Copyright

by

Elad Liebman

2019

The Dissertation Committee for Elad Liebman
certifies that this is the approved version of the following dissertation:

Sequential Decision Making in Artificial Musical Intelligence

Committee:

Peter Stone, Supervisor

Kristen Grauman

Scott Niekum

Maytal Saar-Tsechansky

Roger B. Dannenberg

Sequential Decision Making in Artificial Musical Intelligence

by

Elad Liebman

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2019

To my parents, Zipi and Itzhak, who made me the person I am today (for better or for worse), and to my children, Noam and Omer – you are the reason I get up in the morning (literally)

Acknowledgments

This thesis would not have been possible without the ongoing and unwavering support of my advisor Peter Stone, whose guidance, wisdom and kindness I am forever indebted to. Peter is the kind of advisor who would give you the freedom to work on almost anything, but would be as committed as you are to your research once you've found something you're passionate about. Patient and generous with his time and advice, gracious and respectful in times of disagreement, Peter has the uncanny gift of knowing to trust his students' instincts, but still continually ask the right questions. I could not have hoped for a better advisor. I also wish to thank my other dissertation committee members for their sage advice and useful comments: Roger Dannenberg, Kristen Grauman, Scott Niekum and Maytal Saar-Tsechansky. I must also give thanks to my colleague and friend Corey White, who not only collaborated with me extensively, but also served as my guide into the fields of cognitive modeling and the psychology of decision-making.

Peter's academic patronage comes with the gigantic bonus of being part of the best research lab in the world (as far as I am concerned, at least), the UT Austin Learning Agents Research Group (LARG). My years at The University of Texas at Austin would have been a much poorer experience, academically and personally, without the wisdom, kindness, intelligence and companionship of my fellow labmates throughout the years: Michael Albert, Stefano Albrecht, Shani Alkoby, Samuel Barrett, Ishan Durugkar, Katie Genter, Josiah Hanna, Justin Hart, Matthew Hausknecht, Todd Hester, Yuqian Jiang, Piyush Khandelwal, Matteo Leonetti, Shih-Yun Lo, Patrick Macalpine, Jake Menashe, Sanmit Narvekar, Guni Sharon, Jivko Sinapov, Faraz Torabi, Daniel Urieli, Garrett Warnell, Harel Yedidsion, and Shiqi Zhang. I am particularly thankful to Ishan Durugkar, Piyush Khandelwal and Patrick Macalpine, with whom I had the pleasure of collaborating extensively.

During my years in the PhD program I have also been fortunate enough to be a member of the UT Austin Villa Standard Platform League RoboCup team. I have greatly enjoyed working in our shared codebase over the years. Being involved in RoboCup has led me to countless mean-

ingful insights, professional and otherwise, and I've also learned valuable lessons about teamwork, performing under stress, and the immutability of real-world deadlines.

I would like to thank my M.Sc. advisor, Prof. Benny Chor, for steering me down this treacherous, winding, but ultimately fruitful path. Had Benny not cajoled me into pursuing a graduate degree in computer science, the course of my life might have been completely altered. I will forever treasure his mentorship (and his kindness, and his whip-sharp sense of humor) as I was making my initial steps in academia.

Lastly, I would like to thank my family - my parents, Zipi and Itzhak, who supported me in every way possible throughout my life; my sister, Shira, who's always been there for me; my wife, Meital who went on this insane journey with me (hey Meital, we survived!); and my close friends, Jonathan Mueller and Moran Aharoni, for their staunch friendship and support through thick and thin. I would thank my children, Noam and Omer, who certainly did not help me work on my dissertation (quite the opposite), but did feel every waking moment in my life with a sense of determination and purpose. If I have accomplished anything, I have all of you to thank for it.

Elad Liebman

The University of Texas at Austin
May 2019

Sequential Decision Making in Artificial Musical Intelligence

Publication No. _____

Elad Liebman, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Peter Stone

Over the past 60 years, artificial intelligence has grown from a largely academic field of research to a ubiquitous array of tools and approaches used in everyday technology. Despite its many recent successes and growing prevalence, certain meaningful facets of computational intelligence have not been as thoroughly explored. Such additional facets cover a wide array of complex mental tasks which humans carry out easily, yet are difficult for computers to mimic. A prime example of a domain in which human intelligence thrives, but machine understanding is still fairly limited, is music.

Over the last decade, many researchers have applied computational tools to carry out tasks such as genre identification, music summarization, music database querying, and melodic segmentation. While these are all useful algorithmic solutions, we are still a long way from constructing complete music agents, able to mimic (at least partially) the complexity with which humans approach music.

One key aspect which hasn't been sufficiently studied is that of sequential decision making in musical intelligence. This thesis strives to answer the following question: **Can a sequential decision making perspective guide us in the creation of better music agents, and social**

agents in general? And if so, how? More specifically, this thesis focuses on two aspects of musical intelligence: music recommendation and human-agent (and more generally agent-agent) interaction in the context of music. The key contributions of this thesis are the design of better music playlist recommendation algorithms; the design of algorithms for tracking user preferences over time; new approaches for modeling people's behavior in situations that involve music; and the design of agents capable of meaningful interaction with humans and other agents in a setting where music plays a roll (either directly or indirectly).

Though motivated primarily by music-related tasks, and focusing largely on people's musical preferences, this thesis also establishes that insights from music-specific case studies can also be applicable in other concrete social domains, such as different types of content recommendation. Showing the generality of insights from musical data in other contexts serves as evidence for the utility of music domains as testbeds for the development of general artificial intelligence techniques. Ultimately, this thesis demonstrates the overall usefulness of taking a sequential decision making approach in settings previously unexplored from this perspective.

Contents

List of Tables

List of Figures

1. Introduction

Over the past 60 years, artificial intelligence has grown from a largely academic field of research to a ubiquitous array of tools and approaches used in everyday technology. Artificial intelligence technologies are virtually all around us – from sophisticated navigation systems and anti-collision sensors placed on cars to recommender systems meant to help us pick a book or movie. However, while great emphasis has been placed on improving the performance of such systems, certain meaningful facets have not been as thoroughly explored.

Such additional facets cover a wide array of complex mental tasks which humans carry out easily, yet are difficult for computers to mimic. These tasks include the human ability to understand social and cultural cues, to interpret and infer hidden meanings, to perceive the mental state of their counterparts, and to tailor their responses accordingly. A prime example of a domain in which human intelligence thrives, but machine understanding is still fairly limited, is music.

Music and technology are by no means strangers. Computers have been used for both the creation and the analysis of music almost since they first came into existence. Iannis Xenakis used computers in the early 1960s to generate numerical patterns, which he later transcribed into sheet music. A decade later, Pierre Boulez founded IRCAM (Institut de Recherche et Coordination Acoustic/Musique), where composers, computer scientists, and engineers study music and sound and new tools for creating electro-acoustic and computer-generated music. Two decades after, David Cope used statistical models to try and emulate classical composers [70]. Another important body of work in the application of computational tools in music is that of music information retrieval (MIR). Over the last decade, many researchers have applied computational tools to carry out tasks such as genre identification [49, 262], music summarization [95], music database querying [232] and melodic segmentation [307].

Broadly speaking, a wide array of algorithmic approaches and techniques has been used to tackle a variety of specific musical problems, ranging from note and chord transcription and beat detection to more high level tasks such as similarity analysis and mood detection. While all this work is extremely valuable, the problem of constructing complete music agents, able to mimic (at least partially) the complexity with which humans approach music, remains largely unsolved.

One key aspect which hasn't been sufficiently studied is that of sequential decision making in musical intelligence. Musical tasks are often studied in an isolated manner, extracting chords from a recording, or automatically tagging songs based on their similarity to other, previously tagged songs. Music, however, is inherently sequential. And yet, even tasks which are clearly sequential, such as automatic accompaniment generation, have rarely been studied from an autonomous decision making perspective, i.e. considering that the agent's decisions impact its future sensations and

circumstances. This thesis aims to bridge this gap. It focuses on important cases where framing the task as a sequential decision making problem explicitly is beneficial in the design of autonomous agents in music domains, and connects an agent-based perspective with the need to explicitly reason about people's preferences and intentions over time on multiple timeframes in an organic fashion.

From a more general perspective, this thesis strives to show that treating various cultural tasks as sequential decision making processes is a powerful paradigm for connecting useful but ultimately limited components (knowledge representation, preference elicitation, model learning, etc.) into a complete autonomous agent. This thesis also illustrates how insights from the creation of autonomous agents in music domains can be generalized to be broadly applicable in other cultural and social tasks.

1.1 Research Question and Contributions

With this motivation in mind, this thesis answers the following concrete question:

Can a sequential decision making perspective guide us in the creation of better music agents, and social agents in general? And if so, in what ways?

More specifically, this thesis focuses on two aspects of musical intelligence: music recommendation and multiagent interaction in the context of music, with a special emphasis on person-agent interaction. The key contributions of this thesis are the following:

1. **Delineating a taxonomy of musical intelligence.** The intersection of artificial intelligence and music is a wide, complex and fractious domain, spread across disciplines, paradigms and perspectives. An important step forward in the creation of complex music agents is a clear unified outline of what the state of affairs in artificial intelligence is, which tasks have been attempted and to what extent, which approaches have been used in which contexts, and what gaps remain to be addressed. While others have reviewed more specific questions relating to music and artificial intelligence, such as music recommendation or robotic musicianship, no effort that I am aware of has been made to provide a high level overview of the research lying at the intersection of AI and music, specifically from an AI perspective, in a way that provides some structure to an otherwise rather complicated and disparate body of work. In Chapter 8 I provide a new, broad perspective on previous literature, and propose an overall taxonomy, or framework, useful for examining previous work in context.

2. **Studying the role of sequential decision making in music recommendation.** A specific domain in which intelligent agents, music analysis, and preference elicitation connect is in the construction of musical recommender systems. Most of these systems focus on predicting the preference of individual songs independently based on a learned model of a listener (see

[172, 19, 189], for example). However, a relatively well-known fact in the subfield of cognitive neuroscience, which studies music cognition, is that music is experienced in temporal context and in sequence [190]. Therefore, considering music recommendation as a sequential decision making problem, where past recommendations impact listeners' expectations and mood, is a good example of how this perspective can have an impact in a real-world music domain that is steadily growing in importance and public interest. In this thesis I provide evidence that introducing sequential considerations into the process of learning music preference and designing playlists have a substantial positive impact on listener experience. More importantly, I propose a novel, online, reinforcement-learning based framework for online music recommendation that adapts quickly (within the span of a single listening session) and accounts for sequence as well as single song preferences. To the best of my knowledge this work is the first such algorithm. This contribution is studied at length in chapter 3.

3. **Studying the ability of an artificial agent to model the impact of music on people's behavior.** One of the key issues in building a successful social agent with respect to music is its ability to model and predict human responses to musical cues. This thesis explores whether artificial agents can successfully model certain aspects of music perception in human listeners. From an agent perspective, one critical aspect that falls under this category is modeling how music stimuli affect human behavior under different circumstances. Demonstrating that an agent can successfully do so is a valuable contribution not only in the obvious context of music recommendation, but also, more broadly, to our understanding of human cognition. It also serves as a vital step towards better human-agent interaction in musical environments (and cultural environments in general). To the best of my knowledge, the work represented in this chapter is the first to study in a principled way the differential impact of different kinds of music on human decision-making. This contribution is addressed in Chapter 5.

