic thresholding presented in the previous section to analyze Alice's behavior with the window, rather than applying the baseline truncation-free VI presented in Section 4.2. An example of the learned context regions using the heuristic-based inference algorithm ($\xi = 0.1$) is shown in Figure 5.7, in which the context distributions are represented by the 3-sigma confidence ellipses.

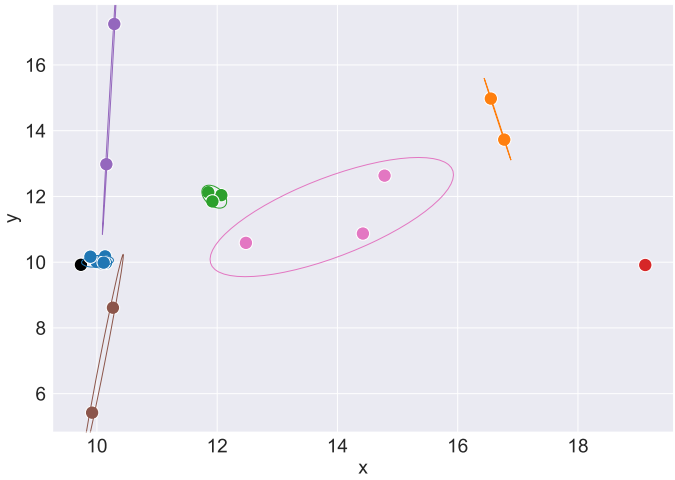


Figure 5.7: Toy data set and context components inferred with the heuristic-based algorithm ($\xi = 0.1$). The heuristic criterion allows to control the desired level of detailed of the learned context regions. The smaller the value of ξ, the less variability in the assigned data.

The optimized heuristic-based VI inference algorithm is trained 100 times, each iteration presenting the data points of the data set in a random order. To show the impact of the heuristic threshold ξ on the inferred context regions, Figure 5.8 shows the value of the maximum number of behavioral patterns at each training time step, K, for different values of the heuristic threshold $\xi = \{0.01, 0.1, 1, 10\}$, averaged over the 100 iterations. Furthermore, the results obtained with the truncation-free VI without heuristic thresholding are also illustrated in Figure 5.8, indicated as $\xi = None$.

When $\xi \to \infty$, the value of the truncated number of components K at every training time step estimated by the heuristic-based inference algorithm is very similar to the value estimated by the first non-heuristic algorithm. As expected, the greater the value of ξ, the less likely it is that the suboptimal convergence test defined in Equation 5.2 is applied. Therefore, the value of the maximum number of context regions, K, is not increased. On the other hand, if $\xi \to 0$, the

suboptimal test always applies, which results in an increase of the value of K at almost every training time step.

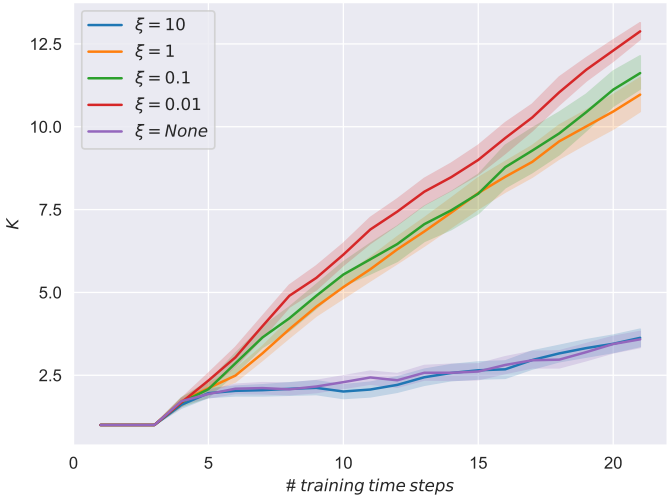
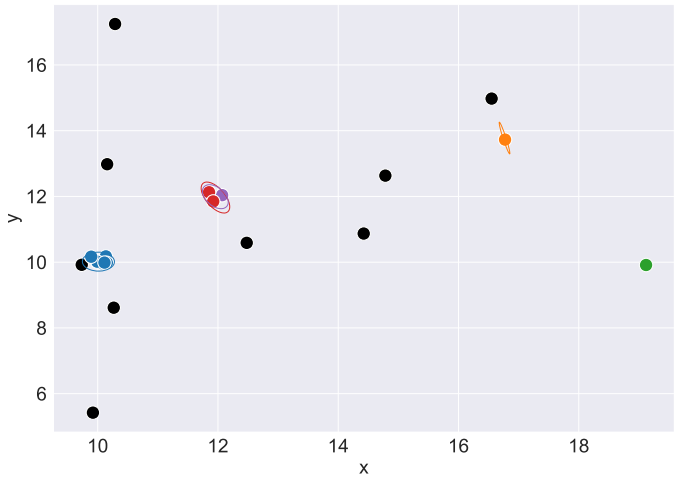


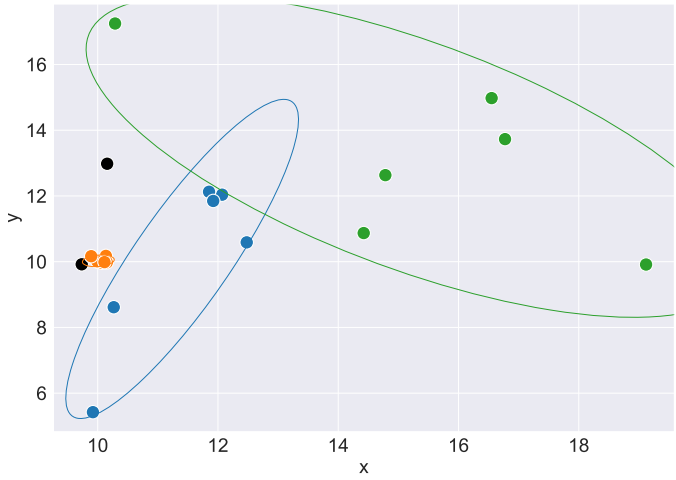
Figure 5.8: Number of components inferred at every training step for different values of ξ. Lower K means simpler models. $\xi = None$ indicates the results obtained from the first truncation-free VI algorithm.

Results reveal some other interesting properties of the optmized truncation-free VI algorithm for the user behavioral model. Firstly, it can be observed that the context regions learned with the baseline truncation-free VI algorithm (Figure 4.6) do not fit the data points so precisely as the heuristic-based VI algorithm (Figure 5.7). The reason for this behavior is the "rich get richer" property of the DP, by which new data points have greater chance to get associated with an already existing large component than to start a new one. The heuristic check of the heuristic-based inference algorithm overcomes the "rich-get-richer" assumption and therefore, smaller context regions are identified. Secondly, the model complexity, represented by the value of K, depends on the value of the heuristic threshold ξ. The greater the value of ξ, the less the model complexity (i.e., smaller value of K). Consequently, the smaller the value of ξ, the more level

of detail the model has. This property can be observed comparising the resulting context regions obtained for $\xi = 0.01$ and $\xi = 10$ in Figures 5.9a (more detailed) and 5.9b (more spread out), respectively.



(a)



(b)

Figure 5.9: Context regions inferred using the truncation-free VI algorithm with heuristic thresholding, with $\xi = 0.01$ (top) and $\xi = 10$ (bottom).

6 Analysis of the User Behavioral Model

The synthetically generated datasets presented in this section aim to illustrate the properties of the user behavioral model. Since the user behavioral model is composed by the context and the action distributions, this problem has been decomposed into two smaller and independent sub-problems. Firstly, in Section 6.1, the properties of the context distribution are examined. Secondly, Section 6.2 analyzes the action distribution.

6.1 Analysis of the Context Distribution

Four datasets have been generated to examine the model's ability to cope with randomly changing context circumstances that govern user actions: *orbital*, *noisy*, *overlapping* and *vanishing*, described in sections 6.1.2.1, 6.1.2.2, 6.1.2.3 and 6.1.2.4, respectively. Each dataset has been designed to represent a single challenging task, isolated from the others.

Data points in the datasets represent the context circumstances recorded when an action on a vehicle comfort system is performed. The context distributions of the synthetically generated behavioral patterns are normally distributed, and samples of the same Normal distribution are assumed to belong to the same behavioral pattern.

For straightforward performance evaluation, it is assumed that the action performed is always an activation of a comfort system ($a_n = 1$). Moreover, the

means of the synthetically generated context distributions are not permitted to overlap in feature space, helping to ensure that the underlying model is recoverable, thus providing a fair comparison of model outputs to ground truth distributions. Furthermore, to ease the visualization of the model's solution, datasets are 2-dimensional.

6.1.1 Methodology and Evaluation Metrics

Five configurations of the user behavioral model are evaluated for each problem. To illustrate the effects of the heuristic threshold in the model's performance, four of the configurations utilize the optimized heuristic truncation-free variational inference method, each of which employs a different value of the threshold parameter $\xi = \{0.01, 0.1, 1, 10\}$. The fifth model configuration employs the first truncation-free VI approach, without heuristic criterion ($\xi = None$).

Moreover, since the heuristic criterion depends on the model's covariance matrix (Equation 5.2), which is regulated by the scale matrix Ψ, the five model configurations are evaluated considering three distinct values of the hyperparameter Ψ, namely $\Psi = \{0.1\boldsymbol{I}, \boldsymbol{I}, 10\boldsymbol{I}\}$. The value of the other prior hyperparameters of the user behavioral model presented in Section 4.1.3 are listed in Table 6.1.

Hyperparameter	Value
α	1
$\boldsymbol{\mu_0}$	$\mathbf{0}$
κ	10^{-6}
$\boldsymbol{\Psi}$	$\{0.1\boldsymbol{I}, \boldsymbol{I}, 10\boldsymbol{I}\}$
ν	$d+1$
β_0	1
α_0	1

Table 6.1: Hyperparameters for the analysis of the context distribution on synthetic data (feature dimension $d = 2$). A vector of d zeros is denoted by $\mathbf{0}$.

The user behavioral model is evaluated for label agreement with ground truth behavioral patterns via Normalized Mutual Information (NMI) [125]. NMI is computed for two data partitions U and V from the following equation:

$$NMI(U,V) \triangleq \frac{\mathbb{I}(U;V)}{(\mathbb{H}(U)+\mathbb{H}(V))/2}, \tag{6.1}$$

where $\mathbb{I}(U;V)$ is the mutual information between U and V, and $\mathbb{H}(U)$, $\mathbb{H}(V)$ are the entropies of U and V, respectively.

The NMI score lies between 0 and 1. NMI has a maximum value of 1 when the solutions U and V are identical. Conversely, when the value of NMI is 0, U and V are mutually independent. Thus, the larger the NMI, the better the performance.

Furthermore, the purity of the model's solution is also evaluated. The purity score, also known as homogeneity, evaluates the extent to which a group of records shares the same class. To compute the purity score, each context region is assigned to the most frequent ground truth label of the data points assigned to it. Then, the accuracy of this assignment is measured by counting the number of correctly assigned data points, divided by the number of total data points. Formally, purity is defined as follows:

$$purity(U,V) \triangleq \frac{1}{N}\sum_{u\in U}\max_{v\in V}|u\cap v|, \tag{6.2}$$

where U is the set of context regions, V refers to ground truth labels of the dataset, and N is the total number of data points.

The purity score ranges between 0 (bad) and 1 (good). It does not penalize for the number of components, so one can trivially achieve a purity of 1 by putting each data point into its own component. Therefore, the error in the number of components is also analyzed. It is measured by the signed deviation from the true number of components $\Delta K = \widehat{K} - K$, where $\widehat{K}$ and K are the estimated and

true number of components, respectively. When $\Delta K > 0$ ($\Delta K < 0$), the model estimated too many (few) components.

Finally, the total training time in logarithmic scale is also provided, so that the different tuning schemes can be compared in terms of computational complexity.

6.1.2 Evaluation of the Synthetic Datasets

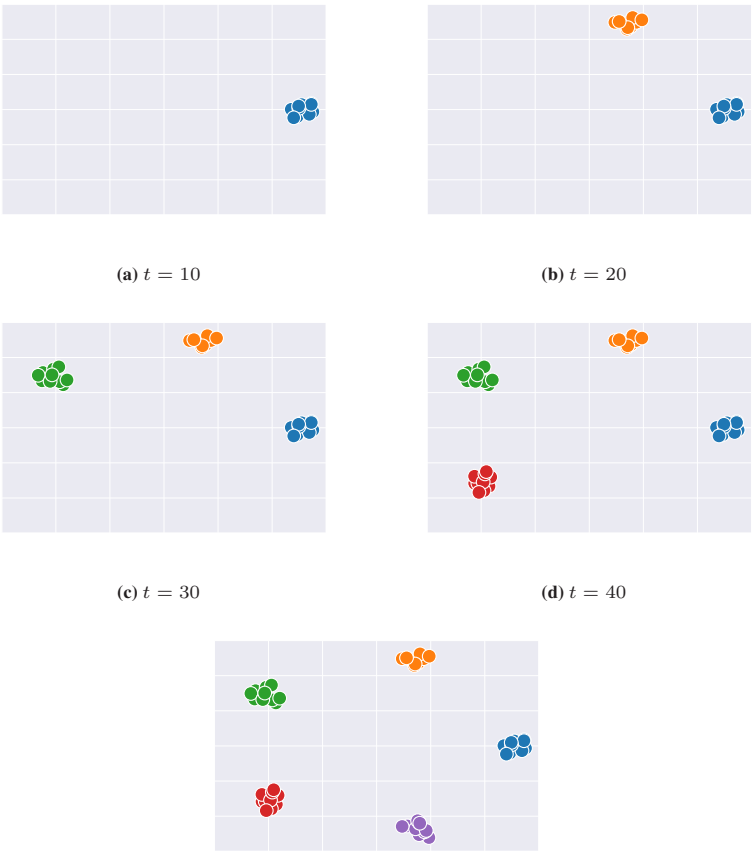
6.1.2.1 Orbital Dataset

An *orbital* dataset represents the context circumstances of many, non overlapping, behavioral patterns. This problem is specially interesting because users frequently have several behavioral patterns, and the user behavioral model must be capable of recognizing all of them. For example, a user may open the window at many different locations, such as her garage, her office's parking or at a shopping mall. Orbital datasets lay the foundation for the generation of the other datasets presented in the following sections.

Data points in an orbital dataset are represented as samples from $C = 5$ normally distributed components, whose means are evenly spaced on a circle of radius $R = 5$ around the origin and covariance matrix $\boldsymbol{\Sigma}(x) = 0.1\boldsymbol{I}$, so that components' means and data points do not overlap.