4. **Designing agents that are able to meaningfully interact with people in musical environments.** Following up on the previous contribution, it is largely an open question of whether agents can utilize their models of people's behavior in a musical context to improve their interactions with people in musical-influenced environments. Simply put, it is not enough to learn a model of people's behavior - it is crucial to learn strategies for how to act with respect to that information, under the assumption that the agent's actions also affect people's perception and behavior. In this thesis I present agents that are able to carry out meaningful interaction with people in a domain that is either directly music-related, or impacted by music (for instance, music is playing in the background), with music being directly used in the agent's representation of the world. Furthermore, I also introduce an approach for multiagent music collaboration in a setting in which each agent has its own individual preferences - an important step towards more organic and general agent-person music generation platforms. The key work that constitutes this contribution is in Chapters 6 and 7. The work in Chapter

6 is the first I am aware of that both studies the impact of background music on complicated sequential person-agent interaction, and demosntrates how this knowledge can be utilized *by an agent* in order to learn a more effective policy. The work in Chapter 7 studies a novel, broadly applicable reinforcement learning learning framework for blending individual agent preferences with shared objectives. Furthermore, this chapter highlights under which conditions individual preferences could be beneficial in multiagent cooperation.

5. **Proposing general algorithms for tracking changes in user preference.** Another challenge that draws both from the literature on the intersection of music and artificial intelligence, and from the sequential decision making literature, is that of tracking changes in people's aesthetic preference. Music, like any cultural domain, is transient - tastes change over time, new fashions appear and disappear in patterns which are often difficult to predict. It is an open challenge to efficiently track all these idiosyncratic changes in taste. In this thesis I formulate the model retraining problem as a sequential decision making task and propose algorithms for model management over time in the face of changing data distributions. This contribution has potential applications in content recommendation, and could also prove useful in more general cases in which predictive models need to be maintained over time (such as automated control). To date, this is the first (and only) work that conceptualizes concept drift adaptation as a reinforcement learning problem and it provides ample evidence for the potential benefits of such an approach. Chapter 4 primarily studies this contribution.

6. **Showing the applicability of insights gathered from music domains in other types of content-based interaction**. A key hypothesis in this thesis is that many of the contributions discussed above have impact that goes well beyond the scope of musical domains exclusively. It is therefore an explicit goal of this thesis to show concrete examples where approaches and insights pertaining to music domains can be generalized successfully in other useful contexts. Such examples include, but are not limited to, showing how continual model learning in the context of musical preference (i.e. Contribution 5), is also largely applicable in continual model management in general; or showing how an approach for multiagent music collaboration in face of different individual preferences (which is an aspect of Contribution 4) is also applicable in a more traditional multiagent reinforcement learning task. Both Chapter 4 and 7 establish this contribution.

If we envision a future where intelligent artificial agents interact with humans, we should strive to give AI the ability to understand and communicate within cultural settings. When taken together, the contributions of this thesis are designed to take meaningful steps in this direction. By combining multiple threads together into a coherent outline, this thesis will get us closer to the final goal of designing complete cultural agents, focusing on their systemic and sequential nature. Beyond its value in helping us understand human culture and behavior, this thesis has multiple

potential non-music-related real world practical benefits, ranging from recommender systems and business intelligence to negotiations and personalized human-computer interaction.

1.2 Thesis Structure

The remainder of this document is organized as follows:

- **Chapter 2 - Background**. In this chapter I discuss information that is needed to understand the work presented in this thesis. Specifically, I provide an introductory overview of reinforcement learning as a discipline; I provide a broad overview of the intersection of music and AI research, focusing specifically on music information retrieval and music recommendation; and I provide some context relating to the literature on music and human behavior. A more in-depth review of the extensive related work is provided in Chapter 8.

- **Chapter 3 - Playlist Recommendation**. In this chapter, in fulfillment of Contribution 2 of this thesis, I focus on playlist recommendation as a prime example for a music recommendation task that is better formulated as a sequential decision-making task. I provide evidence that such a consideration is indeed in line with empirically observed preferences of participants in a lab experiment, and propose an agent architectures for recommending music sequentially.

- **Chapter 4 - Algorithms for Tracking Changes in Preferences over Time**. This chapter tackles the problem of preferences changing abruptly or gradually over time, and considers it an instance of concept drift, a problem that is inherent to many learning systems that need to adapt to data continually. I formulate the model adaptation problem as a sequential decision-making task and study multiple approaches for a meta-agent that can detect when and in what way to adapt underlying models. I show that this approach works well on a music preference related dataset. In this chapter I also provide evidence that this approach works well on an unrelated real-world dataset (an internal dataset provided by AT&T Research, meant for internal systems monitoring). By showing the potential benefits of the proposed approach in non-music-related, real-world settings, this chapter also partially supports Contribution 6 of this thesis.

- **Chapter 5 - Modeling the Impact of Music on Human Decision-Making**. As discussed in Contribution 3, any learning agent in a music setting needs to be able to model how music affects people's preferences and behavior. In fulfillment of Contribution 3, this chapter presents an example framework for modeling the impact of music on human decision-making and study it experimentally in the context of two very different tasks - classifying words by their emotional content and deciding whether to accept or reject gambles. I not

only show that the impact of music on human behavior can be modeled successfully in both cases, but also analyze how different musical features can be related to different aspects of human decision-making in the context of these tasks.

- **Chapter 6 - Impact of Music on Person-Agent Interaction**. Building on the framework introduced in Chapter 5, in this chapter I take this approach one step forward and propose an agent that needs to interact with a person in a social setting - crossing a simulated intersection. In this chapter, in partial fulfillment of Contribution 4, I show how explicitly considering background music in the context of this task is beneficial to the agent's own decision-making, and suggest that this outcome is because considering music in the agent's state representation enables it to better predict human behavior in this setting.

- **Chapter 7 - Multiagent Collaboration Learning: a Music Generation Case Study**. If we envision a setting in which agents and people can collaborate in a musical setting, such agent frameworks must be able to contend with individual *preferences*. The concept of having multiple agents with preexisting and potentially contradictory individual preferences learning to cooperate on a shared task is in many ways a novel concept in reinforcement learning, with direct applications not only to music collaboration and generation (in further accordance with Contribution 4), but also in a more traditional multiagent reinforcement learning problem (the pursuit domain).

- **Chapter 8 - Related Work and a Taxonomy of Music Intelligence Tasks**. In this chapter I provide an in-depth overview of work at the intersection of artificial intelligence and music. To satisfy Contribution 1, and better contextualize the work presented in Chapters 3 - 7, I delineate a taxonomy that breaks down the complex and varied literature that inhabits this space in three dimensions: nature of the task, type of input, and proposed technique. I identify the biggest gaps in the literature as scaling from small, atomic tasks to more complex systems that are able to interact with people over time. This chapter thus helps highlight Contributions 2 - 5 as studied in this thesis.

- **Chapter 9 - Summary and Discussion**. In this chapter I conclude by providing a recap of the work discussed in previous chapters and discuss future directions for each topic discussed in this thesis.

- **Appendix - Glossary**. This appendix lists the terms used in this thesis and a brief explanation for them.

In concordance with Contributions 1 - 6, this thesis explores several themes:

- *Online Recommendation* (with a focus on *music* recommendation) - this thesis deals considerably with the problem of recommending content on the fly, either during a single session (Chapter 3) or contending with evolving taste over time (Chapter 4).

- *Reasoning about Preferences* - both Chapter 3 and Chapter 7 involve the problem of agents reasoning about others' preferences. In Chapter 3, the agent needs to explicitly model a person's taste in music. In Chapter 7, each agent must reason about the other agents' preferences, at least implicitly, in order to learn how to coordinate in a way that satisfies their own taste as well as aids the agents in achieving a shared goal (successfully generating music that pleases an external arbiter).

- *Multiagent Interaction in Music Contexts* - both Chapter 6 and Chapter 7 involve agents operating in a multiagent environment that involves a musical facet. In Chapter 6, an agent needs to interact with a person in a driving simulator while music is playing in the background. In Chapter 7, all agents must explicitly coordinate in order to generate a sequence of chords together, and in doing so balance their individual preferences with the requirements of an external listener.

- *Modeling the Impact of Music on Decision-Making* - both Chapter 5 and Chapter 6 explicitly explore whether (and how) agents can model the impact of music on human decision-making.

A visual illustration of the contents of this thesis as organized by chapters and themes is provided in figure 1.1.

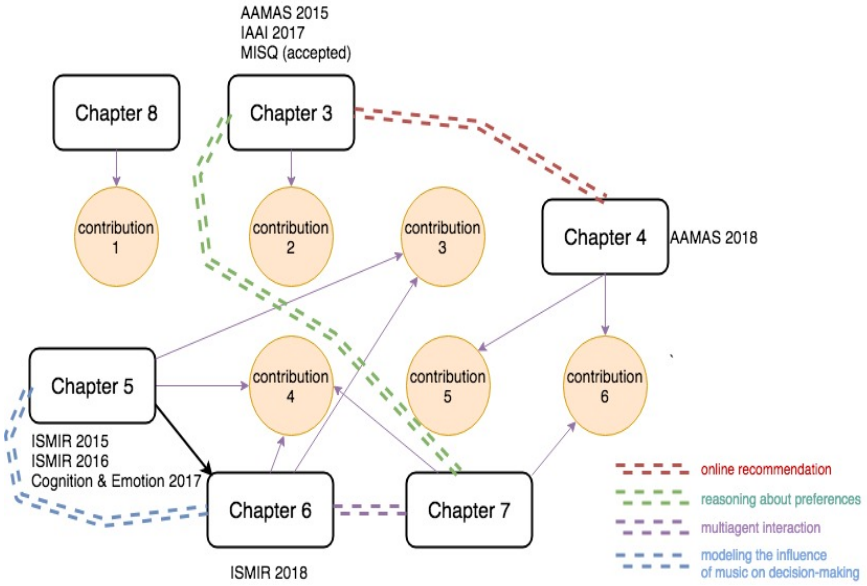

Figure 1.1: A visual illustration of the structure of this thesis, how the different chapters correspond to the contributions of this thesis, and how each chapter connects with other chapters thematically. In addition, if the work in a given chapter has been published, it is mentioned where and when.

Having discussed the research question, goals and contributions of this thesis, and having outlined the overall structure of this thesis, in the next chapter I will provide the background needed in order to better understand the work carried out in Chapters 3 - 7.

2. Background

The purpose of this section is to provide the reader with context helpful for understanding the contributions this thesis. Whereas a fuller review of the (extensive) literature lying at the intersection of AI and music is provided in Chapter 8, in this chapter I focus on the key frameworks this thesis utilizes. In Section 2.1 I present the key learning framework used in this thesis, which is reinforcement learning. In Section 2.2 I provide a broad overview of music AI research, with a particular focus music recommendation, which is a recurring theme in this thesis (particularly in Chapters 3 and 4, but also relevant to Chapter 7), and on the gaps this thesis aims to address. Because human behavior with respect to music is another key aspect of this thesis (particularly in Chapters 3, 5, and 6), in Section 2.3 I provide context relating to the impact of music on human behavior, and discuss how this thesis addresses gaps in this respect as well.

2.1 Reinforcement Learning

Since this thesis revolves in large part around sequential decision-making problems, it is only natural that the core learning framework utilized in it is that of Markov Decision Processes [374]. Markov Decision Processes, or MDPs, are the standard mathematical framework in reinforcement learning, generally used to describe an agent interacting with its environment (see Figure 2.1). In its most general form, an MDP is a 6-tuple of the form (S, A, P, R, T, γ). S represents the state space the agent can be in.[1] A represents the space of actions the agent can take. $P(s, a, s') \rightarrow [0, 1]$ is a (possibly stochastic) transition function which returns the probability of getting to state s' when taking action a in state s. $R(s, a) \rightarrow \mathbb{R}$ is a (possibly stochastic) reward function denoting the one-step value of taking action a in state s. T is a set of terminal states. In *episodic* MDPs, reaching a state $s \in T$ terminates the episode. If the MDP is *continuing*, T is empty and the MDP never terminates. $\gamma \in [0, 1]$ is the discount factor, and denotes how much priority should be given to immediate versus long-term rewards. $\gamma = 1$ implies no discounting, in which case payoff now has the same weight as payoff in a future step.

Broadly speaking, *Reinforcement Learning* (or RL) is a machine learning paradigm that focuses on learning to choose a sequence of actions which maximizes the cumulative reward in a given MDP. A reinforcement learning agent aims to learn a policy $\pi(s \in S) \rightarrow a \in A$, a function from states to actions, which dictates how the agent operates within the MDP. This discounted sum of rewards the agent accrues over time is referred to as the return, $J = \sum_{t=0}^{\infty} \gamma^t r_t$. An optimal

[1] If the states are not fully revealed to the agent, then the problem becomes a Partially Observable MDP (or POMDP). Though many real world problems can be modeled more accurately as POMDPs rather than regular MDPs, this distinction is beyond the scope of this thesis.

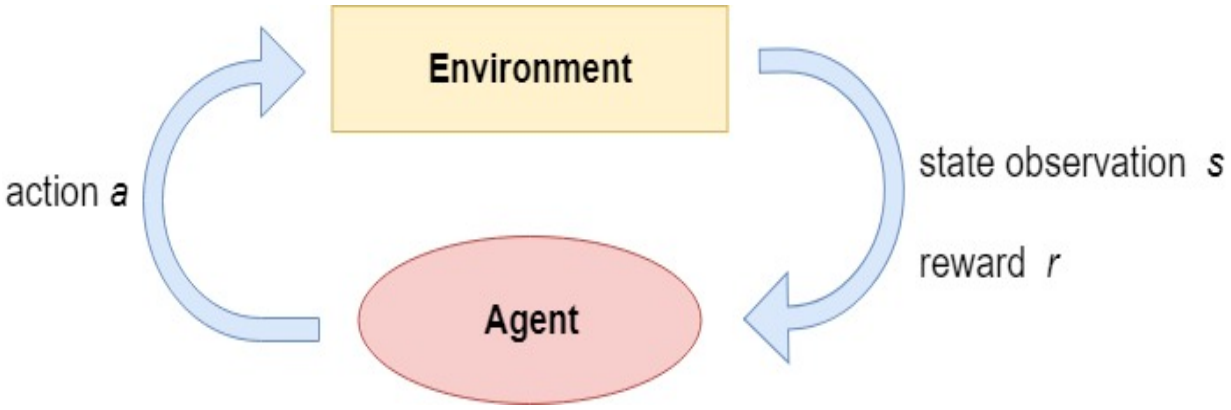

Figure 2.1: A high level visual illustration of the basic premise of the reinforcement learning (RL) paradigm. At each timestep an agent observes its state s with respect to its interaction with the environment, decides on an action a, and then observes a new state s' and receives a reward signal r. The purpose of an RL agent is to learn a policy (a mapping from states to actions $\pi(s \in S) \rightarrow a \in A$) which maximizes its expected rewards over time.

policy π^* is a policy which maximizes the expected return for a given agent. A policy can only be considered optimal if an agent operating under it takes an optimal action $a^* \in A$ in any given state s.

A large family of reinforcement learning algorithms is based on the idea of a *Value Function* $V(s)$, estimating the expected return once reaching a state s. Computing the optimal policy given the value function can be done via *Value Iteration*, a dynamic programming algorithm for calculating the optimal state values through the recurrence relation $V^*_{i+1} = max_{a \in A}(R(s,a) + \sum_{s' \in S} P(s,a,s')V^*_i(s')$. Performing this computation until it converges necessitates knowledge of the MDP transition dynamics through P, which is often unknown. To avoid this additional hardship, an alternative learning approach arises, which focuses on learning the value of *state-action pairs* directly rather than states alone. This approach gives rise to the $Q(s,a)$ function, a state-action analogue to the state value function. Using Q values allows an agent to learn directly from experience. This approach is referred to as *model-free*. Alternatively, an agent may explicitly learn and represent the MDP transition dynamics from observation using a *model*, giving rise to *model-based* reinforcement learning. Model-based approaches are more sample efficient, because they leverage and generalize information better, but tend to be very expensive in terms of necessary computation. Model-free approaches are light in computation but far less sample efficient, and often require a great deal of samples in order to converge to an optimal policy.

Despite the usefulness of value-function based methods, in certain cases it is possible and in fact advantageous to instead try and learn the policy directly, without the use of an intermediate value function representation. This approach is called *Policy Search* and it has proven effective in many cases in which ascribing values to any given state or state-action pair is infeasible but

estimating the value of an overall policy is possible. A large family of methods exist for such policy search based on approximating a policy gradient and optimizing policy parameters [375, 235, 359]. A common paradigm for policy gradient methods is that of *Actor-Critic*. Actor-critic methods [186], a learning paradigm which separates the action selection and policy update (actor) from the evaluation (critic), typically leverage both approaches in such a way that the actor operates in policy space directly while the critic maintains a value function approximation. One such actor-critic policy gradient algorithm, PPO [354], is used in Chapter 7.

For additional details please see [374].

2.2 AI, Music and Agents

As laid out in Chapter 1, while much of this thesis aims to offer general takeaways, the majority of domains and indeed the focal point of this thesis all lie at the intersection of AI and music. There are many ways in which artificial intelligence and music intersect, ranging from analysis of large bodies of existing music to the creation of music itself. Though of interest since the early days of computing, the recent years have seen a veritable explosion in research involving music and AI, driven in large part by the emergence of the Internet and digital music frameworks. Over the last decade, many researchers have applied computational tools to carry out a wide array of musical tasks, ranging from audio classification [398, 399] to music generation [38, 36, 225]. For a much deeper overview of this varied and complex body of work see Chapter 8. In that chapter, I propose a taxonomy for organizing this vast literature along three axes: nature of the task, input type, and algorithmic technique.

This thesis takes a particular interest in music recommendation, with a special emphasis on temporal aspects of music recommendation (Chapters 3 and 4), and multiagent music collaboration (7). The standard task of music recommendation is typically framed as a classification problem: will a given user u_i like a given song s_k. This problem is typically framed as a collaborative filtering problem [345]. However, in this thesis (and particularly in Chapter 3) I assume a different approach: songs are rarely if ever listened to in isolation, one song at a time. Instead, it is broadly recognized that music is typically enjoyed *in sequence*, and that the ordering of the songs alters people's perception and experience of music. In Chapter 4 I also identify that people's musical preferences are continually in flux over time, and that any music recommendation platform must contend with this aspect of human preferences.

Indeed, a glaring gap in the existing music AI literature revolves around the role of *agency* in music AI systems. While the work at the intersection of AI and music is rich and varied, the extent of AI work applied to music that has assumed an agent-based, or systemic, perspective, is somewhat limited. None of these previous works systematically looked at the overall usefulness of applying sequential decision-making techniques towards the development of complex AI system in music-related domains, as I do in this thesis. Though certain low-level skills necessary for music

understanding, such as chord and key recognition or beat extraction, may be considered solved (or close enough to it), the same cannot be said for more complex levels of abstraction. Spanning across disparate aspects of musical intelligence such as playlist recommendation (Chapter 3), agent-person collaboration (Chapter 6) and multiagent music generation (Chapter 7), in this thesis I methodically propose music *agents*, entities capable of sensing aspects of the environment they operate in, making music-related decisions, and learning from experience.

2.3 Music and Human Behavior

Another key facet of this thesis revolves around modeling people's perception and behavior with respect to music, both in terms of preferences (as reflected particularly in Contribution 2) and in connecting agency to modeling human behavior with computational perspectives in musical context, as presented in Contributions 3 - 4.

Indeed, the questions of how music impacts people's decision making, and whether these impacts can be both learned in real-time and leveraged, have been under-explored in the literature both from a machine learning and from a cognitive science perspective (for further details see Chapters 5 and 6). Even though the idea of human-agent interaction in music has accompanied electro-acoustic music as early as the 1970s and the 1980s [269], not much work on agents directly reasoning about the preferences and intentions of other agents in an interactive fashion exit. This thesis bridges the gap, particularly in Chapters 6 and 7, in which agents need to reason explicitly about others' mental state in a completely interactive environment.

2.4 Summary

In this chapter I provided the core background needed to understand the frameworks this thesis as a whole operates in and how the contributions of this thesis fit within these frameworks. I discussed the machine learning paradigm of *reinforcement learning* and the mathematical framework of *Markov Decision Process* which will recur throughout chapters 3 - 7. I provided context on the incorporation of agent-based approaches in music AI research, and an overview of music AI research in general (for a much more in-depth review please see Chapter 8). Lastly, I discussed the intersection of AI and music within the scope of both analyzing human perception of music and designing interactive human-agent systems with music in mind.

This short but essential review provides us both with the core toolset used in the thesis and with a better understanding of the motivation behind it. With these in mind, I am able to dive into the technical contributions of this thesis, discussed at length in Chapters 3 - 7. In the next Chapter, I will focus on one particular instance of agent-person interaction, connecting sequential decision-making, music recommendation and online adaptation.

3. Playlist Recommendation

Music is one of the most widespread and prevalent expressions of human culture. It has accompanied the human experience throughout history, and the enjoyment of music is one of the most common human activities. As an activity, music listening sessions commonly span over a sequence of songs, rather than a single song in isolation. Importantly, it is well established that music is experienced in temporal context and in sequence [79, 175]. This phenomenon not only underlies the notion of structure in music (as in the canonical sonata form [68]), but also implies that the pleasure one derives from a complete song is directly affected by its relative position in a sequence. This notion also underlies the manner in which DJs construct playlists [61], and indeed, research on automated playlist construction has aimed to produce generally appealing playlists [290, 72]. However, such works have not considered the construction of personalized playlists tailored to *individual* users' preferences.

In the field of recommender systems, [5] music has been of particular interest, both academically [5, 287] and commercially [19]. Pandora, Jango, and Last.fm are some examples of popular contemporary commercial applications. To the best of our knowledge, however, research on *personalized* music recommendations has focused mostly on predicting users' preferences over *individual* songs, rather than song *sequences*.