At each time step t, a new data point is generated by sampling one of the normal components. Once $P = 10$ samples have been taken from one component, the next component is sampled, chosen at random. This is repeated until P data points from each of the C components have been taken. Figure 6.1 shows an example of an orbital dataset.

This type of dataset stresses the model, and specially the context distribution, in the following ways:



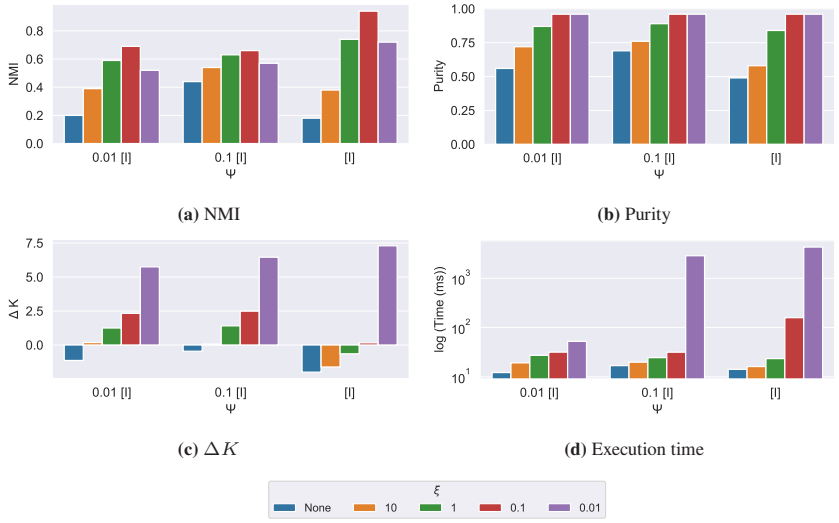
(a) $t = 10$ **(b)** $t = 20$ **(c)** $t = 30$ **(d)** $t = 40$ **(e)** $t = 50$

Figure 6.1: Example data from an orbital dataset at different time steps t, for $C = 5$ and $P = 10$. Once P points have been drawn from one component, the next component is sampled. Components are colored to facilitate their identification.

- The model must be able to compute and update the parameters of the context distribution of the behavioral patterns that are changing as a function of time.

- The model must be able to add behavioral patterns according to the data.
- The model must be able to identify behavioral patterns with changing sizes.

Models are evaluated averaging 100 Monte-Carlo experiments. Results indicating the model's performance for $\Psi = \{0.01\boldsymbol{I}, 0.1\boldsymbol{I}, \boldsymbol{I}\}$ are shown in Figure 6.2.



(a) NMI **(b)** Purity

(c) ΔK **(d)** Execution time

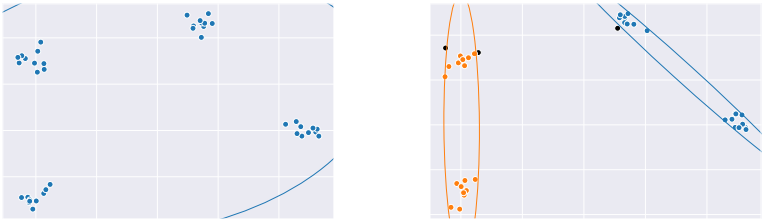
Figure 6.2: Performance metrics for each of the model configurations when tested on the orbital dataset. Metrics were computed for each time step t and then averaged across all Monte-Carlo experiments.

An interesting result is that the average NMI and purity scores obtained by the inference approach without heuristic criterion ($\xi = None$), are lower than the ones obtained by the other four model configurations with the optimized heuristic inference, regardless of Ψ. This result suggests that models inferred using the heuristic criterion are more similar to the ground truth behavioral model than the baseline approach. Moreover, it can also be seen that the baseline model generally underestimates the number of context components (i.e., $\Delta K < 0$). This behavior is expected from standard DP implementations, due to the "rich-get-richer" effect

presented in section 5.4. The model configurations that leverage the heuristic inference algorithm perform better by detecting large context components in the data stream and re-fitting the model with a better value of K.

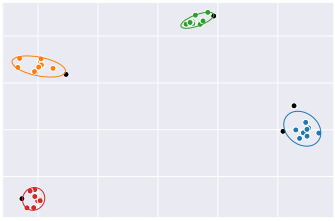
Figure 6.2 also reveals that smaller values of the heuristic threshold ξ lead to a higher overestimation of the number of context components ($\Delta K > 0$) and to more homogeneous components (higher purity). This is due to the fact that, when suboptimal components are detected by the heuristic criterion, the model's number of components K is increased, increasing the model's complexity and hence allowing the model to find more, but smaller, context components. However, very small values of ξ occasion an unnecessary increment of the model's complexity, which is explained by the reduction of the NMI score in models with $\xi = 0.01$, as compared with models with $\xi = 0.1$.

Furthermore, the heuristic detection induces an increase in the average execution time. If the value of the heuristic threshold ξ is smaller than the variance of the data, the execution time is increased more than 10 times, as suboptimal components are constantly detected, leading to frequent model re-fitting.

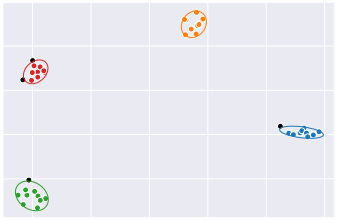


(a) $\xi = None$

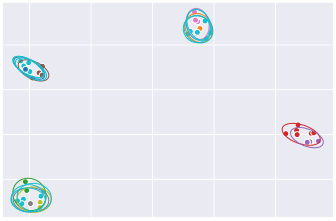
(b) $\xi = 10$



(c) $\xi = 1$



(d) $\xi = 0.1$

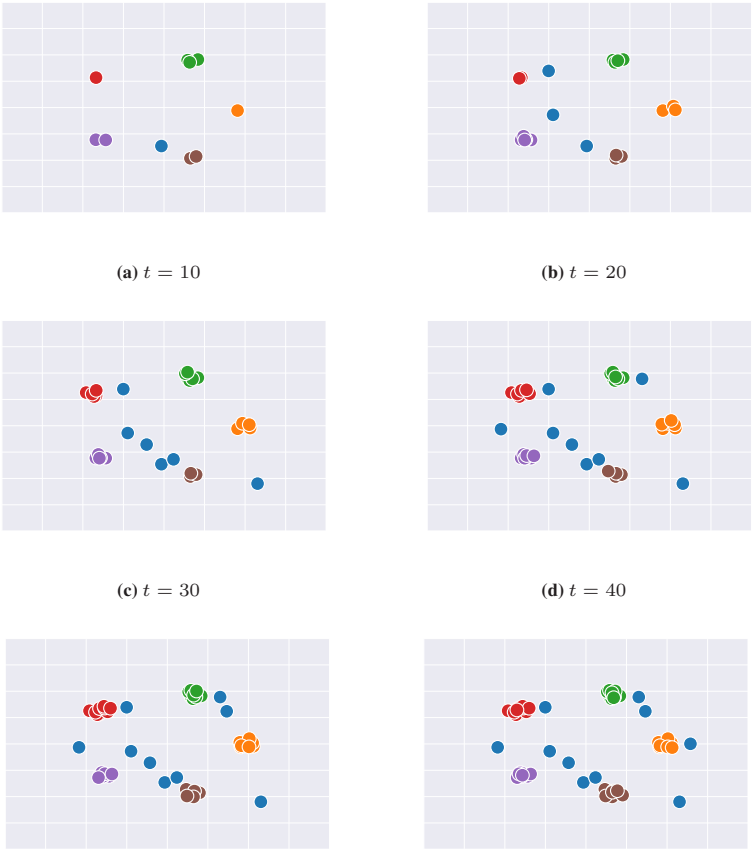


(e) $\xi = 0.01$

Figure 6.3: Context components inferred by the different model configurations on an orbital dataset at $t = 40$ with $\Psi = \boldsymbol{I}$.

6.1.2.2 Noisy Dataset

A *noisy* dataset is an *orbital* dataset that has $N = 10$ additional data points, which seem not to belong to any context region. The N data points pretend to imitate accidental or not frequently performed user actions. For the model, these N points represent noise. An example of a noisy dataset is illustrated in Figure 6.4.



(a) $t = 10$ **(b)** $t = 20$

(c) $t = 30$ **(d)** $t = 40$

(e) $t = 50$ **(f)** $t = 60$

Figure 6.4: Example data from a noisy dataset at different time steps t, for $C = 5$, $P = 10$ and $N = 10$. The N points (blue) pretend to imitate accidental or not frequently performed user actions.

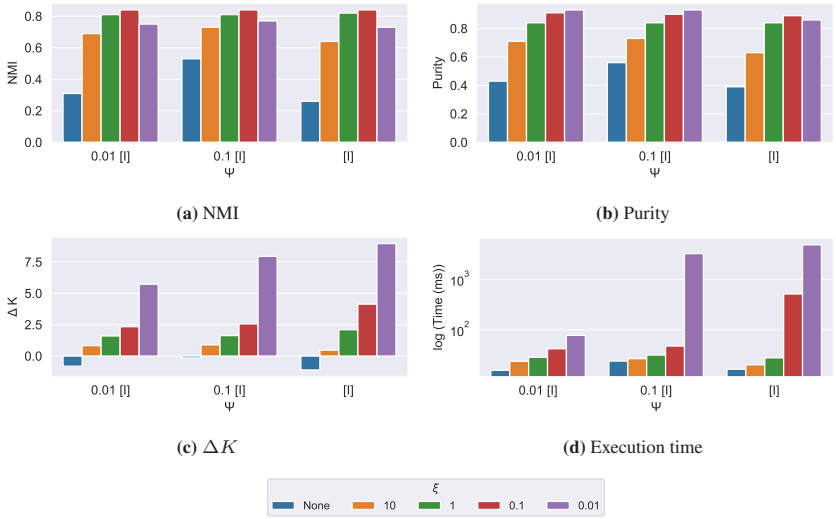
The noise data points are generated by sampling N times a Normal distribution with mean $\mu = (0, 0)$ and covariance matrix $\mathbf{\Sigma} = 10\boldsymbol{I}$, which is 100 times larger than the data generating distributions. At each time step t, a sample from any of

the Normal components is taken, chosen at random. In this example, the data points of the N noise components may lie over the data points of any of the K Normal components.

This type of dataset stresses the model, and specially the context distribution, in the following ways:

- The model must be able to identify behavioral patterns in the presence of noise.

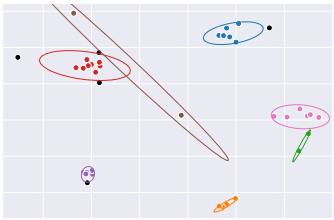
Models are evaluated averaging 100 Monte-Carlo experiments. Results are shown in Figure 6.5. The results obtained by the model in this experiment are, in general, very similar to the results obtained in the previous section. However, it can be observed that the purity scores are lower in this experiment. This is due to the fact that, in noisy datasets the data points of the N components may overlap the data points generated by the other K Normal components, which makes inevitable that models find less homogeneous context components. Moreover, models are more likely to overestimate the number of context components ($\Delta K > 0$) in this second experiment, because models tend to group the noise data points N either together or in many smaller components, as illustrated in Figure 6.6.



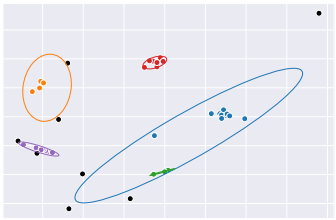
(a) NMI **(b)** Purity

(c) ΔK **(d)** Execution time

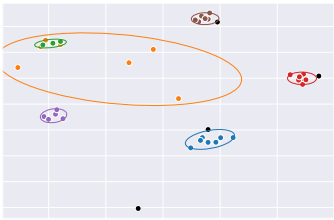
Figure 6.5: Performance metrics for each of the model configurations when tested on the noisy dataset. Metrics were computed for each time step t and then averaged across all Monte-Carlo experiments.



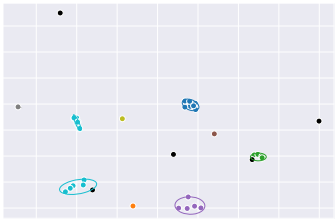
(a) $\xi = None$



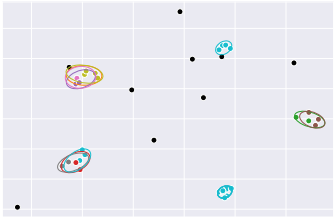
(b) $\xi = 10$



(c) $\xi = 1$



(d) $\xi = 0.1$



(e) $\xi = 0.01$

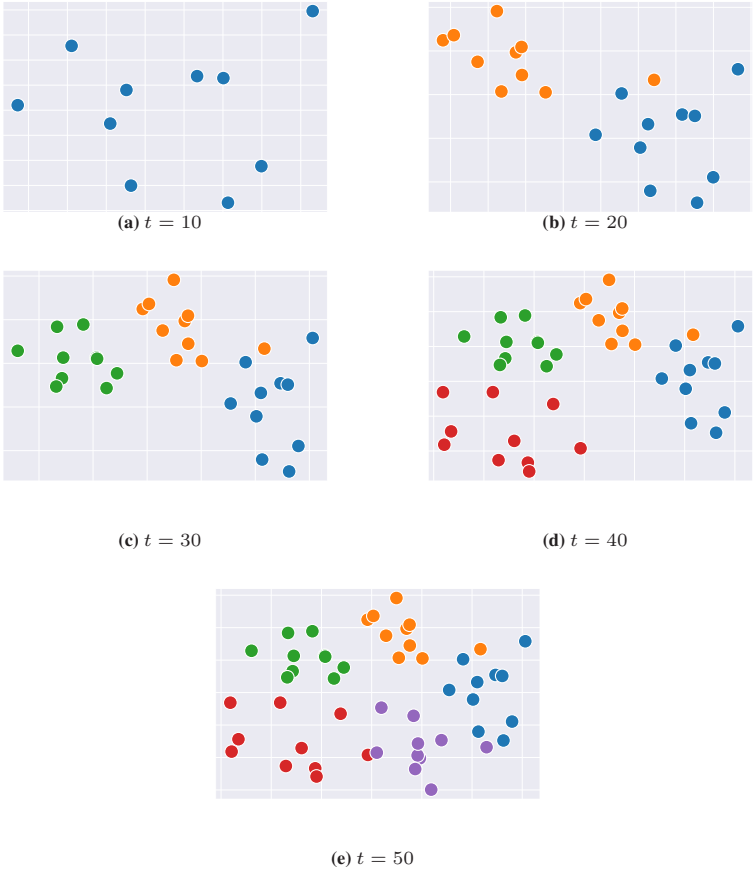
Figure 6.6: Context components inferred by the different model configurations on a noisy dataset at $t = 50$ with $\Psi = \boldsymbol{I}$.