Overall, there has been little effort to relate learning individual listener preferences with holistic playlist generation. In this chapter, in fulfillment of Contribution 2 of my thesis, I propose a framework for *adaptive, personalized* music *playlist* generation. This framework casts the online playlist learning and adaptation problem as a Markov Decision Process (MDP) and learns preferences over songs and transitions on the fly. Towards this, I discuss and identify key design challenges for adapting to an individual's playlist preferences based exclusively on online experiences. This task entails that learning ought to be data-efficient: for transitions to improve and adapt to a user's sequence preference based on little data, adaptation must effectively exploit information from a relatively small number of experiences. I then present thorough empirical studies that explore the properties of the framework we proposed over both simulated data derived from playlists data sets as well as experiments with real human listeners. The empirical evaluations aim to explore whether the sequence of songs indeed has a significant effect on a listener's experience, and to evaluate our framework's ability to learn and adapt effectively to individual users' preferences for song sequences exclusively from online experiences. I then examine the contribution of the proposed design choices towards effective online adaptation. Lastly, I consider the problem of utilizing learned personalized recommendation models to plan sequences effectively in the playlist recommendation domain, and propose a novel planning algorithm for music playlist recommendation. I subsequently analyze the

proposed approach empirically with respect to two utility metrics: personalization and diversity.

The work presented in this chapter illustrates the benefits of considering music recommendation as a sequential decision making problem, both as a real-world testbed for reinforcement learning research and as a means for providing better quality recommendation services in this fast-growing platform. This chapter is based on published research conducted jointly with Maytal Saar-Tsechansky [218, 219], and Piyush Khandelwal [216].

3.1 Music Playlist Recommendation as an MDP

If we consider playlists as depending on the specific sequence of songs, such that the enjoyment of each song is dependent on the songs chosen before it, then at each point in the playlist generation process, the selection of a new song affects the choice of possible songs in the future. From this perspective, playlist recommendation is a sequential decision-making task, and as such, is suitably formulated as a Markov Decision Process (MDP) [373].

Formally, an episodic MDP is a tuple (S, A, P, R, T) where S is the set of states; A the set of actions, $P : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function where $P(s, a, s') = r$ denotes the probability of transitioning from state s to state s' when taking action a. $R : S \times A \rightarrow \mathbb{R}$ is the state-action reward function, where $R(s, a) = r$ means that taking action a from state s will yield reward r. T is the set of terminal states, which end the episode. A more extensive description of the reinforcement learning paradigm is provided in Chapter 2.

For the purposes of our specific application, consider a finite set of n musical tracks (songs) $\mathcal{M} = \{a_1, a_2, \ldots, a_n\}$ and assume that playlists are of length k. Our MDP formulation of the music playlist recommendation task is then as follows.

- To capture the complex dependency of listener experience on the entire sequence of songs heard, a Markov state must include an ordered list of all prior songs in the playlist. Thus, the state space S is the entire ordered sequence of songs played, $S = \{(a_1, a_2, \ldots, a_i) | 1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}$.

  That is, a state $s \in S$ is an ordered tuple of songs ranging in length from 0 when choosing the first song of the playlist to k when the playlist is complete.

- The set of actions A is the selection of the next song to play, $a_k \in A$. This means that the action space is exactly the set of songs: $A = \mathcal{M}$.

- These definitions of S and A induce a deterministic transition function P. As such, we can use the shorthand notation $P(s, a) = s'$ to indicate that when taking action a in state s, the probability of transitioning to s' is 1, and to $s'' \neq s'$ is 0. Specifically, $P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)$.

- $R(s, a)$ is the utility (or pleasure) the current listener derives from hearing song a when in state s. Note that this formulation implies that each listener induces a unique reward function. A key challenge addressed in this chapter is enabling efficient learning of R for a new listener.
- $T = \{(a_1, a_2, \ldots a_k)\}$: the set of playlists of length k.

Solving an MDP typically refers to finding a policy $\pi : S \rightarrow A$ such that from any given state s, executing action $\pi(s)$ and then acting optimally (following the optimal policy π^*) thereafter, yields the highest (expected) sum of rewards over the length of the episode. In our case, since P is deterministic, π^* corresponds to the single sequence of songs that would be most pleasing to the listener.[2] However, we assume that the listener's reward function R is initially unknown. We consider the fundamental challenge of playlist generation as being efficiently modeling R.

In particular, in the reinforcement learning literature, there are two high-level approaches to approximating (learning) π^*: model-free and model-based. *Model-free* approaches learn the value of taking an action a from state s directly. Typical approaches, such as Q-learning and SARSA [373] are computationally efficient and elegant, but require a lot of experiential data to learn. *Model-based* approaches alternatively learn the transition and reward functions (P and R) so as to be able to *simulate* arbitrary amounts of experiential data in order to find an approximate solution to the MDP in an approach that can be thought of as *planning* through forward lookahead search. Compared to model-free methods, most model-based algorithms are significantly more computationally expensive, especially if they re-solve the MDP whenever the model changes. However, in many applications, including playlist recommendation, where data is considerably more scarce than computation, this tradeoff of computational expense for data efficiency is a good one. We therefore adopt a model-based learning approach in this chapter (see Sections 3.2 and 3.4 for details).

In the MDP defined above, the transition function P is trivially known. Therefore the only unknown element of the model necessary for model-based learning is R, the current listener's utility (enjoyment) function. Indeed modeling R in such a way that generalizes aggressively and accurately across both songs and song transitions is the biggest technical challenge in this work. Consider that even for a moderately sized music corpus of 10^3 songs, and for playlist horizons of 10 songs, the size of the state space alone would be 10^{30}. It is impractical for a learning agent to even explore any appreciable size of this state space, let alone learn the listener's utility for each possible state (indeed our objective is to learn a new user's preferences and generate a personalized song sequence within a single listening session of 25–50 songs). Therefore to learn efficiently, the agent must internally represent states and actions in such a way that enables generalization of the listener's preferences.

Having laid out the framework used to formally describe the playlist recommendation problem discussed in this chapter, I now present DJ-MC, a concrete learning agent framework for online,

[2] We consider the problem as finding a single playlist in isolation, ignoring the fact that the same listener may not want to hear similar sequences repeatedly. In practice, the stochasticity of our approach makes it exceedingly unlikely that the same sequence would be presented to a given listener multiple times, as will become clear below.

adaptive playlist recommendation. In Sections 3.2 - 3.4 I describe the various design components for the agent architecture; In Sections 3.5 - 3.6 I present the empirical analysis of the proposed agent architecture; and in Sections 3.7 - 3.10 I discuss planning extensions to the core agent architecture.

3.2 Modeling

As motivated in the previous section, learning a listener's preference function over a large set of songs and sequences requires a compact representation of songs that is still rich enough to capture meaningful differences in how they are perceived by listeners. To this end, we represent each song as a vector of song *descriptors*.

Specifically, in this chapter, the playlist recommendation agent DJ-MC uses spectral auditory descriptors that include details about the spectral fingerprint of the song, its rhythmic characteristics, its overall loudness, and their change over time. We find that these descriptors enable a great deal of flexibility (for instance, in capturing similarities between songs from vastly different backgrounds, or the ability to model songs in unknown languages). Nonetheless, our framework is in principle robust to using any sufficiently expressive vector of song descriptors. Section 3.2.1 specifies in detail the descriptors used by DJ-MC.

In order to further speed up learning, we make a second key representational choice, namely that the reward function R corresponding to a listener can be factored as the sum of two distinct components: 1) the listener's preference over *songs* in isolation, $R_s : A \to \mathbb{R}$ and 2) his preference over *transitions* from past songs to a new song, $R_t : S \times A \to \mathbb{R}$. That is:

$$R(s, a) = R_s(a) + R_t(s, a) \tag{3.1}$$

Sections 3.2.1 - 3.2.2 describe DJ-MC's reward model in detail. Section 3.2.3 then evaluates the extent to which the chosen descriptors are able to differentiate meaningfully between song sequences that are clearly good and clearly bad.

3.2.1 Modeling Songs

As motivated above, we assume each song can be factored as a vector of scalar descriptors that reflect details about the spectral fingerprint of the song, its rhythmic characteristics, its overall loudness, and their change over time. For the purpose of our experiments, we used the acoustic features in the Million Song Dataset representation [33] to extract 12 meta-descriptors, out of which 2 are 12-dimensional, resulting in a 34-dimensional song descriptor vector. The complete set of descriptors is summarized in Table 3.1 in Section 3.3.

3.2.2 Modeling The Listener Reward Function

Despite an abundance of literature on the psychology of human musical perception [377], there is no canonical model of the human listening experience. In this work we model listening as being dependent not only on preferences over the descriptors laid out above, but also over feature *transitions*. This model is fairly consistent with many observed properties of human perception, such as the stochastic dependence on remembering earlier events, and evidence of working memory having greater emphasis on the present [79, 34, 377].

We now proceed to specify the two components of R: R_s and R_t.

Listener Reward Function over Songs R_s

To model R_s, we use a sparse encoding of the song descriptors to generate a binary feature vector. R_s is then a linear function of this feature vector: that is, we assume that each feature contributes independently to the listener's utility for the song.

Specifically, for each song descriptor, we collect statistics over the entire music database, and quantize the descriptor into 10-percentile bins. Following standard reinforcement learning notation, we denote the feature vector for song a as $\theta_s(a)$. It is a vector of size $\#bins \times \#descriptors = 10 \times 34 = 340$ consisting of $\#descriptors$ containing 1's at coordinates that correspond to the bins song a populates, and 0 otherwise, meaning $\theta_s(a)$ behaves as an indicator function (the weight of $\theta_s(a)$ will be 34 overall).

For each feature, we assume the listener has a value representing the pleasure they obtain from songs with that feature active. These values are represented as a weight vector $\phi_s(u)$. Thus $R_s(a) = \phi_s(u) \cdot \theta_s(a)$. The parameters of $\phi_s(u)$ must be learned afresh for each new user.

Listener Reward Function over Transitions R_t

A main premise of this work is that in addition to the actual songs played, a listener's enjoyment depends on the *sequence* in which they are played. To capture this dependence, we assume that

$$E[R_t((a_1, \ldots, a_{t-1}), a_t)] = \sum_{i=1}^{t-1} \frac{1}{i^2} r_t(a_{t-i}, a_t)$$

where $r_t(a_i, a_j)$ represents the listener's utility for hearing song a_j sometime after having heard a_i. The term $\frac{1}{i^2}$ represents the notion that a song that was played i songs in the past has a probability of $\frac{1}{i}$ of affecting the transition reward (i.e. being "remembered"), and when it does, its impact decays by a second factor of $\frac{1}{i}$ (its impact decays over time).

It remains only to specify the song to song transition reward function $R_t(a_i, a_j)$. Like R_s, we can describe R_t as a linear function of a sparse binary feature vector: $R_t(a_i, a_j) = \phi_t(u) \cdot \theta_t(a_i, a_j)$ where $\phi_t(u)$ is a user-dependent weight vector and θ_t is a binary feature vector.

Were we to consider the transitions between all 340 features of both a_i and a_j, θ_t would need to be of length $340^2 > 100,000$. For the sake of learnability, we limit θ_t and ϕ_t to only represent transitions between 10-percentile bins of the same song descriptors. That is, for each of the 34 song descriptors, there are 100 features, one of which is 1 and 99 of which are 0, indicating which pair of 10-percentile bins were present in songs a_i and a_j. Therefore, overall, θ_t consists of 3,400 binary features, 34 of which are 1's.

Clearly, this representation is limiting in that it cannot capture the joint dependence of listener utility on transitions between multiple song descriptors. Especially for the pitch class features, these are likely to be relevant. We make this tradeoff in the interest of enabling learning from relatively few examples. Empirical results indicate that this representation captures enough of real peoples' transition reward to make a difference in song recommendation quality.

Like $\phi_s(u)$, the parameters of $\phi_t(u)$ must be learned afresh for each new user. Thus all in all, there are 3740 weight parameters to learn for each listener.

With even that many parameters, it is infeasible to experience songs and transitions with all of them active in just 25 songs. However DJ-MC is able to leverage knowledge of even a few transition examples to plan a future sequence of songs that is biased in favor of the positive ones and against the negative ones.

3.2.3 Expressiveness of the Listener Model

This representation of the listener's utility function as a 3740-dimensional sparse binary feature vector is just one of many possible representations. A necessary property of a useful representation is that its features are able to differentiate between commonly perceived "good" vs. "bad" sequences, and the DJ-MC agent internally relies on this property when modeling the listener reward function. To evaluate whether our features are expressive enough to allow this differentiation, we examine the transition profile for two types of transitions, "poor" vs. "fair", both derived from the same population of songs. We generate "fair" transitions by sampling pairs of songs that appeared in an actual sequence. We generate "poor" transitions by interleaving songs so that each one is distinctly different in character (for instance, a fast, loud track followed by a soft piece). The difference between the two profiles can be seen in Figure 3.2.3. More definitive evidence in favor of the adequacy of our representation is provided by the successful empirical application of our framework, discussed in Section 3.6.

3.3 Data

A significant component of this work involves extracting real-world data for both songs and playlists to rigorously test our approach. In this section we discuss the different data sources we used to model both songs and playlists. For songs, we relied on the Million Song Dataset [33], a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

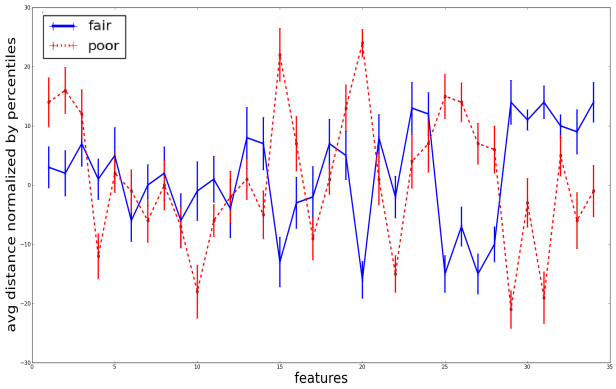

Figure 3.1: Example of fair vs. poor transition profiles, based on the same set of 20 songs. The plot shows the average transition delta for each feature. Both the fair transitions and the poor ones are constructed from the same core set of 20 songs taken from 5 different albums. In the case of fair transitions, we maintain the original order. In the case of poor transitions, the albums are randomly interleaved. The results indicate that qualitatively different sequences are indeed distinguishable in our feature model. In this specific example, 19 of the 34 features are discriminative (confidence intervals do not overlap). We expect different features to be discriminative for different transition profiles.

The dataset covers $44,745$ different artists and 10^6 different tracks. All the features described in Table 3.1 are derived from this representation. An example of the audio input for a single track is presented in Figure 3.3. It should be noted that our agent architecture (described in detail in Section 3.4) is agnostic to the choice of a specific song corpus, and we could have easily used a different song archive.

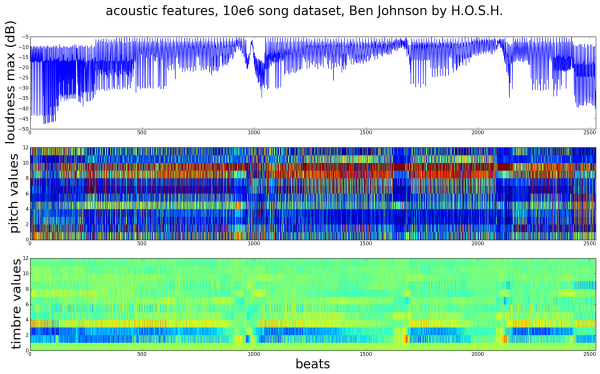

Figure 3.2: Example of loudness, pitch and timbre data for an example track over time (time units are in beats). Loudness is in dB, pitch measurements are represented as a vector of 12 values between 0 and 1 representing dominance per pitch class. Timbre is approximated as the decomposition of the spectro-temporal surface of the song according to 12 pre-computed basis functions.

To initially test our approach in simulation (a process described in detail in Section 3.5), we also needed real playlists to extract song transition data from. A good source of playlists needs to

Descriptors	Descriptor Indices
10th and 90th percentiles of tempo	1,2
average and variance of tempo	3,4
10th and 90th percentiles of loudness	5,6
average and variance of loudness	7,8
pitch dominance	9–20
variance of pitch dominance	21
average timbre weights	22–33
variance in timbre	34

Table 3.1: Descriptors used for song representation. Tempo data was based on beat durations. Thus the first descriptor is the 10th percentile of beat durations in the song. Loudness was straightforwardly obtained from amplitude. Pitch dominance weights each of the 12 possible pitch classes based on their presence in the song averaged over time. Timbre weights are the average weights of the 12 basis functions used by the Echo Nest analysis tool to capture the spectro-temporal landscape of the song.

be sufficiently rich and diverse, but also reflect real playlists "in the wild". In this chapter, we chose to use the Art of the Mix Archive, collected by Berenzweig et al. [27]. Berenzweig et al. gathered 29,000 playlists from The Art of the Mix (www.artofthemix.org), a repository and community center for playlist hobbyists. These playlists were (ostensibly) generated by real individual users, rather than a commercial radio DJ or a recommendation system, making this corpus particularly appealing for listener modeling.

3.4 DJ-MC

In this section we introduce DJ-MC, a novel reinforcement learning approach to a playlist-oriented, personalized music recommendation system. The DJ-MC agent architecture contains two major components: learning of the listener parameters (ϕ_s and ϕ_t) and planning a sequence of songs. The learning part is in itself divided into two parts - initialization and learning on the fly. Initialization is critical if we wish to engage listeners quickly without losing their interest before the agent has converged on a good enough model. Learning on the fly enables the system to continually improve until it converges on a reliable model for that listening session. In simulation, we assume the user is able to specify an initial list of songs that they like (this is similar to most initialization practices used by commercial music recommendation systems). However, in Section 3.6 we show this step can be replaced with random exploration, while still achieving a marked improvement at the exploitation stage.

The planning step enables the selection of the next appropriate song to play. As pointed out in Section 3.1, given the sheer scope of the learning problem, even after various abstraction steps, solving the MDP exactly is intractable. For this reason we must approximate the solution. From

a practical perspective, from any given state, the objective is to find a song that is "good enough" to play next. For this purpose we utilize Monte Carlo Tree Search.

In Sections 3.4.1 and 3.4.2 we describe the initialization steps taken by DJ-MC. In Section 3.4.3 we describe the core of the learning algorithm, which learns on the fly. in Section 3.4.4 we describe the planning step. The full agent pseudocode is provided in Algorithm 5.

3.4.1 Learning Initial Song Preferences

To initialize the listener's song model, DJ-MC polls the listener for his k_s favorite songs in the database and passes them as input to Algorithm 1. As a form of smoothing (or of maintaining a uniform prior), each element of $\phi_s(u)$ is initialized to $1/(k_s + \#bins)$, where $\#bins$ is the granularity of discretization of each song descriptor – in our case 10 (line 2). Then for each favorite song a, $\phi_s(u)$ is incremented by $1/(k + \#bins) \cdot \theta_s(a)$ (line 5). At the end of this process, the weights in $\phi_s(u)$ corresponding to each song descriptor sum to 1.

Algorithm 1 Initialize Song Preferences R_s

1: **Input:** Song corpus, $\mathcal{M}$
   Number of preferred songs to be provided by listener, k_s
2: initialize all coordinates of ϕ_s to $1/(k_s + \#bins)$
3: $preferredSet = \{a_1, \ldots, a_{k_s}\}$ *(chosen by the listener)*
4: **for** $i = 1$ **to** k_s **do**
5: $\quad \phi_s = \phi_s + \frac{1}{(k_s+1)} \cdot \theta_s(a_i)$
6: **end for**

3.4.2 Learning Initial Transition Preferences

In the second stage, the listener is queried for preferences regarding transitions, following the procedure in Algorithm 2. As in the case of initializing song preferences, the predicted value of a transition from bin i to bin j for each feature is initialized to $1/(k_t + \#bins)$ where k_t is the number of transitions queried and $\#bins$ is the number of feature transition bins – in our case 100 (line 2).

We wouldn't want to query transitions for too small a subset of preferred songs, because that won't necessarily reveal enough about the preferred transitions. For this reason we explore the preferences of the listener in a targeted fashion, by presenting them with different possible transitions that encapsulate the variety in the dataset, and directly asking which of a possible set of options the listener would prefer. On the other hand, we would also like to exclude regions in the search space where expected song rewards are low.

To accomplish both ends, DJ-MC first chooses a 50-% subset of the songs $\mathcal{M}^*$ of the song corpus $\mathcal{M}$ which, based on its song rewards model, obtains the highest song reward R_s (line 3). Then, DJ-MC queries transition preferences over this upper median of songs by eliciting user feedback. It does so by applying the δ-medoids algorithm, a novel method for representative

Algorithm 2 Initialize Transition Preferences R_t

1: **Input:** Song corpus $\mathcal{M}$
   Number of transitions to poll the listener, k_t
2: initialize all coordinates of ϕ_t to $1/(k_t + \#bins)$
3: Select upper median of $\mathcal{M}$, $\mathcal{M}^*$, based on R_s
4: δ = 10th percentile of all pairwise distances between songs in $\mathcal{M}$
5: representative set $\mathcal{C} = \delta$ -medoids ($\mathcal{M}^*$)
6: $song_0$ = choose a song randomly from $\mathcal{C}$
7: **for** $i = 1$ **to** k_t **do**
8: $\quad song_i \leftarrow$ *chosen by the listener from* $\mathcal{C}$
9: $\quad \phi_t = \phi_t + \frac{1}{(k_t+1)} \cdot \theta_t(song_{i-1}, song_i)$
10: **end for**

selection (line 5) [215]. This algorithm returns a compact but close-fitting subset of representatives such that no sample in the dataset is more than a parameter δ away from a representative, thus providing a diverse sample of the upper median of songs. δ is initialized to be the 10-th percentile of the distance histogram between all pairs of songs in the database (line 4). We denote the representative subset $\mathcal{C}$. To model transitions, DJ-MC chooses songs from $\mathcal{C}$, and queries the listener which song $a_i \in \mathcal{C}$ they would like to listen to next (line 8).[3] For modeling purposes, we assume the listener chooses the next song he would prefer by simulating the listening experience, including the non-deterministic history-dependent transition reward, and choosing the one with the maximal total reward. DJ-MC then proceeds to update the characteristics of this transition, by increasing the weight of transition features by $1/(k + \#bins)$ (line 9) , similarly to how it updated the model for song preferences (so again, the weights of each individual descriptor sum up to 1). The full details of the algorithm are described in Algorithm 2.

3.4.3 Learning on the Fly

After initialization, DJ-MC begins playing songs for the listener, requesting feedback, and updating ϕ_s and ϕ_t accordingly. For ease of use DJ-MC does not require separate ratings for songs and transitions. Rather, it can assign credit to each component individually from a single unified reward signal. It does so by computing the relative contributions of the song and transition rewards to the total reward as predicted by its model. This update procedure is presented in Algorithm 3.