6.1.2.3 Overlapping Dataset

An *overlapping* dataset is an *orbital* dataset in which the samples of the C normally distributed components overlap, as illustrated in Figure 6.7. This is accomplished by placing the means of the components over a circle of radius $R = 1$, instead of $R = 5$. The purpose of this problem is to analyze the performance of the user behavioral model when samples are close to each other in the context space.

Models are evaluated by averaging 100 Monte-Carlo experiments. Results in Figure 6.8 show that model configurations with smaller values of $\xi = \{0.1, 0.01\}$ obtain better NMI and purity scores. However, they also tend to overestimate the number of behavioral patterns, leading to longer execution times. These results are particularly noticeable in the model configuration with $\xi = 0.01$, as exemplified in Figure 6.9e. Since the threshold value for detecting suboptimal context regions (i.e., regions that are too large and dispersed) is significantly smaller that the underlying variance of the components ($\Sigma = 0.1\boldsymbol{I}$), the model increases its complexity by incrementing the upper limit of context components K at each iteration. The heuristic criterion is likely to be met repeatedly, leading to the model assigning a single data point to a different component.

On the other hand, model configurations with $\xi = \{None, 10, 1\}$ tend to underestimate the number of components ($\Delta K < 0$), which indicates that data points of different components are grouped together, as can be seen in the example showed in Figure 6.9a, 6.9b, and 6.9c, respectively. These results are motivated by the "rich-get-richer" property of the DP, by which the DP tends to assign data points to components that already have many data points. In general, the results obtained from the analysis of the overlapping datasets suggest that, if overlapping behavioral patterns are expected, the value of ξ shall be equal to or smaller than the variance of the expected context components.

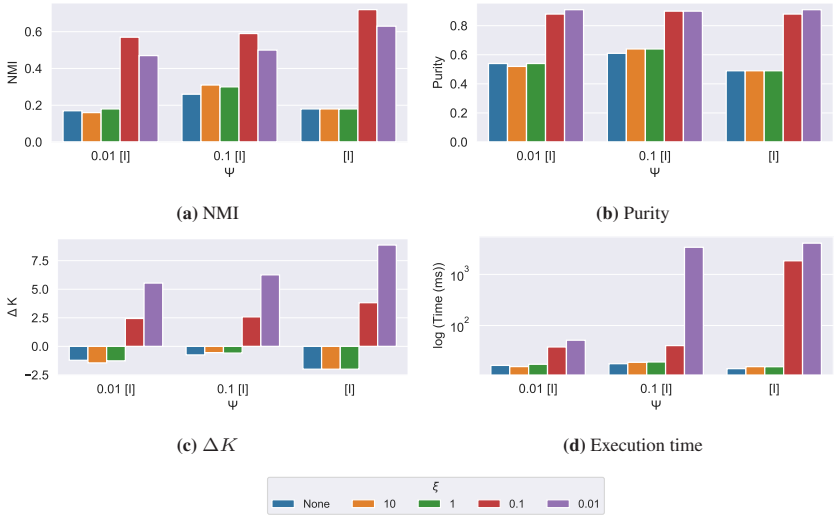


(a) $t = 10$ **(b)** $t = 20$

(c) $t = 30$ **(d)** $t = 40$

(e) $t = 50$

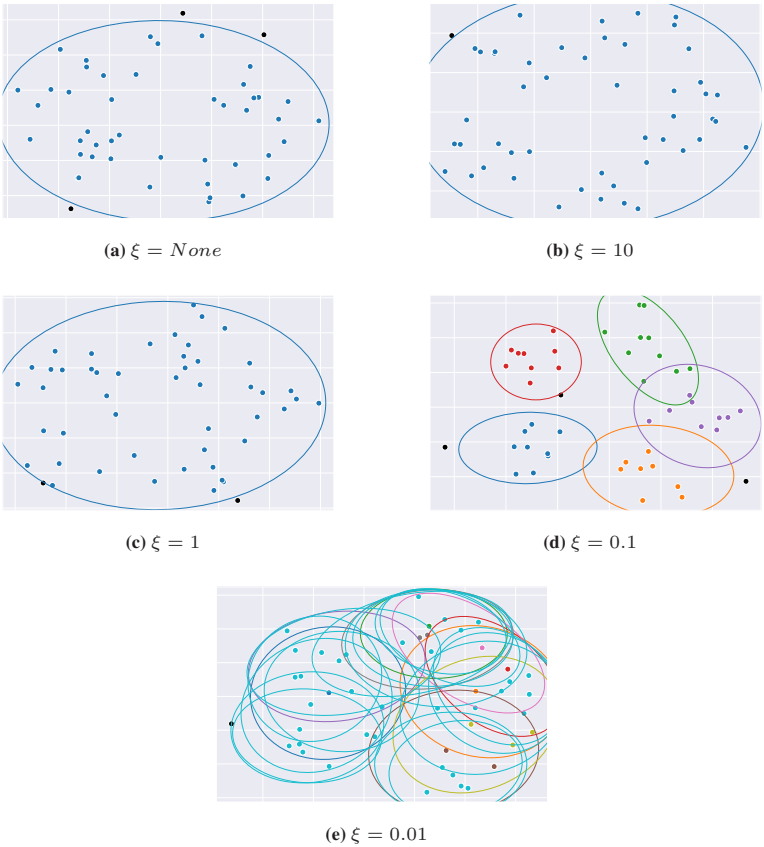
Figure 6.7: Example data from an overlapping dataset at different time steps t, for $C = 5$ and $P = 10$.



(a) NMI **(b)** Purity

(c) ΔK **(d)** Execution time

Figure 6.8: Performance metrics for each of the model configurations when tested on the overlapping dataset. Metrics were computed for each time step t and then averaged across all Monte-Carlo experiments.



(a) $\xi = None$ **(b)** $\xi = 10$

(c) $\xi = 1$ **(d)** $\xi = 0.1$

(e) $\xi = 0.01$

Figure 6.9: Context components inferred by the different model configurations on an overlapping dataset at $t = 50$ with $\Psi = \boldsymbol{I}$.

6.1.2.4 Vanishing Dataset

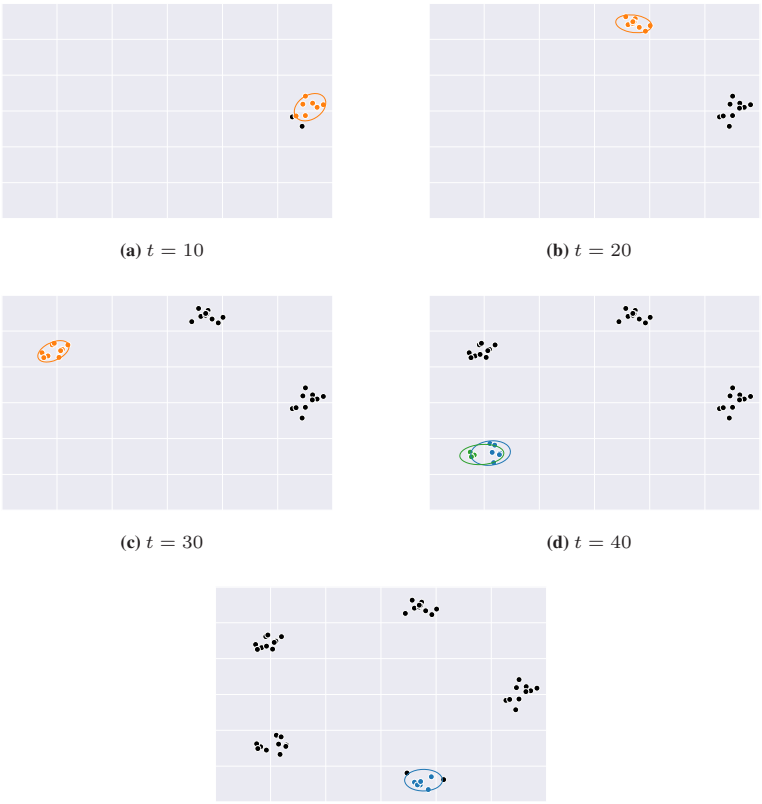
A *vanishing* dataset is an *orbital* dataset. However, in this problem, the window length of the model's sliding window mechanism is set to $W = 10$. Hence, the user behavioral model is forced to forget data points older than $t = 10$ time steps. This problem is specially interesting to analyze the performance of the "forgetting mechanism" of the user behavioral model. In particular, it analyzes how the context distributions are affected when users modify their routines. For example, a user may activate the seat heater in winter, but as soon as it becomes warmer in summer, the user may not want to heat up the seat anymore. Hence, the user behavioral model must adapt the learned behavioral patterns accordingly: the associated context distribution may slowly disappear, and the action distribution may reveal an increased uncertainty on the user's behavior and ultimately, discourage any automated activation of the seat heater.

This type of dataset stresses the model, and specially the context distribution, in the following ways:

- The model must be able to compute and update the parameters of behavioral patterns that are changing as a function of time.
- The model must be able to remove behavioral patterns according to the data.
- The model must be able to identify behavioral patterns with changing sizes.

Results in Figure 6.11 show that the values of the NMI and purity metrics are very similar to the outcomes obtained for the orbital dataset in Section 6.1.2.1. However, in this vanishing problem the model's execution time is considerably reduced. This is because in the vanishing example, models identify less context components compared to the orbital dataset, which is due to the fact that the window length has been reduced to 10.

An example of the inferred components by a user behavioral model with $\xi = 0.1$ and $\Psi = \boldsymbol{I}$ is shown in Figure 6.10.



(a) $t = 10$ **(b)** $t = 20$ **(c)** $t = 30$ **(d)** $t = 40$ **(e)** $t = 50$

Figure 6.10: Context components inferred by the user behavioral model with $\xi = 0.1$ and $\Psi = \boldsymbol{I}$ on a vanishing dataset with $C = 5$, $P = 10$ and $W = 10$.

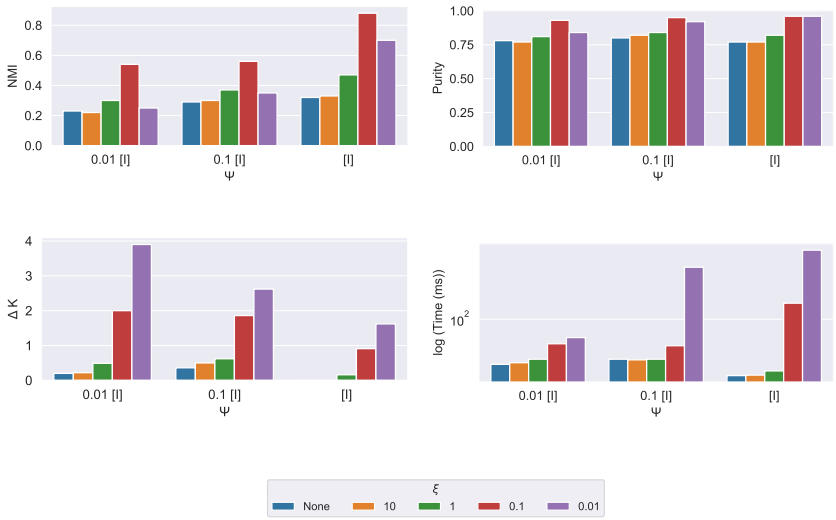


Figure 6.11: Performance metrics for each of the model configurations when tested on the vanishing dataset. Metrics were computed for each time step t and then averaged across all Monte-Carlo experiments.

6.2 Analysis of the Action Distribution

This section analyzes the influence of the number of user actions and the weigh of the positive and negative indirect user feedback on the action distribution, and specially on the model's uncertainty of future user actions, measured by its mean and standard deviation.

The synthetically generated datasets aim to represent different kinds of user behavior: either constant and invariable or irregular and changing. The samples in the datasets represent the actions performed by a hypothetical user. In particular, each sample describes whether the user activated ($a_n = 1$) or missed activating ($a_n = 0$) the comfort functionality. These two kinds of user interactions with the comfort functionality represent a positive indirect and a negative indirect feedback, respectively, and are specially interesting because they can be recorded in an offline experimental setting, as the one described in Chapter 7.

The focus of this analysis is to gain understanding of the action distribution of the user behavioral model, as the performance of the context distribution has been already analyzed in the previous chapter. Therefore, the datasets presented in this section contain information regarding the user actions only, and do not describe the context in which the actions are performed. It is assumed that all data points in the datasets belong to the same behavioral pattern.

6.2.1 Methodology and Evaluation Method

Data points are generated by drawing C samples from a uniform distribution. If the value of the sample is equal to or smaller than a threshold $T \in [0, 1]$, the value of the data point is set to 1 (activation). Otherwise, the data point is set to 0 (missed activation). Hence, the greater the value of the threshold T, the more often activation samples are generated. That is, datasets generated with T close to 1 symbolize the behavior of a user, who regularly follows the same routine. On the other hand, datasets generated with T close to 0 represent the behavior of a user with an irregular conduct, who seems not to follow any particular routine.

For each dataset generated with C samples and threshold T, three values of the positive and negative feedback weights are evaluated. The higher the value of the positive indirect feedback weight w_{pos_i}, the greater the model's inclination to believe that every user action follows a routine. In other words, higher values of w_{pos_i} encourage the model to believe that the user's behavior is constant and stable. On the other hand, the higher the value of the negative indirect feedback weight w_{neg_i}, the more the model's confidence on future user actions is reduced when an expected action is not observed. That is, the higher the value of w_{neg_i}, the more cautious and conservative the learned user behavioral patterns will be.

For all problems represented in the datasets, the user behavioral model is configured with the parameter values listed in Table 6.2. Behavioral patterns are inferred using the truncation-free variational inference method without heuristic ($\xi = None$). The value of the hyperparameters α_0 and β_0 of the action distribution at start are chosen to represent no strong prior beliefs about the user's preferences, so that any value between 0 and 1 is equally likely before observing any data. Hence, their values are set to $\alpha_0 = \beta_0 = 1$, which is known as a non-informative prior distirbution.

Hyperparameter	Value
α	1
$\boldsymbol{\mu_0}$	$\mathbf{0}$
κ	10^{-6}
$\boldsymbol{\Psi}$	$\boldsymbol{I}$
ν	$d+1$
β_0	1
α_0	1

Table 6.2: Hyperparameters for the analysis of the action distribution on synthetic data (feature dimension $d = 1$). A vector of d zeros is denoted by $\mathbf{0}$.

The model's uncertainty of future user actions is evaluated by analyzing the value of the mean and standard deviation (σ^2) of the resulting action distribution after training. The higher the value of the mean, the higher the probability of a future user action. Model uncertainty is measured by the standard deviation of the action distribution. The smaller the standard deviation, the less uncertainty and, accordingly, the more accentuate the prediction would be.