Specifically, let r be the reward the user assigns after hearing song a in state s, and $\bar{r}$ be the average rewards assigned by this listener so far (line 4). We define $r_{incr} = log(\frac{r}{\bar{r}})$ (line 5). This factor determines both direction and magnitude for the update (negative if $r < \bar{r}$, positive otherwise, and greater the farther r is from average). Let $R_s(a_i)$ and $R_t(a_{i-1}, a_i)$ be the expected song and transition rewards yielded by our model, respectively. DJ-MC uses the proportions of these values to set weights for credit assignment (this is essentially a maximum likelihood estimate).

[3] If $\mathcal{C}$ is too large, it can be replaced at this step with a smaller subset, depending on the parameters of the system and the size of $\mathcal{C}$.

Algorithm 3 Model Update

1: **Input:** Song corpus, $\mathcal{M}$
   Planned playlist duration, K
2: **for** $i \in \{1, \ldots, K\}$ **do**
3: Use Algorithm 4 to select song a_i, obtaining reward r_i
4: let $\bar{r} = average(\{r_1, \ldots, r_{i-1}\})$
5: $r_{incr} = log(r_i/\bar{r})$
   weight update:
6: $w_s = \frac{R_s(a_i)}{R_s(a_i)+R_t(a_{i-1},a_i)}$
7: $w_t = \frac{R_t(a_{i-1},a_i)}{R_s(a_i)+R_t(a_{i-1},a_i)}$
8: $\phi_s = \frac{i}{i+1} \cdot \phi_s + \frac{1}{i+1} \cdot \theta_s \cdot w_s \cdot r_{incr}$
9: $\phi_t = \frac{i}{i+1} \cdot \phi_t + \frac{1}{i+1} \cdot \theta_t \cdot w_t \cdot r_{incr}$
10: Per $d \in$ descriptors, normalize ϕ_s^d, ϕ_t^d
   (where ϕ_x^d denotes coordinates in ϕ_x corresponding to 10-percentile bins of descriptor d)
11: **end for**

Concretely, we define the update weights for the song and transition to be
$w_s = \frac{R_s(a_i)}{R_s(a_i)+R_t(a_{i-1},a_i)}$ and
$w_t = \frac{R_t(a_{i-1},a_i)}{R_s(a_i)+R_t(a_{i-1},a_i)}$ respectively (lines 6-7).

Finally, the agent uses the credit assignment values determined at the previous step to partition the given reward between song and transition weights, and update their values (lines 8-9). Following this step, DJ-MC normalizes both the song and transition reward models so that the weights for each feature sum up to 1 (line 10). This update procedure as a whole can be perceived as a temporal-difference update with an attenuating learning rate, which balances how much the model "trusts" the previous history of observations compared to the newly obtained signal. It also guarantees convergence over time.

3.4.4 Planning

Equipped with the listener's learned song and transition utility functions R_s and R_t, which determine the MDP reward function $R(s, a) = R_s(a) + R_t(s, a)$, DJ-MC employs a tree-search heuristic for planning, similar to that used in [400]. As in the case of initializing the transition weights (Algorithm 2), DJ-MC chooses a subset of 50-percent of the songs in the database, which, based on R_s, obtain the highest song reward (line 2). At each point, it simulates a trajectory of future songs selected at random from this "high-yield" subset (lines 7-11). The DJ-MC architecture then uses R_s and R_t to calculate the expected payoff of the song trajectory (line 12). It repeats this process as many times as possible, finding the randomly generated trajectory which yields the highest expected payoff (lines 13-16). DJ-MC then selects the first song of this trajectory to be the next song played (line 19). It uses just the first song and not the whole sequence because as modeling noise accumulates, its estimates become farther off. Furthermore, as we discussed in Subsection

Algorithm 4 Plan via Tree Search

1: **Input:** Song corpus $\mathcal{M}$, planning horizon q
2: Select upper median of $\mathcal{M}$, $\mathcal{M}^*$, based on R_s
3: $BestTrajectory = null$
4: $HighestExpectedPayoff = -\infty$
5: **while** computational power not exhausted **do**
6: $trajectory = []$
7: **for** $1, \ldots, q$ **do**
8: $song \leftarrow$ selected randomly from $\mathcal{M}^*$
(avoiding repetitions)
9: optional:
$song\_type \leftarrow$ selected randomly from $song\_types(\mathcal{M}^*)$
(avoiding repetitions, $song\_types(\cdot)$ reduces the set to clusters)
10: add $song$ to $trajectory$
11: **end for**
12: expectedPayoffForTrajectory $= R_s(song_1) + \sum_{i=2}^{q} (R_t((song_1, \ldots, song_{i-1}), song_i) + R_s(song_i))$
13: **if** expectedPayoffForTrajectory $>$ HighestExpectedPayoff **then**
14: HighestExpectedPayoff = expectedPayoffForTrajectory
15: $BestTrajectory = trajectory$
16: **end if**
17: **end while**
18: optional: if planning over song types, replace $BestTrajectory[0]$ with concrete song.
19: return $BestTrajectory[0]$

5.3, DJ-MC actively adjusts ϕ_s and ϕ_t online based on user feedback using Algorithm 4. As a result, replanning at every step is advisable.

If the song space is too large or the search time is limited, it may be infeasible to sample trajectories starting with all possible songs. To mitigate this problem, DJ-MC exploits the structure of the song space by clustering songs according to song types (line 9).[4] It then plans over abstract song types rather than concrete songs, thus drastically reducing search complexity. Once finding a promising trajectory, DJ-MC selects a concrete representative from the first song type in the trajectory to play (line 18).

Combining initialization, learning on the fly, and planning, the full DJ-MC agent architecture is presented in Algorithm 5.

3.5 Evaluation in Simulation

Due to the time and difficulty of human testing, especially in listening sessions lasting hours, it is important to first validate DJ-MC in simulation. To this end, we tested DJ-MC on a large set of

[4] In principle, any clustering algorithm could work. For our experiments, we use the canonical k-means algorithm [236].

Algorithm 5 Full DJ-MC Architecture

Input: $\mathcal{M}$ - song corpus, K - planned playlist duration, k_s - number of steps for song preference initialization, k_t - the number of steps for transition preference initialization

Initialization:

Call Algorithm 1 with corpus $\mathcal{M}$ and parameter k_s to initialize song weights ϕ_s.

Call Algorithm 2 with corpus $\mathcal{M}$ and parameter k_t to initialize transition weights ϕ_t.

Planning and Model Update:

Run Algorithm 3 with corpus $\mathcal{M}$ and parameter K

(Algorithm 3 iteratively selects the next song to play by calling algorithm 4, and then updates R_s and R_t. This is repeated for K steps.)

listener models built using real playlists made by individuals and included in The Art of the Mix archive. For each experiment, we sample a 1000-song corpus from the Million Song Dataset.

One of the issues in analyzing the performance of DJ-MC was the nonexistence of suitable competing approaches to compare against. Possible alternatives are either commercial and proprietary, meaning their mechanics are unknown, or they do not fit the paradigm of online interaction with an unknown individual user. Still, we would like our evaluation to give convincing evidence that DJ-MC is capable of learning not only song preferences but also transition preferences to a reasonable degree, and that by taking transition into account DJ-MC is able to provide listeners with a significantly more enjoyable experience.

In order to measure the improvement offered by our agent, we compare DJ-MC against two alternative baselines: an agent that chooses songs randomly, and a greedy agent that always plays the song with the highest song reward, as determined by Algorithm 1. As discussed in the introduction, we expect that the greedy agent will do quite well since song reward is the primary factor for listeners. However we find that by learning preferences over transitions, DJ-MC yields a significant improvement over the greedy approach.

To represent different listener types, we generate 10 different playlist clusters by using k-means clustering on the playlists (represented as artist frequency vectors). We generate 1000 different listeners by first sampling a random cluster, second sampling 70% of the song transition pairs in that cluster, and third inputting this data to Algorithms 1 and 2 to train the listener's song and transition weights. For the experiments reported here we used a playlist length of 30 songs, a planning horizon of 10 songs ahead, a computational budget of 100 random trajectories for planning, a query size of 10 songs for song reward modeling and 10 songs for transition rewards. As shown in Figure 3.5, DJ-MC performs significantly better than the baselines, most noticeably in the beginning of the session. This result indicates that the learning architecture is robust enough to learn an unknown listener model which belongs to the same model class as the agent's internal model. However in the next subsection we show how the agent is able to learn an approximation of a listener model belonging to a different, more compless model class. Furthermore, in Section 3.6

we show the simulated results are in line with empirical results obtained from a user study with human participants.

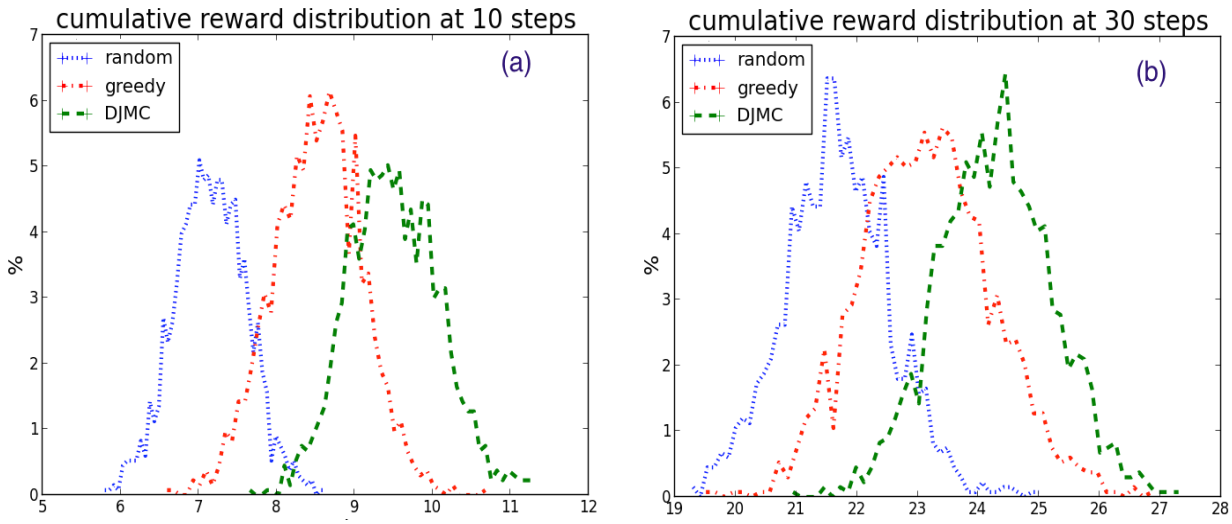

Figure 3.3: Cumulative reward histograms for playlists of length 10 (a) and 30 (b), with listeners based on real playlist data. The DJ-MC agent outperforms both random and greedy agents, particularly for the first 10 songs. Results are highly significant (p-value $<< 0.01$).

Performance with Complex Listener Models

A key design objective for DJ-MC is to faciliate generalization from limited online experiences. DJ-Mc achieves this objective through sparse song data representation and by modeling a listener's preferences with a simple, linear mapping of songs and transitions vectors. Specifically, DJ-MC represents listeners such that each feature contributes independently of other feature values to the enjoyment from a given songs or song transitions. While this representational choice aims to facilitate learning, actual listeners' preferences can be arbitrarily complex. Indeed, our understanding of human listening psychology suggests that there is a dependence across features which impacts how music is experienced and enjoyed [377]. For instance, it is common for listeners to prefer fast and loud music, or slow and quiet music, rather than slow, loud music or fast, quiet music [102].

Before we explore DJ-MCs performance with human listeners it is useful to understand whether DJ-MC's simple representation of listeners indeed contributes to generalization when learning and adapting to a listener's preferences on the fly. Thus far, listeners preferences were simulated with a linear mapping, namely having the same complexity used to model listeners internally by DJ-MC. We now explore how DJ-MC's adaptation to a listener performs when listener's enjoyment from songs and transitions reflect a fundamentally different (and more complex) mapping than that used internally by DJ-MC.

We aim to exolore two related questions. First, we aim to evaluate DJ-MC's robustness to encountering different and more complex listeners than those represented in DJ-MC's internal model. If indeed DJ-MC's simple internal representation of listeners is robust it will generalize well and yield increasingly better rewards over time when adapting to such listeners as well. Second, we aim to understand whether DJ-MC's simple represenation of listeners indeed generalizes better on the fly from its experiences than a DJ-MC variant whose internal representation of listeners reflects the true complexity of the listener's preferences. If an internal representation that matches the complexity of real users is beneficial, then DJ-MC's simple design not only does not faciliate learning on the fly, but undermines it. Together, these experiments provide inights into the robustness of DJ-MC's design and the contribution of DJ-MC's simple listener's represenation to generalization and adaption online, using limited experiences.

Representing Feature-Dependent Listeners

Let us first describe a listener model for the simulation which now reflects complex dependecies between song features. As before, we let the reward function R, corresponding to a listener, be factored as the sum of the listener's preference over *songs* and *song transitions*, namely $R(s,a) = R_s(a) + R_t(s,a)$. However, a listener's enjoyment now relies on the joint distributions of pairs of features. Thus, a listener implicitly assigns weights to bins corresponding to the joint distribution of each feature-pair, in a 10×10 grid partitioned by the 10-th percentiles of each feature. Transitions are similarly represented by the transitions between feature-pair bins. Thus, the indicator and weight vectors ϕ_s, θ_s are defined over feature pairs rather than independent features. Formally, $R_s(a) = \phi_s(u) \cdot \theta_s(a)$ and $E[R_t((a_1, \ldots, a_{t-1}), a_t)] = \sum_{i=1}^{t-1} \frac{1}{i^2} r_t(a_{t-i}, a_t)$, with $R_t(a_i, a_j) = \phi_t(u) \cdot \theta_t(a_i, a_j)$, as before.

3.5.1 Performance of DJ-MC with Feature Dependent Listeners

We simulate feature-dependent listeners by first drawing 30 pairs of features at random (ensuring that each of the original 34 features are selected at least once). Subsequently, $30 \cdot 10 \cdot 10$ song weights are assigned to the 100 bins per feature pair, reflecting a random draw of listener's preferences over songs. Similarly, $30 \cdot 10^2 \cdot 10^2$ transition weights are assigned to the 10000 bins per feature-pair transition, reflecting the corresponding listener's preferences over song transitions. The experiments follow the protocol described in Section 3.5. Both the greedy and the DJ-MC algorithms model song and transition rewards as an independent sum of feature rewards, while listeners exhibit more complex, feature-dependent preferences.

The reward histogram achieved by DJ-MC, the greedy algorithm, and the random benchmark are shown in Figures 3.4 and 3.5. The differences in performance between DJ-MC and random as well as DJ-MC and the greedy approach are statistically significant ($p << 0.01$). As shown, similar to the bahavior observed before, both DJ-MC and the greedy approaches produce higher

rewards than random, thus are able to learn and adapt to listeners using a different (and simpler) listener model. This result demonstrates that the design choice of a simple listener model yields a robust performance in the face of listeners with different preferences complexity than modeled internally by DJ-MC. This result will be later revisited through experiments with human listeners with potentially heterogeneous and certainly unknown preference patterns.

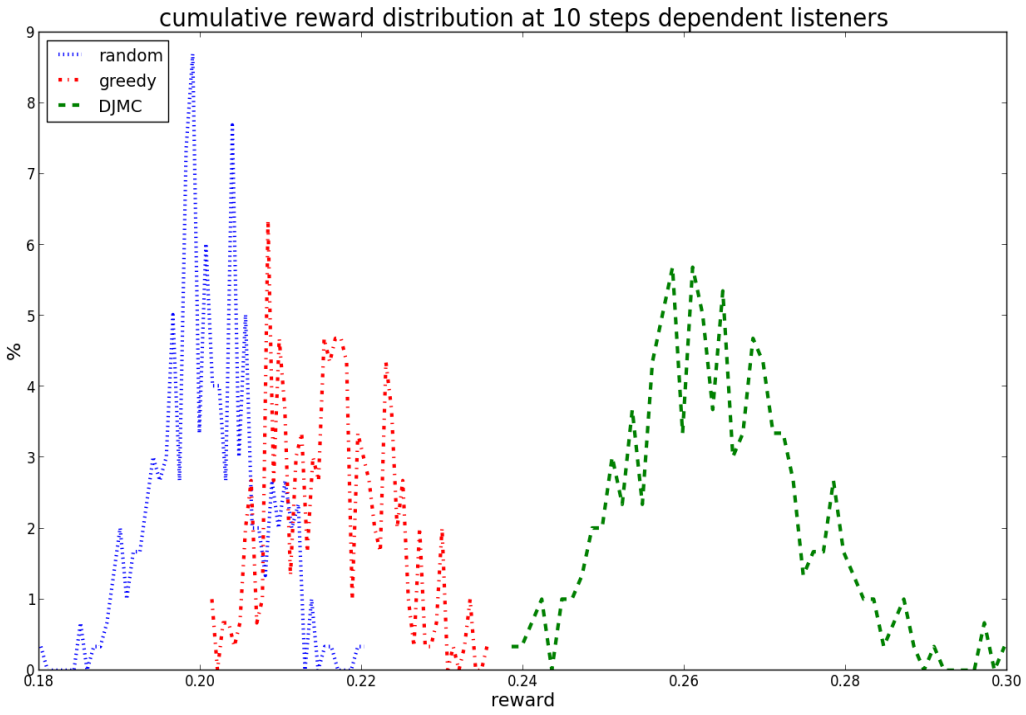

Figure 3.4: Cumulative reward histograms after 10 steps, with randomly drawn, feature-dependent listeners. The DJ-MC agent outperforms both random and greedy agents (p-value $<<$ 0.01).

3.5.2 A Feature-Dependent DJ-MC

The results in Section 3.5.1 demonstrate that DJ-MC is robust to having a different listener model than the actual listeners it adapts to. However, it is important to established whether generalization from limited experiences with listeners indeed benefits from having a simple listener representation relative to having an internal listener model that is more complex so as to match the true complexity of the listeners DJ-MC adapts to. While we will not be able to discern the true preference patterns of real human listeners, to explore this question, it is possible in a simulation to increase the complexity of DJ-MC's representation of listeners to match that of the complex, simulated listeners . Recall that DJ-MC's design was based on the notion that a simple internal listener representation would

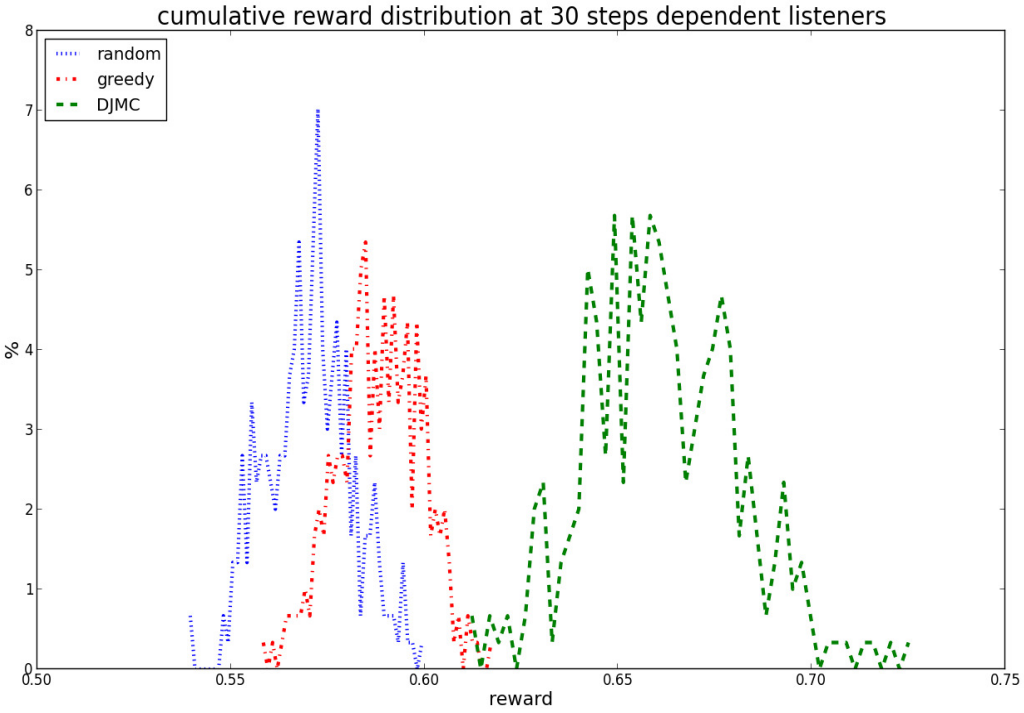

Figure 3.5: Cumulative reward histograms after 30 steps, with randomly drawn, feature-dependent listeners. The DJ-MC agent outperforms both random and greedy agents (p-value $<< 0.01$).

promote generalization and effective online adaptation; this is because the system cannot rely on abundant historical data, but learns exclusively from limited experiences with the listener.

To explore this question we consider a variant of agent that takes advantage of knowledge of pairwise feature dependencies in the reward function can outperform the standard DJ-MC algorithm proposed in this chapter. We present a variant of DJ-MC that represents listeners via feature pair dependencies. Specifically, listeners in the feature-dependent DJ-MC are represented by features that constitute the entire set of feature pairs. DJ-MC therefore learns $10 \cdot 10$ weights per feature pair and $10^2 \cdot 10^2$ weights per feature pair transition. Importantly, becuase it is unknown which k feature pairs are active for a given listener, DJ-MC learns the entire set of 561 feature pairs.

Figures 3.5.2 and 3.5.2 show the reward histograms for DJ-MC, the feature-dependent DJ-MC variant, Greedy, and the random approach. As shown, the original, feature-independent DJ-MC approach outperforms the feature-dependent DJ-MC. Both DJ-MC variants outperform the greedy and random alternatives. These results demonstrate that the complexity of DJ-Mc's internal listener model used for its adaptation to listeners, is an important contributor to DJ-MC's effective generalization and online adaption. In particular, the feature-dependent DJ-MC variant entails learning a complex model, and as shown this undermines adapation based on limited experiences

available in an online setting. The result strongly demonstrates that a simple listener represtation is key to faciliate learning and adaptation to listeners preferences. Furthermore, this result also suggest that DJ-MC's robust performance in the face of listeners with varying complexity can be attributed to DJ-MC's internal listener represetation. Even when a listener is arbitrarily complex, matching DJ-MC's internal listener representation to this complexity is not advisable. Instead, a simple representation, while not reflecting listener's true complexity, is crtical to faciliate effective learning and adaption in real time.