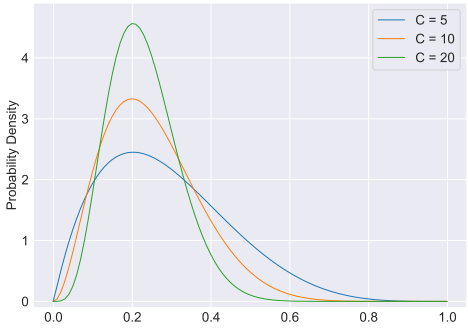
6.2.2 Results and Discussion

The action distribution is evaluated by averaging 100 Monte-Carlo experiments for datasets generated with $C = [5, 10, 15]$, $T = [0.2, 0.5, 0.8]$ and different values of the positive and negative indirect feedback weights. An overview of the various configuration options evaluated and the resulting mean and standard deviations of the trained action distribution is presented in Table 6.3.

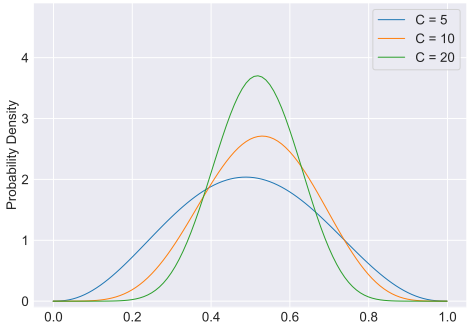
Based on the results, it appears that increasing the number of samples (represented by a higher value of C) leads to a decrease in the standard deviation of the action distribution. This suggests that as more evidence is gathered, uncertainty regarding future user actions is reduced. Furthermore, it can be observed that the more consistent the user behavior is (higher T), the more probable is that the user performs the same action in the future, denoted by a higher mean and a smaller standard deviation. Finally, results indicate that the feedback weights influence the model's learning rate. Independently of the number of samples C, it holds that the higher the positive indirect feedback weight (w_{pos_i}), the higher the model's probability on a future user action (greater mean). Also, the higher the negative indirect feedback weight (w_{neg_i}), the less the model's probability on a future user action (smaller mean). Hence, the higher the value of the positive indirect feedback weight of a model, the less evidence (samples) it will need to believe that the user has a routine behavior and will tend to perform an action in the future.

		$w_{pos_i} = 1$ $w_{neg_i} = 1$		$w_{pos_i} = 2$ $w_{neg_i} = 1$		$w_{pos_i} = 1$ $w_{neg_i} = 2$	
T	C	**mean**	σ^2	**mean**	σ^2	**mean**	σ^2
0.2	5	0.29	0.16	0.38	0.16	0.18	0.11
	10	0.25	0.12	0.38	0.12	0.15	0.08
	20	0.23	0.09	0.33	0.09	0.13	0.05
0.5	5	0.49	0.18	0.65	0.15	0.37	0.15
	10	0.52	0.14	0.63	0.11	0.34	0.11
	20	0.52	0.10	0.66	0.08	0.34	0.08
0.8	5	0.72	0.16	0.83	0.11	0.60	0.16
	10	0.74	0.12	0.85	0.08	0.66	0.12
	20	0.78	0.09	0.87	0.05	0.66	0.09

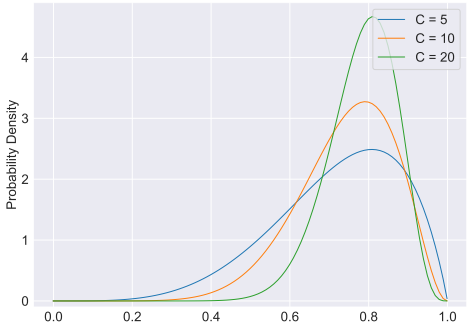
Table 6.3: Results of the analysis of the action distribution on synthetic data. The mean and variance values of the action distributions inferred by the user behavioral model in each experiment are averaged for the 100 Monte-Carlo simulations.



(a) $T = 0.2$



(b) $T = 0.5$



(c) $T = 0.8$

Figure 6.12: Probability density function of the action distribution inferred by the user behavioral model for $T = \{0.2, 0.5, 0.8\}$ and $w_{pos_i} = w_{neg_i} = 1$.

7 Experimental Evaluation of the User Behavioral Model

Contrary to the evaluation performed in Chapter 6, which focused on comparing the results of the user behavioral model to a ground truth and allowed quantitative comparisons between algorithm configurations using synthetic data, this section presents an exploratory demonstration of the user behavioral model. The user behavioral model is used to extract behavioral patterns from four different datasets, each containing data of a different user. In this section, the user behavioral model acts as a lens through which the user behavior is examined, since the number and nature of the behavioral patterns of users are unknown in such real-world settings.

This experiment aims to evaluate the user behavior with two vehicle comfort functions: the power windows and the seat heater. In particular, the user behavioral model is employed to reveal the situations in which users frequently opened the power window and turned on the seat heater at its highest intensity level.

7.1 Dataset Description

The user behavioral model relies on personal data, such as location and time, to be able to find patterns in the behavior of a user. Unfortunately, datasets with user personal data in the vehicular domain are very difficult to find. Even if some companies and institutions may collect personal user data, they do not publish them for privacy and confidentially reasons. Therefore, a major achievement in this dissertation is to collect user personal data in the vehicle comfort domain in order to be able to evaluate the proposed user behavioral model.

Nevertheless, to ensure the privacy and confidentiality of the individuals involved in this study, certain personal information, particularly geographic locations, have been deliberately blurred or obscured in the figures presented in this chapter.

7.1.1 Data Collection

Activities to collect user data were carried out from October 2020 to May 2021. In total, four users participated in the experiment. They were informed about the nature of the collected data, and were asked to act normally during the whole data collection process. Participants were able to stop the data collection process at any time, and they were encouraged to stop recording if they were not driving the vehicle. At the time of the experiment, there were coronavirus-related restrictions to slow down the spread of the COVID-19 disease, which might have had an impact on the users' behavior and routines.

Participants' data were extracted from the vehicles using data loggers, which sampled the information of controller area network (CAN) buses at 10 Hz. The selection process of the features of the dataset followed the instructions provided in [53], which suggests that, if domain knowledge is at hand, it should be utilized. After a careful review with experts in the vehicle comfort domain at Mercedes-Benz, the features that were expected to have the greatest influence on users' behavior with the seat heater and power window functions were selected. Table 7.1 provides the list of the features analyzed for this experiment, which includes five context features, representing the information about the current state of the environment and the vehicle, and two action features, which describe the users' interactions with the selected comfort systems.

7.1.2 Data Pre-Processing

Before using the data to extract user behavior, the four datasets (one per user) are analyzed and sanitized to avoid misleading results. This pre-processing procedure

Feature description	Feature type
GPS latitude	Context
GPS longitude	Context
Date and time information	Context
Vehicle speed	Context
Outside temperature	Context
Window request (driver side)	Action
Seat heater intensity level request (driver seat)	Action

Table 7.1: Overview of the features recorded for the experimental evaluation of the user behavioral model. Information was recorded from the participant's vehicles using a data logger.

consists of two steps: data cleansing and data transformation, which can be applied in both offline and online settings.

7.1.2.1 Data Cleansing

A careful inspection of the data shows that some records contain undetermined or impossible data combinations (e.g. $gps\_long = 180$), especially during vehicle start-up. Such problems are very common in real data gathered through sensors, as instrumental failure or problems of linking to other systems cause anomalies and measuring errors [126]. Therefore, records with invalid data are removed from the dataset.

Data inspection also shows that datasets contain many duplicated records, which are records with the same values in all features, excluding the time information. For example, in situations where the vehicle is parked, the values of all features (except the time) would typically remain constant. To reduce the size of the datasets, the duplicated records are eliminated as well.

7.1.2.2 Data Transformation

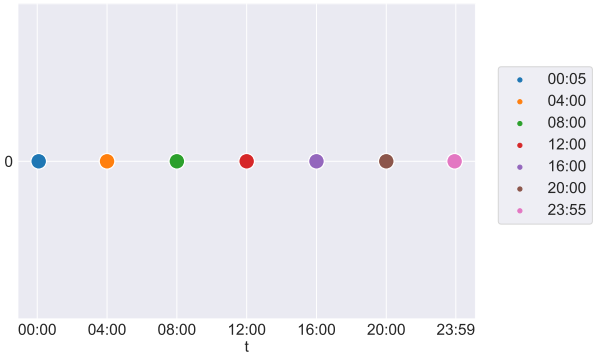
A fundamental characteristic of behavioral patterns is their temporal and periodical nature: only actions frequently repeated under similar circumstances become routines. Therefore, it is fundamental to extract the cyclical properties of the time. Following [127] and [128], the time information of the date-time feature (with format *year-month-day hour:minute:second*), is transformed into two variables by Equation 7.1. Other cyclical temporal information, such as days of the week and seasons, can be extracted from data as well. However, they are not considered in this first experiment because they are correlated with the time information, and hence the information would be redundant.

$$\begin{aligned} t_{sin} &= sin(\frac{2\pi t}{24}) \\ t_{cos} &= cos(\frac{2\pi t}{24}) \end{aligned} \tag{7.1}$$

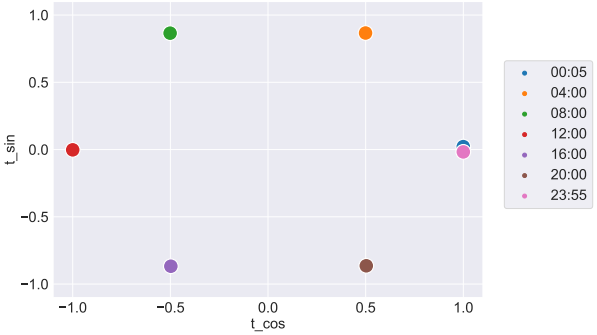
An intuitive way to explain this data transformation scheme is to plot the two-feature transformation as a 24-hour clock, as shown in Figure 7.1. Whereas the points 23:55 and 00:05 are only 10 minutes apart, the distance between the two data points is very large in the one-feature space (Figure 7.1a). However, once the time information is transformed into two features using the sine and cosine transformation, the two data points are very close (Figure 7.1b). This representation of the time information facilitates the modeling process.

7.1.3 Dataset Overview

An overview of the four datasets and the features analyzed after the pre-processing steps is provided in Table 7.2 and Table 7.3, respectively.



(a) Time one-feature space.



(b) Time two-feature space, after transformation via Equation 7.1.

Figure 7.1: A scatter plot of the sine/cosine transformation for encoding the cyclical nature of time.

User	U1	U2	U3	U4
Length (experiment duration in days)	102	132	174	189
#of request to open window completely	121	53	135	82
#of request to set seat heater at highest intensity level	149	44	83	95

Table 7.2: Overview of the four datasets used in the experimental evaluation.

Feature	Description	Type	Categories
gps_long	GPS latitude	Numerical	
gps_lat	GPS longitude	Numerical	
t_sin	Time information (sin)	Numerical	
t_cos	Time information (cos)	Numerical	
v_speed	Vehicle speed	Numerical	
o_temp	Outside temperature	Numerical	
win_drv_req	Window request (driver)	Categorical	*completely open*, *intermediate*, *completely closed*
sh_drv_lvl	Seat heater level request (driver)	Categorical	*off*, *low*, *intermediate*, *high*

Table 7.3: Features extracted from the datasets.

7.2 Analysis of the User Behavior with the Power Windows

The user behavioral model is employed to analyze the user behavior with the power windows. More specifically, the user behavioral model is used to find out the situations in which each user frequently opens the window and would probably open it in the future.

7.2.1 Model Configuration

In this example, model hyperparameters are chosen to initially favor behavioral patterns with small context regions in the location space. In other words, at the beginning of the learning process, the user behavioral model is encouraged to find out behavioral patters that involve opening the window at nearly the same place, such as at the entrance of a shopping center or a garage.

For this purpose, the value of the model hyperparameters used for the analysis of each data set is defined as listed in Table 7.4. The relatively small value of the heuristic threshold on the location feature space, ξ_{loc}, aims to favor small context regions on the latitude and longitude space, whereas the other features are not limited in any way. The selection of ξ_{loc} follows the criterion by which, on average, data points belonging to the same context region must not deviate from the mean by more than 10 meters. This criterion corresponds to setting the maximum standard deviation of the location variable, denoted as σ_{max}, to 10 meters. Because 1 meter is approximately 10^{-5} latitude/longitude degrees, this is equivalent to $\sigma_{lat,max} = \sigma_{lon,max} = 10^{-4}$ degrees. Since ξ_{loc} defines the maximum desired generalized variance of a context region (see Equation 5.2), in order to satisfy the previously mentioned constraint, the value of ξ in the location dimension is set as follows:

$$\xi_{loc} \triangleq \text{GV}_{max}(\mathcal{N}(\mu, \Sigma)) \approx (\sigma_{loc,max})^2 = (10^{-4})^2 = 10^{-8}$$

Hyperparameter	Initial value
α	1
$\boldsymbol{\mu_0}$	$\mathbf{0}$
κ	10^{-6}
$\boldsymbol{\Psi}$	$\text{diag}(10^{-9}, 10^{-9}, 10^{-4}, 10^{-4}, 10^{2}, 10^{-2})$
ν	$d+1$
β_0	1
α_0	1
ξ_{loc}	10^{-8}
w_{pos_i}	1
w_{neg_i}	2

Table 7.4: Parameters and hyperparameters of the user behavioral model to evaluate the user behavior with the power window system (feature dimension $d = 6$). A vector of d zeros is denoted by $\mathbf{0}$.

An important aspect of the user behavioral model is its ability to predict future user actions, given by the action distribution. In order to obtain behavioral patterns where users would very probably open the window next time when the same conditions are met, a higher negative indirect feedback weight (w_{neg_i}) than a positive feedback weight (w_{pos_i}) is chosen. Thus, the model is penalized more strongly when the user misses an action than rewarded when the user opens the window. Moreover, the model is rewarded every time users open the window completely, and punished when users do not open the window in situations where the model would have thought, that is, when users miss opening the window in a context region. Finally, it is assumed that instances of user behavior older than 2 months are no longer relevant for learning user behavioral patterns. Hence, the length of the time-based window is set to $W = 60$ days.

7.2.2 Results and Discussion

The data of each user is analyzed by the user behavioral model, one at a time. The behavioral patterns learned by the model are presented for its evaluation on an interactive map, as the one shown in Figure 7.2. The interactive map helps to understand the spatio-temporal character of the behavioral patterns learned by the user behavioral model from the user's past actions. The resulting behavioral patterns are represented on the interactive map at one-minute intervals, providing a dynamic view of how these patterns evolve over time. This spatio-temporal visualization is achieved through the use of a time slider, located at the bottom corner of the map, which enables to scroll through the timeline of the experiment.

On the interactive map, the geospatial information of the context regions of the learned behavioral patterns is represented taking advantage of the Gaussian nature of the context distribution. Therefore, context regions of the inferred behavioral patterns are graphically illustrated using the 3-sigma confidence ellipse of the corresponding multivariate Normal distribution $\mathcal{N}(\mu_k, \Sigma_k)$ in the latitude and longitude space. The description of the behavioral patterns concerning the other context features, such as time and temperature, cannot be shown graphically on

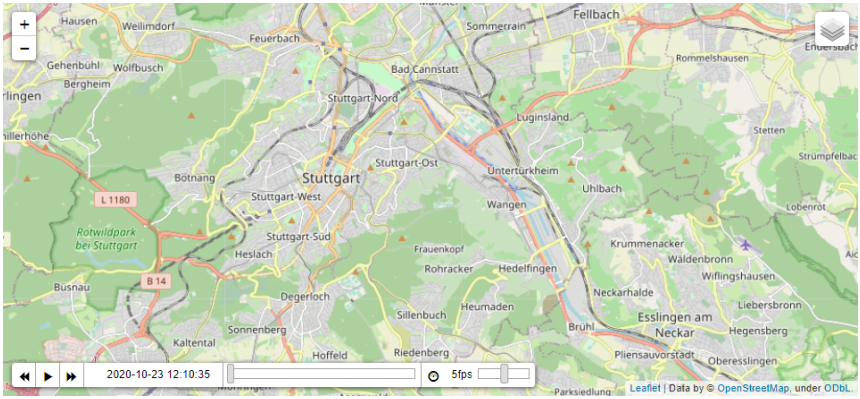


Figure 7.2: Image from an interactive map used to understand the spatio-temporal character of the behavioral patterns. The time slider in the bottom-left corner allows to select the time interactively.

the two-dimensional map. Therefore, the range in which each feature is likely to fall, based on the mean and variance, is represented in text form. For each context distribution k, the expected range of a context feature i is calculated as follows:

$$[\mu_{k,i} - 3\sigma_{k,i}, \mu_{k,i} + 3\sigma_{k,i}]$$

An example of the graphical representation of a learned behavioral pattern on the interactive map is shown in Figure 7.3.

Similarly, the action distribution of the learned behavioral patterns is presented both graphically and in text form on the interactive map. The probability of a future action is determined based on threshold values. The probability of a user action is considered to be high if the mean of the corresponding action distribution is greater than $\tau_{act} = 0.8$ and the variance is less than 0.2. For this purpose, the confidence ellipses representing the context regions are colored either in green or blue according to the action probability. Green ellipses represent behavioral patterns where the model predicts a high probability of future user action. Hence, green-colored context regions indicate the circumstances in which a self-learning

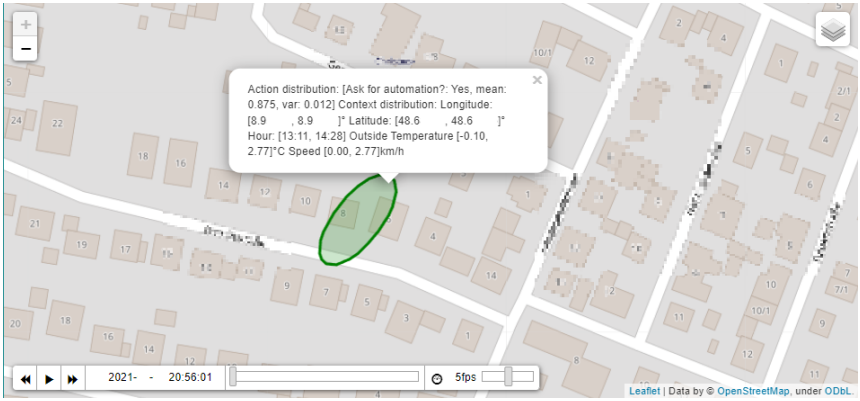


Figure 7.3: Example of a behavioral pattern inferred by the user behavioral model, represented graphically and in text form.

comfort system would automatically control the comfort system on the user's behalf. On the other hand, blue colored context regions represent behavioral patterns with a lower probability of a future user actions. Moreover, the value of the mean and variance of the action distribution of the behavioral patterns is also included in text form in the pop-up window in which the context information is presented.

The user behavioral model was trained for each dataset, corresponding a different user. The resulting behavioral patterns of each user, represented on interactive maps, were submitted for evaluation. A thorough inspection revealed that the identified behavioral patterns are not only reasonable, but also match the routines of the users.

As an illustration, the evolution of a behavioral pattern of user U1 learned by the user behavioral model is presented in Figure 7.4. Over a span of 10 days, user's U1 window-opening activities lead to the refinement and adaptation of the model, incorporating new evidence of the user's frequently performed actions. As a result, the behavioral pattern after those 10 days (bottom) aligns more precisely with the user's past actions.

(a)

(b)

Figure 7.4: Example of a behavioral pattern of user U1 with the power window system. The behavioral pattern changes with time, adapting to changes in the user behavior.

7.3 Analysis of the User Behavior with the Seat Heater

The user behavioral model is employed to analyze the user behavior with the seat heater system. In particular, the user behavioral model is employed to find out the situations in which each user frequently starts the seat heater at the highest intensity level and would perform the same action again in the future.

7.3.1 Model Configuration

The initial values of the model hyperparameters used for the analysis of the user behavior with the seat heater are shown in Table 7.5. For this analysis, a larger initial value of the heuristic threshold ξ_{loc} is selected than in the evaluation with the power window (Section 7.2), which allows context distributions to cover larger areas on the location space. This choice is based on expert consultations, which have indicated that users generally activate the seat heater based on temperature considerations, with the vehicle's location being a less significant factor. Therefore, the value of ξ_{loc} is selected to ensure that the maximum generalized variance within the data points assigned to the same context regions is not greater than 1000 meters. Given that 1000 meters are approximately 10^{-2} degrees in the latitude/longitude space, the value of ξ_{loc} is set to $\xi_{loc} \triangleq \mathrm{GV}_{max}(\mathcal{N}(\mu, \Sigma)) \approx (\sigma_{loc,max})^2 = (10^{-2})^2 = 10^{-4}$.

In this second experiment, the user behavioral model receives rewards (i.e., positive feedback) every time users select the highest seat heater intensity level. Conversely, it is punished when users do not activate the seat heater in situations where the model would have expected them to do so. When the user's current situation aligns with context region of a learned behavioral pattern, as per Equation 4.17, but the user misses activating the seat heater, it is considered as a negative indirect feedback.

Hyperparameter	Initial value
α	1
$\boldsymbol{\mu_0}$	$\mathbf{0}$
κ	10^{-6}
$\boldsymbol{\Psi}$	$\text{diag}(10^{-9}, 10^{-9}, 10^{-4}, 10^{-4}, 10^{2}, 10^{-2})$
ν	$d+1$
β_0	1
α_0	1
ξ_{loc}	10^{-4}
w_{pos_i}	1
w_{neg_i}	2

Table 7.5: Parameters and hyperparameters of the user behavioral model to evaluate the user behavior with the seat heater system (feature dimension $d = 6$). A vector of d zeros is denoted by $\mathbf{0}$.

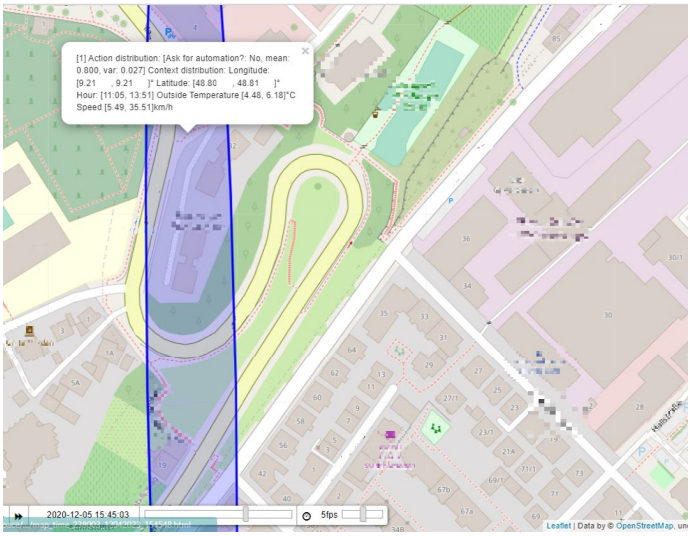
Finally, the forgetting mechanism of the user behavioral model is implemented by considering instances older than 2 months no longer relevant for learning user behavioral patterns. As a result, the length of the time-based sliding window is set to $W = 60$ days.

7.3.2 Results and Discussion

The user behavioral model is employed to analyze the behavior of each user with the seat heater system. The resulting behavioral patterns are represented on interactive maps. The behavioral patterns inferred by the user behavioral model were carefully inspected, affirming their alignment with the routines reported by the participants.

An example of a behavioral pattern of one of the participants learned by the user behavioral model is shown in Figure 7.5. As in the previous example, the evolution of a behavioral pattern becomes evident as the the user performs more actions over

time. Not only have the context circumstances describing the behavioral pattern change, but also the model becomes more certain about the user's future actions. This is evident as the color of the context region changes from blue, indicating higher uncertainty, to green, indicating lower uncertainty. This result suggests that if the user U4 was in a situation similar to the one described by the behavioral pattern in Figure 7.5b, a self-learning seat heater system would have had enough confidence to autonomously activate the seat heater on the user's behalf.



(a)

(b)

Figure 7.5: Example of a behavioral pattern of user U4 with the seat heater system. The behavioral pattern changes with time, adapting to changes in the user behavior. Also, the uncertainty decreases with time.

8 Conclusion and Outlook

8.1 Conclusion

The growing popularity of personalized services has pushed the automotive industry to rethink the interior space in vehicles to meet customer demands for more individual and intelligent comfort systems. This dissertation presents the framework to develop personalized, self-learning comfort systems, which are capable of learning the individual preferences of each user, continuously adapting to changes in the user's behavior.

Three research questions are addressed in this work:

RQ 1 *What are the requirements for a self-learning comfort system?*

RQ 2 *How can the architecture and the components of a self-learning comfort system be designed to satisfy the specified requirements?*

RQ 3 *How can individual user behavioral patterns with vehicle comfort functionalities be learned and predicted using machine learning algorithms?*

The first research question is addressed in Chapter 3.2, for which the current foundations and state of the art in intelligent systems, the vehicle environment and Bayesian learning is acknowledged.

Once the requirements are described, a generic functional architecture is further presented in Chapter 3.3, answering the second research question. As a result, four fundamental components for a self-learning comfort system are identified: the input/output handler, the decision handler, the interaction handler, and the

user behavioral model, being the last one the most essential one, as it provides the system with learning and adaptive capabilities.

For the realization of the user behavioral model and hence, to answer the third research question, a novel Bayesian nonparametric algorithm is presented in Chapter 4. The proposed user behavioral model allows the identification of the recurrent user actions and the context circumstances at which they happen, thereby facilitating the identification and prediction of user behavioral patterns. The model continuously refines its knowledge based on the user's direct and indirect feedback, and is designed to handle the uncertainty inherent in the environment an intuitive manner.

To infer the parameters of the user behavioral model, a novel truncation-free variational inference algorithm is presented. This contribution is fundamental, as it allows the model to dynamically adapt its complexity to each individual user, preventing over- and underfitting.

In Chapter 5, a preliminary prototype of a self-learning window system in a vehicle is presented. This prototype serves as an initial evaluation of the feasibility of a self-learning comfort system and the truncation-free implementation of the user behavioral model, offering valuable insights. Based on the knowledge gained with the prototype, an optimized heuristic-based inference algorithm is introduced, which enables to incorporate expert knowledge into the learning procedure by adjusting the desired level of detail of the learned contextual information.

The performance of the user behavioral model is evaluated using synthetically generated datasets in Chapter 6. The analysis presented firstly investigates the model's ability to learn the context circumstances describing behavioral patterns, followed by a comprehensive examination of how the model handles user feedback.

Finally, real-word user data is employed to assess the performance of the user behavioral model in Chapter 7. To this goal, a dedicated six-month experiment with four participants was conducted to collect user behavioral data with vehicle comfort functionalities. Each dataset is employed to train an individual user behavioral model, as a self-learning comfort system would do.

8.2 Outlook

While the overall research on self-learning vehicle systems introduced in this thesis shows very promising results, there are some challenges to be addressed in future work.

Even though the proposed nonparametric Bayesian model has very few model hyperparameters in comparison to other methods, it is still necessary to manually define them before its implementation. In future work, it should be explored how to simplify this process. The development of connected cars offers a promising opportunity to define the model hyperparameters based on the data collected from many different users, which can provide insights about common user behaviors. Hence, future work shall investigate the techniques to represent the initial assumptions about user behavior using fleet data instead of expert knowledge, while complying with the data privacy and protection norms for vehicle applications. For this purpose, differential privacy ([129]) and federated learning ([130]) techniques are promising, as they ensure that individual data points remain private while contributing to the overall learning process.

Other aspect of the user behavioral model that could be revised in the future is its forgetting mechanism, which is based on the time-based sliding window technique. Although it is simple and effective, other methods that offer more flexibility and adaptability should be investigated. Some of these techniques include exponential weighting and decay functions. Exponential weighting methods assign exponentially decreasing weights to older data points. This approach gives more weight to recent data while gradually reducing the impact of older data. Decay functions, on the other hand, utilize specific decay functions that define the rate at which the importance of older data diminishes over time.

Furthermore, the proposed model is designed to work only with vehicle functionalities with binary activation states (action vs. inaction). In future versions, the presented model can be enhanced to handle functionalities with more activation states (e.g. activation levels) by using a Categorical distribution instead of a Bernoulli distribution for the implementation of the action distribution.

To provide customers with a consistent and seamless user experience among multiple vehicles, especially within car-sharing applications, the portability and transferability of the learned user preferences and behavioral patterns is vital. Even though the presented components of a self-learning system enable it and meet the current legal and data privacy regulations, it should be validated on a new prototype experiment.

While the validation of the user behavioral model with real data marked a significant milestone, the pursuit of broader and more diverse real-world data remains a crucial step. Collecting more real-world data will allow to validate the model against a wider spectrum of user behaviors, which will contribute to a more robust understanding of the model's performance and its adaptability. Furthermore, future work shall investigate additional context features that help to represent user's behavior and indirect feedback, which can improve the precision and accuracy of the user behavioral model. Special attention should be paid to including information about user emotions (happiness, disgust, surprise) and cognitive processes (stress) into the user behavioral model, for which data about the heart rate or skin temperature have shown to be valuable [78]. Nevertheless, cross-validation techniques ([131], [132]) and exploratory analysis shall be employed to discover other relevant context features.

In the feasibility prototype for a self-learning window system presented in Section 5, the design of the user interface utilized was not fully functional, because it is not the main focus of this dissertation. Hence, the concept, development and validation of a functional user interface for a self-learning comfort system is an open research topic. User interfaces play an important role for the acceptance of interactive systems [25]. They are fundamental to build trust in the system, which is a key factor for acceptance. For this purpose, enhancing the user understanding on the learned behavioral patterns and the decisions made by the self-learning system can be beneficial to build trust and confidence on the system, as it would increase the system predictability and comprehensibility. In this context, recent advancement in the field of explainable artificial intelligence methods are promising and should be explored ([133], [134]).

Finally, the transferability of the results need to be verified. Even though, it was possible to evaluate a self-learning window system and get significant results about its feasibility, the prototype was conducted in an early stage within the development process. In future work, self-learning comfort systems should be evaluated in a context of real usage, for which a long-term user study with more participants must be conducted. A long-term user study would reveal the effectiveness and evolutional behavior of a self-learning comfort system. This leads to the question of which method can be used to measure driver acceptance and distraction.

A First Truncation-free VI Algorithm

Algorithm 2: First Truncation-free CAVI

Data: Data set $B^{u,f}_{\text{recent}} = \{(a_t, x_t)\}$, $t = \{1, 2, ..., W\}$ via (4.20)
Input: Normal-inverse-Wishart hyperparameters μ_0, κ_0, Ψ_0, ν_0,
Beta hyperparameters α_0, β_0,
DP concentration hyperparameter α,
initial number of behavioral patterns K_0,
length of the time-based sliding window W
Output: Set of K behavioral patterns assignments z_k,
each associated with a context distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and an action distribution $Ber(\rho_k)$

$K \leftarrow K_0$;
for $t = 1, 2, ...$ **do**
  if $t = 1$ *or new component added* **then**
    Initialize $q^t(Z)$ via k-means with $K = K_0$;
  end
  else
    Initialize $q^t(Z)$ using $q^{t-1}(\Theta)$;
  end

```
9
10      repeat
11          Update q^t(Θ) via (2.13);
12          Update q^t(Z) via (2.14);
13          Compute component counts N_k, for k = 1, 2, ..., K + 1;
14          if N_{K+1} > 0 then
15              Add a new component;
16              K ← K + 1;
17          end
18          if any N_k = 0 for k = 1, ..., K then
19              K ← K − |{N_k = 0, k = 1, ..., K}|;
20              Remove components k with N_k = 0;
21          end
22      until convergence via (4.19);
23  end
24  for each assignment z_n do
25      Get set of actions from B^{u,f}_recent assigned to z_n;
26      Calculate ρ_n via (4.9) ;
27  end
```

B Optimized Truncation-free VI Algorithm

Algorithm 3: Optimized Truncation-free CAVI with Heuristic Thresholding

Data: Data set $B^{u,f}_{\text{recent}} = \{(a_t, x_t)\}$, $t = \{1, 2, ..., W\}$ via (4.20)
Input: Normal-inverse-Wishart hyperparameters μ_0, κ_0, Ψ_0, ν_0
Beta hyperparameters α_0, β_0
DP concentration hyperparameter α,
initial number of behavioral patterns K_0,
length of the time-based sliding window W,
heuristic threshold ξ
Output: Set of K behavioral patterns assignments z_k,
each associated with a context distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and an action distribution $Ber(\rho_k)$

$K \leftarrow K_0$;
for $t = 1, 2, ...$ **do**
  if $t = 1$ *or new component added* **then**
    Initialize $q^t(Z)$ via k-means with $K = K_0$;
  end
  else
    Initialize $q^t(Z)$ using $q^{t-1}(\Theta)$;
  end

```
9
10      repeat
11          Update q^t(Θ) via (2.13);
12          Update q^t(Z) via (2.14);
13          Compute component counts N_k, k = 1, 2, ..., K + 1;
14          Sort N_k in descending order;
15          Perform suboptimal convergence test via (5.2);
16          if N_{K+1} > 0 or suboptimal components identified then
17              Add a new component;
18              K ← K + 1;
19          end
20          if any N_k = 0 for k = 1, ..., K then
21              Remove components k with N_k = 0;
22              K ← K − |{N_k = 0, k = 1, ..., K}|;
23          end
24      until convergence via (4.19);
25  end
26  for each assignment z_n do
27      Get set of actions from B^{u,f}_recent assigned to z_n ;
28      Calculate ρ_n via (4.9) ;
29  end
```

Data: Data set $B^{u,f}_{\text{recent}} = \{(a_t, x_t)\}$, $t = \{1, 2, ..., W\}$ via (4.20)
Input: Normal-inverse-Wishart hyperparameters μ_0, κ_0, Ψ_0, ν_0
Beta hyperparameters α_0, β_0
DP concentration hyperparameter α,
length of the time-based sliding window W,
initial number of behavioral patterns K_0,
heuristic threshold ξ
Output: Set of K behavioral patterns assignments z_k,
each associated with a context distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and an action distribution $Ber(\rho_k)$

$K \leftarrow K_0$;
for $t = 1, 2, ...$ **do**
  if $t = 1$ *or new component added* **then**
    Initialize $q^t(Z)$ via k-means with $K = K_0$;
  end
  else
    Initialize $q^t(Z)$ using $q^{t-1}(\Theta)$;
  end
  begin
    repeat
      Update $q^t(\Theta)$ via (2.13);
      Update $q^t(Z)$ via (2.14);
      Compute component counts N_k, $k = 1, 2, ..., K + 1$;
      Sort N_k in descending order;
      Perform suboptimal convergence test via
       $\text{GV}(\mathcal{N}(\mu, \Sigma)) > \xi$;
      if $N_{K+1} > 0$ *or suboptimal components identified* **then**
        Add a new component;
        $K \leftarrow K + 1$;
      end
      if *any* $N_k = 0$ *for* $k = 1, ..., K$ **then**
        Remove components k with $N_k = 0$;
        $K \leftarrow K - |\{N_k = 0,\ k = 1, ..., K\}|$;
      end
    until *convergence via* $\sum_{k=1}^{K} ||\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_k^{i-1}||^2 < \epsilon$;
  end
for *each assignment* z_n **do**
  Get set of actions from $B^{u,f}_{\text{recent}}$ assigned to z_n ;
  Calculate ρ_n via (4.9) ;

C VI for Bayesian Nonparametric Mixture Models

Given a collection of observed data $X = \{x_1, x_2, \ldots, x_N\}$, latent variables $Z = \{z_1, z_2, \ldots, z_N\}$, and model parameters $\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$, a Bayesian nonparametric (BNP) model can be defined as follows:

$$\begin{aligned} p(X, Z, \Theta) &= p(X|Z, \Theta)p(Z)p(\Theta) \\ &= [\prod_{n=1}^{N} p(x_n|z_n, \theta_n)]p(Z)[\prod_{k=1}^{\infty} p(\theta_k)] \end{aligned} \tag{C.1}$$

The posterior distribution of a BNP is defined as:

$$p(Z|X, \Theta) = \exp\{\log p(X, Z|\Theta) - \log p(X|\Theta)\} \tag{C.2}$$

where the second term, the log marginal probability of the observations, is defined as:

$$\log p(X|\Theta) = \log \int p(Z, X|\Theta)\, dZ \tag{C.3}$$

The log marginal probability of the observations is typically difficult to compute given that the latent variables become dependent when conditioning on observed data. Hence, the goal of the variational inference algorithm is to find an approximate posterior distribution $q(Z, \Theta)$ that minimizes the Kullback-Leibler (KL) divergence from the true posterior $p(Z|X, \Theta)$.

Under the mean-field variational approximation, the variational distribution $q(Z, \Theta)$ factorizes as:

$$q(Z, \Theta) = q(Z)q(\Theta) = \prod_{n=1}^{N} q(z_n) \prod_{k=1}^{\infty} q(\theta_k). \tag{C.4}$$

The updates for the variational distribution involve optimizing the parameters to minimize the KL divergence. Using the mean-field variational approximation, the terms of the ELBO can be decomposed as [101]:

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}[\log p(X, Z, \Theta) - \mathbb{E}[\log q(Z, \Theta)]] \\ &= \sum_{k=1}^{\infty} \mathbb{E}[\log p(\theta_k)] + \sum_{n=1}^{N} (\mathbb{E}[\log p(z_n)] + \mathbb{E}[\log p(x_n | z_n, \theta_{z_n})]) \\ &\quad - \sum_{n=1}^{N} \mathbb{E}[\log q(z_n)] - \sum_{k=1}^{\infty} \mathbb{E}[\log q(\theta_k)],\end{aligned} \tag{C.5}$$

where the expectations are taken with respect to $q(Z, \Theta)$.

Applying Equation 2.10, the coordinate ascent update equations for the model parameters and latent variables are defined in Equation C.6 and C.7, respectively.

$$\log q^*(\theta_k) = \log p(\theta_k) + \sum_{n=1}^{N} \mathbb{E}[\log p(x_n | \theta_k)] + \text{const} \tag{C.6}$$

$$\log q^*(z_n) = \log p(z_n | Z_{\not n}) + \mathbb{E}[\log p(x_n | \theta_{z_n})] + \text{const}, \tag{C.7}$$

The expectations are taken with respect to the variational density $q(\cdot)$ for all of the other variables, i.e., $q(Z, \Theta_{\not k})$ for C.6 and $q(Z_{\not n}, \Theta)$ for C.7.

The implementation of the updates C.6 and C.7 is simplified when the model $p(X|Z, \Theta)$ is in the conjugate exponential family. For example, if the components

in a mixture of a multivariate Normal distributions with unknown mean and unknown covariance have a Normal Wishart prior, the coordinate ascent update equations for the cluster parameters and cluster assignments can be calculated in close form.

D Probability Distributions

In this appendix, the main properties of the probabilities distributions mentioned throughout this dissertation are summarized. For each distribution, the key statistics are listed, such as the expectation $\mathbb{E}[x]$, the variance (or covariance), the mode, and the entropy $H[x]$.

Bernoulli

The Bernoulli distribution characterizes the probability distribution of a single binary variable, often denoted as $x \in \{0, 1\}$. This distribution is frequently used to model outcomes like coin flips, where x may represent "success" (1) or "failure" (0). Governed by a single parameter $\mu \in [0, 1]$, the Bernoulli distribution quantifies the probability of achieving a "success" ($x = 1$).

$$\text{Ber}(x|\mu) = \mu^x(1-\mu)^{1-x} \tag{D.1}$$

$$\mathbb{E}[x] = \mu \tag{D.2}$$

$$\text{var}[x] = \mu(1-\mu) \tag{D.3}$$

$$\text{mode}[x] = \begin{cases} 1 \text{ if } \mu \geq 0.5, \\ 0 \text{ otherwise} \end{cases} \tag{D.4}$$

$$\text{H}[x] = -\mu \ln \mu - (1-\mu)\ln(1-\mu) \tag{D.5}$$

The Bernoulli is a special case of the binomial distribution for the case of a single observation. Its conjugate prior for μ is the beta distribution.

Beta

The Beta distribution is defined over a continuous variable $\mu \in [0, 1]$ and is commonly utilized to represent the probability for some binary event. It is governed by two parameters a and b that are constrained by $a > 0$ and $b > 0$ to ensure that the distribution can be normalized.

$$\text{Beta}(\mu|a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1 - \mu)^{b-1} \tag{D.6}$$

$$\mathbb{E}[\mu] = \frac{a}{a + b} \tag{D.7}$$

$$\text{var}[\mu] = \frac{ab}{(a + b)^2(a + b + 1)} \tag{D.8}$$

$$\text{mode}[\mu] = \frac{a - 1}{a + b - 2} \tag{D.9}$$

The beta is the conjugate prior for the Bernoulli distribution, for which a and b can be interpreted as the effective prior number of observations of $x = 1$ and $x = 0$, respectively. Its density is finite if $a \geq 1$ and $b \geq 1$, otherwise there is a singularity at $\mu = 0$ and/or $\mu = 1$. For $a = b = 1$, it reduces to a uniform distribution. The beta distribution is a special case of the K-state Dirichlet distribution for $K = 2$.

Binomial

The binomial distribution gives the probability of observing m occurrences of $x = 1$ in a set of N samples from a Bernoulli distribution, where the probability of observing $x = 1$ is $\mu \in [0, 1]$.

$$\text{Bin}(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m} \tag{D.10}$$

$$\mathbb{E}[m] = N\mu \tag{D.11}$$

$$\text{var}[m] = N\mu(1-\mu) \tag{D.12}$$

$$\text{mode}[m] = \lfloor (N+1)\mu \rfloor \tag{D.13}$$

where $\lfloor (N+1)\mu \rfloor$ denotes the largest integer that is less than or equal to $(N+1)\mu$, and the quantity

$$\binom{N}{m} = \frac{N!}{m!(N-m)!} \tag{D.14}$$

denotes the number of ways of choosing m objects out of a total of N identical objects. The particular case of $N = 1$ is known as the Bernoulli distribution, and for large N the binomial distribution is approximately Gaussian. The conjugate prior for μ is the beta distribution.

Dirichlet

The Dirichlet is a multivariate distribution over K random variables $0 \leq \mu_k \leq 1$, where $k = 1, ..., K$, subject to constraints

$$0 \leq \mu_k \leq 1, \sum_{k=1}^{K} \mu_k = 1. \tag{D.15}$$

Denoting $\boldsymbol{\mu} = (\mu_1, ..., \mu_K)^{\mathrm{T}}$ and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K)^{\mathrm{T}}$, the key statistics are:

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = C(\boldsymbol{\alpha}) \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} \tag{D.16}$$

$$\mathbb{E}[\mu_k] = \frac{\alpha_k}{\hat{\alpha}} \tag{D.17}$$

$$\mathrm{var}[\mu_k] = \frac{\alpha_k(\hat{\alpha} - \alpha_k)}{\hat{\alpha}^2(\hat{\alpha} + 1)} \tag{D.18}$$

$$\mathrm{cov}[\mu_j \mu_k] = -\frac{\alpha_j \alpha_k}{\hat{\alpha}^2(\hat{\alpha} + 1)} \tag{D.19}$$

$$\mathrm{mode}[\mu_k] = -\frac{\alpha_k - 1}{\hat{\alpha} - K} \tag{D.20}$$

$$\mathbb{E}[\ln \mu_k] = \psi(\alpha_k) - \psi(\hat{\alpha}) \tag{D.21}$$

where

$$C(\boldsymbol{\alpha}) = \frac{\Gamma(\hat{\alpha})}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \tag{D.22}$$

and

$$\hat{\alpha} = \sum_{k=1}^{K} \alpha_k. \tag{D.23}$$

D.24 is known as the digamma function. The parameters α_k are subject to the constraint $\alpha_k > 0$ in order to ensure that the distribution can be normalized.

$$\psi(a) \equiv \frac{d}{da} \ln \Gamma(a) \tag{D.24}$$

The Dirichlet forms the conjugate prior for the multinomial distribution and represents a generalization of the beta distribution. In this case, the parameters α_k can be interpreted as effective numbers of observations of the corresponding values of the K-dimensional binary observation vector $\boldsymbol{x}$. As with the beta distribution, the Dirichlet has finite density everywhere provided $\alpha_k \geq 1$ for all k.

Gamma

The Gamma is a probability distribution over a positive random variable $\tau > 0$ governed by parameters a and b that are subject to the constraints $a > 0$ and $b > 0$ to ensure that the distribution can be normalized.

$$\text{Gam}(\tau|a,b) = \frac{1}{\Gamma(a)} b^a \tau^{a-1} e^{-b\tau} \tag{D.25}$$

$$\mathbb{E}[\tau] = \frac{a}{b} \tag{D.26}$$

$$\text{var}[\tau] = \frac{a}{b^2} \tag{D.27}$$

$$\text{mode}[\tau] = \frac{a-1}{b} \text{ for } a \geq 1 \tag{D.28}$$

The gamma distribution is the conjugate prior for the precision (inverse variance) of a univariate Gaussian. For $a \geq 1$ the density is everywhere finite, and the special case of $a = 1$ is known as the exponential distribution.

Gaussian

The Gaussian is the most widely used distribution for continuous variables. It is also known as the normal distribution. In the case of a single variable $x \in (-\infty, \infty)$ it is governed by two parameters, the mean $\mu \in (-\infty, \infty)$ and the variance $\sigma^2 > 0$.

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{D.29}$$

$$\mathbb{E}[x] = \mu \tag{D.30}$$

$$\text{var}[x] = \sigma^2 \tag{D.31}$$

$$\text{mode}[x] = \mu \tag{D.32}$$

$$H[x] = \frac{1}{2}\left(1 + \ln(2\pi\sigma^2)\right) \tag{D.33}$$

The inverse of the variance $\tau = 1/\sigma^2$ is called the precision, and the square root of the variance σ is called the standard deviation. The conjugate prior for μ is the Gaussian, and the conjugate prior for τ is the gamma distribution. If both μ and τ are unknown, their joint conjugate prior is the Gaussian-gamma distribution.

For a D-dimensional vector $\boldsymbol{x}$, the Gaussian is governed by a D-dimensional mean vector $\boldsymbol{\mu}$ and a $D \times D$ covariance matrix $\boldsymbol{\Sigma}$ that must be symmetric and positive definite.

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{D.34}$$

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \tag{D.35}$$

$$\text{cov}[\mathbf{x}] = \boldsymbol{\Sigma} \tag{D.36}$$

$$\text{mode}[\mathbf{x}] = \boldsymbol{\mu} \tag{D.37}$$

$$H[\mathbf{x}] = \frac{1}{2}\left(D\ln(2\pi e) + \ln|\boldsymbol{\Sigma}|\right) \tag{D.38}$$

The inverse of the covariance matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix, which is also symmetric and positive definite. Averages of random variables tend to a Gaussian, by the central limit theorem, and the sum of two Gaussian variables is

again Gaussian. The conjugate prior for $\boldsymbol{\mu}$ is the Gaussian, the conjugate prior for $\boldsymbol{\Lambda}$ is the Wishart, and the conjugate prior for $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ is the Gaussian-Wishart.

Gaussian-Wishart

The Gaussian-Wishart is the conjugate prior distribution for a multivariate Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda})$ in which both the mean $\boldsymbol{\mu}$ and the precision $\boldsymbol{\Lambda}$ are unknown. It is also called the normal-Wishart distribution. It comprises the product of a Gaussian distribution for $\boldsymbol{\mu}$, whose precision is proportional to $\boldsymbol{\Lambda}$, and a Wishart distribution over $\boldsymbol{\Lambda}$.

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{\mu}_0, \beta, \boldsymbol{W}, \nu) = \mathcal{N}\left(\boldsymbol{\mu}|\boldsymbol{\mu}_0, (\beta\Lambda)^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}|\boldsymbol{W}, \nu) \tag{D.39}$$

Multinomial

The multinomial distribution is a multivariate generalization of the binomial and gives distribution over counts m_k for a K-state discrete variable to be in state k given a total number of observations N.

$$\text{Mult}(m_1, m_2, \ldots, m_K|\boldsymbol{\mu}, N) = \binom{N}{m_1 m_2 \ldots m_K} \prod_{k=1}^{K} \mu_k^{m_k} \tag{D.40}$$

$$\mathbb{E}[m_k] = N\mu_k \tag{D.41}$$

$$\text{var}[m_k] = N\mu_k(1 - m_k) \tag{D.42}$$

$$\text{cov}[m_j m_k] = -N\mu_j\mu_k \,,\; j \neq k \tag{D.43}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^{\text{T}}$ and the quantity

$$\binom{N}{m_1 m_2 \dots m_K} = \frac{N!}{m_1! m_2! \dots m_K!} \tag{D.44}$$

gives the number of ways of taking N identical objects and assigning m_k of them to bin k for $k = 1, \dots, K$. The value of μ_k gives the probability of the random variable taking state k, and so these parameters are subject to the constraints $0 \leq \mu_k \leq 1$ and $\sum_k \mu_k = 1$. The conjugate prior distribution for the parameters μ_k is the Dirichlet.

Uniform

The uniform distribution is a probability distribution for a continuous variable x defined over a finite interval $x \in [a, b]$ where $b > a$.

$$\mathrm{U}(x|a, b) = \frac{1}{b-a}, \quad a \leq x \leq b \tag{D.45}$$

$$\mathbb{E}[x] = \frac{a+b}{2} \tag{D.46}$$

$$\mathrm{var}[x] = \frac{(b-a)^2}{12} \tag{D.47}$$

$$H[x] = \ln(b-a) \tag{D.48}$$

Wishart

The Wishart distribution is the conjugate prior for the precision matrix of a multivariate Gaussian.

$$\text{Wish}(\boldsymbol{\Lambda}|\boldsymbol{W},\nu) = \frac{|\boldsymbol{\Lambda}|^{\frac{\nu-D-1}{2}} \exp\left(-\frac{1}{2}\text{Tr}(\boldsymbol{W}^{-1}\boldsymbol{\Lambda})\right)}{2^{\frac{\nu D}{2}}|\boldsymbol{W}|^{\frac{\nu}{2}}\Gamma_D\left(\frac{\nu}{2}\right)} \tag{D.49}$$

$$\mathbb{E}[\boldsymbol{\Lambda}] = \nu\boldsymbol{W} \tag{D.50}$$

where $\boldsymbol{W}$ is a $D \times D$ symmetric, positive definite matrix, and $\psi(\cdot)$ is the digamma function defined by D.24. The parameter ν is called the number of degrees of freedom of the distribution and is restricted to $\nu > D - 1$ to ensure that the Gamma function in the normalization factor is well-defined. In one dimension, the Wishart reduces to the gamma distribution $Gam(\lambda|a, b)$ given by given by D.25 with parameters $a = \nu/2$ and $b = 1/2W$.

List of Abbreviations

AAM Automation Acceptance Model

BNP Bayesian nonparametric

CAN Controler Area Network bus

CAVI Coordinate Ascent Variational Inference

CRP Chinese Restaurant Process

DP Dirichlet Process

ELBO Evidence Lower Bound

I/O Input/Output

ISO International Organization for Standardization

IW Inverse Wishart distribution

KL Kullback-Leibler divergence

MCMC Markov Chain Monte Carlo

MVN Multivariate Normal distribution

NIV Normal inverse Wishart distribution

NMI Normalized Mutual Information

REQ Requirement

RQ Research Question

TAM Technology Acceptance Model

UAS User Assistance System

UI User interface

VI Variational Inference

List of Algorithms

List of Figures

List of Tables

Publications

[1] M. Guinea, I. Litton, R. Smiroldo, I. Nitsche, and E. Sax, "A proactive context-aware recommender system for in-vehicle use," *International Conference on Vision, Image and Signal Processing*, 2020.

[2] M. Guinea, M. Stang, I. Nitsche, and E. Sax, *Acceptance of Smart Automated Comfort Functionalities in Vehicles*, pp. 331–338. Advances in Intelligent Systems and Computing, Springer International Publishing, 2021.

[3] M. Stang, M. Guinea Marquez, and E. Sax, *CAGEN - Context-Action Generation for Testing Self-learning Functions*, pp. 12–19. Advances in Intelligent Systems and Computing, Springer International Publishing, 2021.

Bibliography

[1] O. Gusikhin, N. Rychtyckyj, and D. Filev, "Intelligent systems in the automotive industry: applications and trends," *Knowledge and Information Systems*, vol. 12, no. 2, pp. 147–168, 2007.

[2] H. Budde-Meiwes, J. Drillkens, B. Lunz, J. Muennix, S. Rothgang, J. Kowal, and D. U. Sauer, "A review of current automotive battery technology and future prospects," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 227, no. 5, pp. 761–776, 2013.

[3] K. Perera and D. Dias, "An intelligent driver guidance tool using location based services," *Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*, pp. 246–251, 2011.

[4] X. Amatriain and J. Basilico, "Recommender systems in industry: A netflix case study," vol. 45, pp. 385–419, 2015.

[5] B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, and D. Epema, "Understanding user behavior in spotify," *2013 Proceedings IEEE INFOCOM*, pp. 220–224, 2013.

[6] Accenture, "Mobility services: In car personalization," *Accenture*, 17.9.2019.

[7] IBM, "Automotive 2025: industry without borders.," 2014.

[8] P. K. Murali, M. Kaboli, and R. Dahiya, "Intelligent in-vehicle interaction technologies," *Advanced Intelligent Systems*, vol. 4, no. 2, p. 2100122, 2022.

[9] K. D. Ersche, T.-V. Lim, L. H. E. Ward, T. W. Robbins, and J. Stochl, "Creature of habit: A self-report measure of habitual routines and automatic tendencies in everyday life," *Personality and individual differences*, vol. 116, pp. 73–85, 2017.

[10] C. M. Bishop, *Pattern recognition and machine learning*. Information science and statistics, New York: Springer, 2006.

[11] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: A modern approach*. Prentice Hall series in artificial intelligence, Upper Saddle River: Prentice Hall, 3rd ed. ed., 2010.

[12] Y. Tsypkin, "Self-learning–what is it?," *IEEE Transactions on Automatic Control*, vol. 13, no. 6, pp. 608–612, 1968.

[13] P. Maes, "Agents that reduce work and information overload," *Readings in human–computer interaction*, pp. 811–821, 1995.

[14] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K.-R. Müller, "A self-learning system for detection of anomalous sip messages," *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks. IPTComm 2008.*, vol. 5310, pp. 90–106, 2008.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, 2019.

[16] V. H. Bhide and S. Wagh, "i-learning iot: An intelligent self learning system for home automation using iot," *2015 International Conference on Communications and Signal Processing (ICCSP)*, pp. 1763–1767, 2015.

[17] T. Huang and D. Liu, "A self-learning scheme for residential energy system control and management," *Neural Computing and Applications*, vol. 22, no. 2, pp. 259–269, 2013.

[18] M. Esrafilian-Najafabadi and F. Haghighat, "Towards self-learning control of hvac systems with the consideration of dynamic occupancy patterns: Application of model-free deep reinforcement learning," *Building and Environment*, vol. 226, p. 109747, 2022.

[19] J. Zhang, P. D. Roberts, and J. E. Ellis, "A self-learning fault-diagnosis system," *Transactions of the Institute of Measurement and Control*, vol. 13, no. 1, pp. 29–35, 1991.

[20] M. R. Endsley and E. O. Kiris, "The out-of-the-loop performance problem and level of control in automation," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, no. 2, pp. 381–394, 1995.

[21] M. R. Endsley, "Automation and situation awareness: Automation and human performance: Theory and applications," pp. 163–181, 2018.

[22] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, no. 1, pp. 32–64, 1995.

[23] N. Yorke-Smith, S. Saadati, K. Myers, and D. Morley, "Like an intuitive and courteous butler: a proactive personal agent for task management," *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pp. 337–344, 2009.

[24] Myers, Karen, and N. Yorke-Smith, *Proactive behavior of a personal assistive agent*. 2008.

[25] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, p. 319, 1989.

[26] H. Rafique, A. O. Almagrabi, A. Shamim, F. Anwar, and A. K. Bashir, "Investigating the acceptance of mobile library applications with an extended technology acceptance model (tam)," *Computers & Education*, vol. 145, no. 2, p. 103732, 2020.

[27] K. Wu, Y. Zhao, Q. Zhu, X. Tan, and H. Zheng, "A meta-analysis of the impact of trust on technology acceptance model: Investigation of moderating influence of subject and context type," *International Journal of Information Management*, vol. 31, no. 6, pp. 572–581, 2011.

[28] M. Ghazizadeh, J. D. Lee, and L. Ng Boyle, "Extending the technology acceptance model to assess automation," *Cognition, Technology & Work*, vol. 14, no. 1, pp. 39–49, 2012.

[29] R. J. Holden and B.-T. Karsh, "The technology acceptance model: its past and its future in health care," *Journal of biomedical informatics*, vol. 43, no. 1, pp. 159–172, 2010.

[30] Michael A. Nees, "Acceptance of self-driving cars: An examination of idealized versus realistic portrayals with a self- driving car acceptance scale," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 60, no. 1, pp. 1449–1453, 2016.

[31] R. Bader, *Proactive Recommender Systems in Automotive Scenarios*. PhD thesis, 2013.

[32] M. Guinea, M. Stang, I. Nitsche, and E. Sax, "Acceptance of smart automated comfort functionalities in vehicles," *Human Interaction, Emerging Technologies and Future Applications IV – Proceedings of the 4th International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIET – AI 2021)*, 2021.

[33] G. Fischer, "User modeling in human-computer interaction," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1/2, pp. 65–86, 2001.

[34] E. Frias-Martinez, S. Y. Chen, and X. Liu, "Survey of data mining approaches to user modeling for adaptive hypermedia," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 6, pp. 734–749, 2006.

[35] I. Zukerman and D. W. Albrecht, "Predictive statistical models for user modeling," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1/2, pp. 5–18, 2001.

[36] G. I. Webb, M. J. Pazzani, and D. Billsus, "Machine learning for user modeling," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1/2, pp. 19–29, 2001.

[37] Y. Gal, *Uncertainty in Deep Learning*. Doctor of philosophy, Department of Engineering University of Cambridge, 2016.

[38] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[39] Z. Ghahramani, "Bayesian non-parametrics and the probabilistic approach to modelling," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 371, no. 1984, p. 20110553, 2013.

[40] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," pp. 1050–1059, 2016.

[41] D. J. C. MacKay, "A practical bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[42] S. S. Rao and L. Berke, "Analysis of uncertain structural systems using interval analysis," *AIAA Journal*, vol. 35, no. 4, pp. 727–735, 1997.

[43] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[44] T. Denoeux and M.-H. Masson, "Evclus: evidential clustering of proximity data," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 34, no. 1, pp. 95–109, 2004.

[45] T. Denoeux, "A k-nearest neighbor classification rule based on dempster-shafer theory," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 5, pp. 804–813, 1995.

[46] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[47] A. M. Khan, "Cognitive connected vehicle information system design requirement for safety: Role of bayesian artificial intelligence," *Journal on Systemics, Cybernetics and Informatics*, vol. 11, pp. 54–59, 2013.

[48] G. J. Dimitrakopoulos and I. E. Panagiotopoulos, "In-vehicle infotainment systems: Using bayesian networks to model cognitive selection of music genres," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6900–6909, 2021.

[49] A. M. Khan, "Bayesian-monte carlo model for collision avoidance system design of cognitive connected vehicle," *International Journal of Intelligent Transportation Systems Research*, vol. 11, no. 1, pp. 23–33, 2013.

[50] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and Ubiquitous Computing* (H.-W. Gellersen, ed.), (Berlin, Heidelberg), pp. 304–307, Springer Berlin Heidelberg, 1999.

[51] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human–Computer Interaction*, vol. 16, no. 2-4, pp. 97–166, 2001.

[52] W. Woerndl and G. Groh, "Utilizing physical and social context to improve recommender systems," pp. 123–128, 2007.

[53] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning*, vol. 3, pp. 1157–1182, 2003.

[54] K. Baumann, "Cross-validation as the objective function for variable-selection techniques," *TrAC Trends in Analytical Chemistry*, vol. 22, no. 6, pp. 395–406, 2003.

[55] C. Shao, K. Paynabar, T. H. Kim, J. Jin, S. J. Hu, J. P. Spicer, H. Wang, and J. A. Abell, "Feature selection for manufacturing process monitoring using cross-validation," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 550–555, 2013.

[56] S. O. Hansson, "Decision theory: A brief introduction," pp. 1–, 2005.

[57] P. C. Fishburn, "Utility theory," *Management Science*, vol. 14, no. 5, pp. 335–378, 1968.

[58] E. Horvitz, "Principles of mixed-initiative user interfaces," in *CHI 99* (M. G. Williams and S. Pemberton, eds.), Conference on human factors in computing systems, (New York), pp. 159–166, ACM Press and Harlow : Pearson Education, 1999.

[59] G. Mom, *Evolution of automotive technology: A handbook*. [Place of publication not identified]: SAE International, 2015.

[60] J. P. Trovao, "An overview of automotive electronics [automotive electronics]," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 130–137, 2019.

[61] P. Coopers, "Automotive electronics cost as a percentage of total car cost worldwide from 1950 to 2030.," 2016.

[62] Mercedes-Benz, "Das interieurdesign der neuen mercedes-maybach s-klasse," 2020.

[63] Council of the European Union, "Council regulation (eu) 2016/269," 2016.

[64] Council of the European Union, "Council regulation (eu) 2002/58/ec," 2002.

[65] ISO 26262, "Road vehicles – functional safety," 2011.

[66] ISO 21434, "Road vehicles – cybersecurity engineering," 2021.

[67] ISO 9241, "Ergonomics of human-system interaction," 2019.

[68] ISO 21448, "Road vehicles — safety of the intended functionality," 2022.

[69] D. Kinalzyk, "Sotif process and methods in combination with functional safety," vol. 1442, pp. 612–623, 2021.

[70] J. R. Treat, "A study of precrash factors involved in traffic accidents," *undefined*, 1980.

[71] Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, "Driver inattention monitoring system for intelligent vehicles: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 596–614, 2011.

[72] K. Kircher, "Driver distraction: a review of the literature," *Statens väg- och transportforskningsinstitut*, p. 58, 2007.

[73] L. Tijerina, E. Parmer, and M. J. Goodman, "Driver workload assessment of route guidance system destination entry while driving: A test track study," *Proceedings of the 5th ITS World Congress*, vol. 347, no. 202, pp. 1–8, 1998.

[74] T. A. Ranney, "Driver distraction: A review of the current state-of-knowledge," *Digital Public Library of America*, 2008.

[75] R. Oppermann, "Adaptive user support: ergonomic design of manually and automatically adaptable software," 1994.

[76] S. Rogers, C.-N. Fiechter, and C. Thompson, "Adaptive user interfaces for automotive environments," *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, pp. 662–667, 2000.

[77] W. Piechulla, C. Mayser, H. Gehrke, and W. König, "Reducing drivers' mental workload by means of an adaptive man–machine interface," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 6, no. 4, pp. 233–248, 2003.

[78] F. Nasoz, C. L. Lisetti, and A. V. Vasilakos, "Affectively intelligent and adaptive car interfaces," *Information Sciences*, vol. 180, no. 20, pp. 3817–3836, 2010.

[79] R. Bader, W. Woerndl, and V. Prinz, "Situation awareness for proactive in-car recommendations of points-of-interest (poi)," *Proceedings of Workshop Context Aware Intelligent*, vol. 33rd Annual German Conference on Artificial Intelligence (KI 2010), pp. 30, 31, 2010.

[80] R. Bader, W. Woerndl, A. Karitnig, and G. Leitner, "Designing an explanation interface for proactive recommendations in automotive scenarios," vol. 7138, pp. 92–104, 2011.

[81] R. Bader, O. Siegmund, and W. Woerndl, "A study on user acceptance of proactive in-vehicle recommender systems," in *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '11, (New York, NY, USA), pp. 47–54, Association for Computing Machinery, 2011.

[82] S. R. Garzon, "Intelligent in-car-infotainment systems: A contextual personalized approach," in *2012 Eighth International Conference on Intelligent Environments*, pp. 315–318, IEEE, 26.06.2012 - 29.06.2012.

[83] T. Lavie and J. Meyer, "Benefits and costs of adaptive user interfaces," *International Journal of Human-Computer Studies*, vol. 68, no. 8, pp. 508–524, 2010.

[84] J. Ingi Árnason, J. Jepsen, A. Koudal, M. Rosendahl Schmidt, and S. Serafin, "Volvo intelligent news: A context aware multi modal proactive recommender system for in-vehicle use," *Pervasive and Mobile Computing*, vol. 14, pp. 95–111, 2014.

[85] D.-C. Kim, J. Gao, X. Wang, and C.-R. Yang, "A framework for personalized medicine with reverse phase protein array and drug sensitivity," pp. 426–429, 2011.

[86] G. Alterovitz, C. Tuthill, I. Rios, K. Modelska, and S. Sonis, "Personalized medicine for mucositis: Bayesian networks identify unique gene clusters which predict the response to gamma-d-glutamyl-l-tryptophan (scv-07) for the attenuation of chemoradiation-induced oral mucositis," *Oral oncology*, vol. 47, no. 10, pp. 951–955, 2011.

[87] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-based recommendation system using bayesian user's preference model in mobile devices," vol. 4611, pp. 1130–1139, 2007.

[88] Y. Zhang and J. Koren, "Efficient bayesian hierarchical user modeling for recommendation system," pp. 47–54, 2007.

[89] S. Niekum, "A brief introduction to bayesian nonparametric methods for clustering and time series analysis," *Carnegie Mellon University*, 2015.

[90] P. Orbanz and Y. W. Teh, "Bayesian nonparametric models," in *Encyclopedia of Machine Learning*, pp. 81–89, Springer, Boston, MA, 2011.

[91] C. Lu and X. Tang, "Surpassing human-level face verification performance on lfw with gaussianface," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[92] C. E. Rasmussen, B. J. de La Cruz, Z. Ghahramani, and D. L. Wild, "Modeling and visualizing uncertainty in gene expression clusters using dirichlet process mixtures," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 6, no. 4, pp. 615–628, 2009.

[93] M. Medvedovic and S. Sivaganesan, "Bayesian infinite mixture model based clustering of gene expression profiles," *Bioinformatics (Oxford, England)*, vol. 18, no. 9, pp. 1194–1206, 2002.

[94] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[95] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[96] T. S. Ferguson, "A bayesian analysis of some nonparametric problems," *Annals of Statistics*, vol. 1, no. 2, pp. 209–230, 1973.

[97] N. L. Hjort, *Bayesian nonparametrics*, vol. 28 of *Cambridge series in statistical and probabilistic mathematics*. Cambridge: Cambridge University Press, 2009.

[98] Y. W. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning*, pp. 280–287, Springer, Boston, MA, 2011.

[99] E. B. Sudderth, *Graphical models for visual object recognition and tracking: Graphical models for visual object recognition and tracking*. PhD thesis, Massachusetts Institute of Technology, 2006.

[100] R. M. Neal, "Markov chain sampling methods for dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 249–265, 2000.

[101] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[102] S. Sun, "A review of deterministic approximate inference techniques for bayesian machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, 2013.

[103] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016.

[104] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for agile methodology," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 864–869, 2017.

[105] C. Ebert and L. Hochstein, "Devops in practice," *IEEE Software*, vol. 40, no. 1, pp. 29–36, 2023.

[106] K. R. May, B. E. Noah, and B. N. Walker, "Driving acceptance: Appliying structural equation modeling to in-vehicle automation acceptance," in *Adjunct proceedings, AutomotiveUI 2017* (A. Löcken, S. Boll, I. Politis, S. Osswald, R. Schroeter, D. Large, M. Baumann, I. Alvarez, L. Chuang, S. Feuerstack,

M. Jeon, H. H. van Huysduynen, and N. Broy, eds.), (New York, New York), pp. 190–194, Association for Computing Machinery, 2017.

[107] R. Knote, A. Janson, L. Eigenbrod, and M. Söllner, "The what and how of smart personal assistants: Principles and application domains for is research," 2018.

[108] K. Kompass, W. Huber, and T. Helmer, "Safety and comfort systems: Introduction and overview," in *Handbook of intelligent vehicles 2012*, vol. 13, pp. 605–612.

[109] J. Pearl and D. Mackenzie, *The book of why: The new science of cause and effect*. New York: Basic Books, first edition ed., 2018.

[110] D. Görür and C. E. Rasmussen, "Dirichlet process gaussian mixture models: Choice of the base distribution," *Journal of Computer Science and Technology*, vol. 25, no. 4, pp. 653–664, 2010.

[111] M. C. Hughes and E. Sudderth, "Memoized online variational inference for dirichlet process mixture models," *Advances in Neural Information Processing Systems*, vol. 26, pp. 1133–1141, 2013.

[112] V. Huynh, D. Phung, and S. Venkatesh, "Streaming variational inference for dirichlet process mixtures," *Asian Conference on Machine Learning*, pp. 237–252, 2016.

[113] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[114] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[115] J. Lücke and D. Forster, "k-means as a variational em approximation of gaussian mixture models," *Pattern Recognition Letters*, vol. 125, no. 1, pp. 349–356, 2019.

[116] A. Zubaroğlu and V. Atalay, "Data stream clustering: a review," *Artificial Intelligence Review*, 2020.

[117] T. M. Ghanem, M. A. Hammad, M. F. Mokbel, W. G. Aref, and A. K. Elmagarmid, "Incremental evaluation of sliding-window queries over data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 57–72, 2007.

[118] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 48, p. 1, 2018.

[119] C. W. Elverum, T. Welo, and S. Tronvoll, "Prototyping in new product development: Strategy considerations," *Procedia CIRP*, vol. 50, pp. 117–122, 2016.

[120] Y. P. Raykov, A. Boukouvalas, F. Baig, and M. A. Little, "What to do when k-means clustering fails: A simple yet principled alternative algorithm," *PloS one*, vol. 11, no. 9, p. e0162259, 2016.

[121] S. Kocherlakota and K. Kocherlakota, "Generalized variance," 2004.

[122] T. W. Anderson, *An introduction to multivariate statistical analysis*. Wiley series in probability and statistics, Hoboken, N.J. and Great Britain: Wiley-Interscience, 3rd ed. ed., 2003.

[123] A. Declercq and Justus H. Piater, "Online learning of gaussian mixture models: A two-level approach," pp. 605–611, 2008.

[124] Dahua Lin, "Online learning of nonparametric mixture models via sequential variational approximation," pp. 395–403, 2013.

[125] K. P. Murphy, *Machine learning: A probabilistic perspective / Kevin P. Murphy*. Adaptive computation and machine learning series, Cambridge, Mass. and London: MIT Press, 2012.

[126] M. J. Zaki and W. Meira, *Data mining and machine learning: Fundamental concepts and algorithms / Mohammed J. Zaki, Wagner Meira, Jr.* Cambridge: Cambridge University Press, second edition ed., 2020.

[127] A. van Wyk, "Encoding cyclical features for deep learning," *Kaggle*, 2018.

[128] X. Zhu, J. Guo, S. Li, and T. Hao, "Facing cold-start: A live tv recommender system based on neural networks," *IEEE Access*, vol. 8, pp. 131286–131298, 2020.

[129] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation—TAMC*, vol. 4978 of *Lecture Notes in Computer Science*, pp. 1–19, Springer Verlag, 2008.

[130] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *(Foundations and trends \textregistered in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[131] C. Qi, J. Diao, and L. Qiu, "On estimating model in feature selection with cross-validation," *IEEE Access*, vol. 7, pp. 33454–33463, 2019.

[132] C. Shao, K. Paynabar, T. H. Kim, J. Jin, S. J. Hu, J. P. Spicer, H. Wang, and J. A. Abell, "Feature selection for manufacturing process monitoring using cross-validation," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 550–555, 2013.

[133] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, "Xai-explainable artificial intelligence," *Science robotics*, vol. 4, no. 37, 2019.

[134] F. K. Dosilovic, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," *41st International convention on information (MIPRO)*, pp. 0210–0215, 2018.