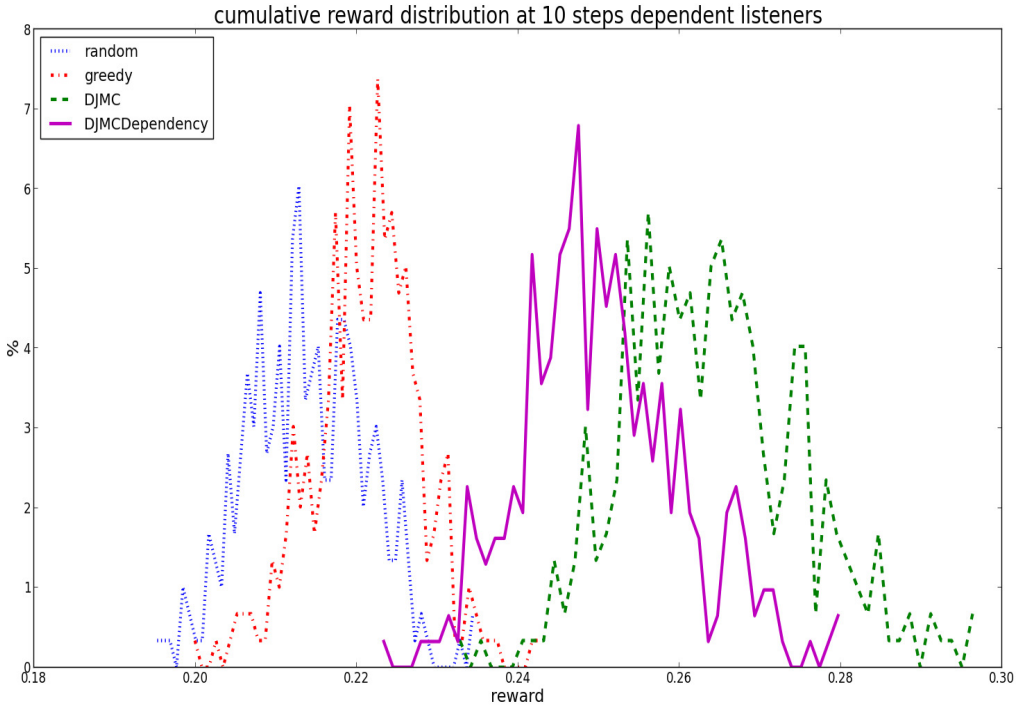

Figure 3.6: Cumulative reward histograms for playlists of length 10, with random feature-dependent listeners, comparing DJ-MC and the baselines to a DJ-MC variant that learns feature dependencies. The DJ-MC agent outperforms all the other agents. Results are highly significant (p-value $<<$ 0.01).

3.6 Evaluation on Human Listeners

DJ-MC's evaluation in simulation explored its robustness to different listeners and whether DJ-MC's internal listener model facilitates adaptation in an online setting or whether it would be beneficial to model listeners' true underlying complexity. To address these questions, we simulate synthetic listeners who are affected by song transitions. Let us now evaluate DJ-MC's performance

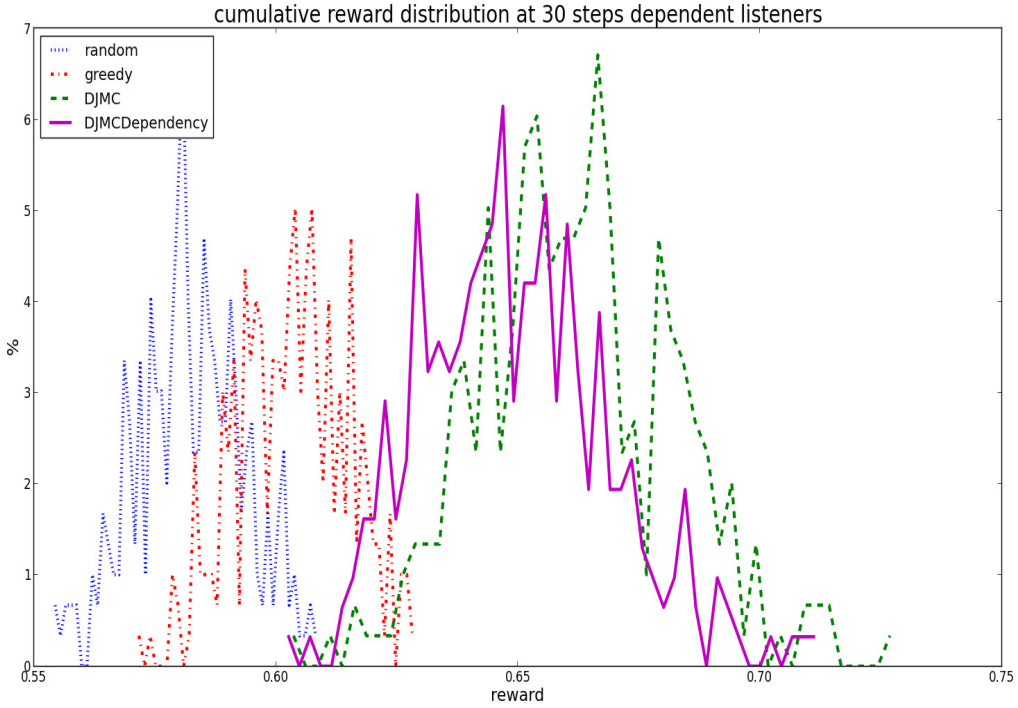

Figure 3.7: Cumulative reward histograms for playlists of length 30, with random feature-dependent listeners, comparing DJ-MC and the baselines to a DJ-MC variant that learns feature dependencies. The DJ-MC agent outperforms all the other agents. Results are highly significant (p-value $<<$ 0.01).

with human listeners, whose preferences are unknown, potentially heterogeneous and complex. We therefore aim to establish whether listeners experience is indeed affected at all by transitions: if human listeners experience is unaffected by transitions we expect listeners to attribute random rewards to different transitions. Furthermore, it would not be possible to derive patterns for transition preferences. By contrast, if listeners' experiences are affected by transitions and if DJ-MC is able to learn these preferences, DJ-MC will produce increasingly better rewards over time. Another goal of this study is to examine the tradeoff DJ-MC makes between playing songs the listener enjoys and choosing enjoyable transitions. The greedy approach learns and adapts to the listener's song preferences and is expected to yield high song rewards. If DJ-MC's attention to transition preferences improves the overall rewards this would also suggests that DJ-MC's design enables it to learn human's unknown and arbitrarily complex transition preferences.

3.6.1 Experimental Setup

We evaluated DJ-MC in a lab experiment with *47* graduate students at the McCombs School of Business at the University of Texas at Austin. The song corpus included songs from the Million Song Dataset, and that appeared in the Rolling Stone Magazine's list of 500 greatest albums of all time.[5] Each participant interacted with a playlist generator that was based on either DJ-MC or the greedy approach. The participants were divided into *2* groups: one group interacted with the greedy approach, and the other interacted with DJ-MC. To keep the duration of the experiment reasonable, each song was played for *60* seconds before transitioning (with a cross-fade) to the next song. After each song, participants were asked whether they liked or disliked the song as well as the transition to it, and provided this feedback via a graphic user interface (a thumbs up or down button). The feedback produced separate (albeit not necessarily independent) signal for song quality and song transition enjoyment, so that we can later draw insights about DJ-MC's contribution. However, as before, DJ-MC receives and learns from only a single, unified feedback of the listener's overall enjoyment, As before, DJ-MC then distributes the listener's feedback between song and song-transition reward based on its internal model of the listener. Rewards were calculated such that a "Like" signal, either for transition or song, corresponds to a +1 reward value, whereas a 0 reward is assigned for a "dislike". Recall that, in simulations, initialization was done by the listener selecting 10 songs they enjoy. As noted above, initialization can also be done by letting the learning agent to initially explore the space of songs and transitions uniformly at random. To make initialization easier for listeners in the experiment, DJ-MC performed random exploration for *25* songs, followed by onine learning and exploitation of the learned model over additional *25* songs.

Participants were divided into *2* groups - *24* interacted with the greedy approach and the remaining *23* interacted with DJ-MC. To estimate the distribution and perform hypothesis tests, we applied bootstrap resampling and estimated the empirical bootstrap distribution of the sample mean rewards produced by each approach [103, 124]. Following this bootstrap method (commonly used in statistical analysis for hypothesis testing with limited sample sizes), the empirical bootstrap distribution of the aggregate reward is constructed for each approach by resampling the participants of the corresponding approach with replacement. Because a bootstrap estimate's accuracy increases with more bootrstrap samples, the process was repeated $N = 10,000$ times to produce the bootstrap distribution of the sample mean.

Figure 3.8 presents the histogram of human listener rewards achieved by DJ-MC, greedy and the random approach. As shown, DJ-MC produces higher average reward than either greedy or random, and this result is statistically significant at $p << 0.05$. DJ-MC's performance with human listeners demonstrates that DJ-MC's architecture enables online adaptation to human listeners' preferences when preference may be arbitrarily complex relative to DJ-MCs internal representation of listeners. This result also confirms that human listeners' experiences are affected by the order in which songs are played, because learning and adapting to these preferenaces yields better

[5] http://www.rollingstone.com/music/lists/500-greatest-albums-of-all-time-20120531

experiences to listeners.

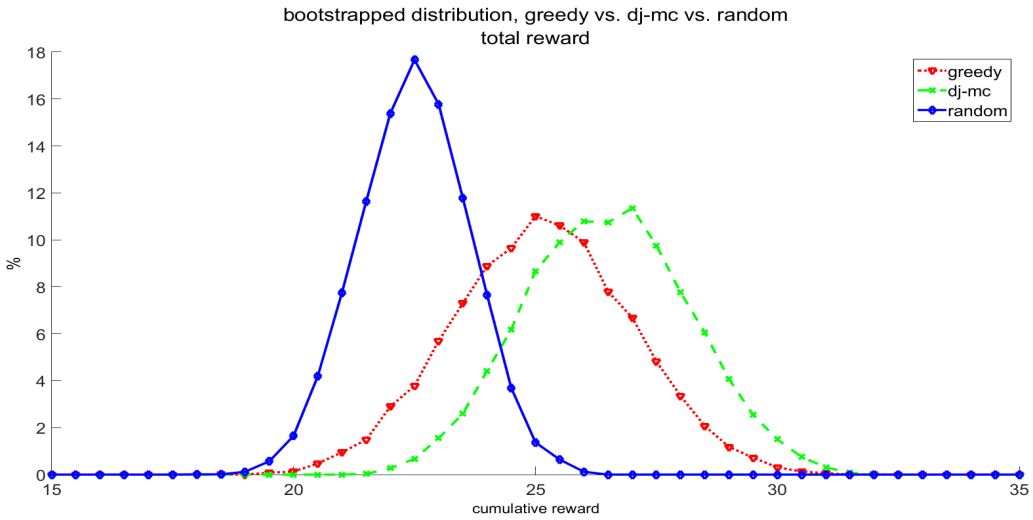

Figure 3.8: (a) Histogram of cumulative song rewards for the first 25 songs (b) Histogram of cumulative song rewards for the songs 25-50. (c) Histogram of cumulative transition rewards for the first 25 songs. (d) Histogram of cumulative transition rewards for the songs 25-50. Histograms computed via bootstrap resampling the original data 250,000 times.

Recall that while the greedy approach plays songs it estimates will derive the highest reward, irrespective of the ensuing transitions, DJ-MC may trade-off song reward to offer a more enjoyable sequence, or visa versa. It is interesting therefore to explore whether DJ-MC achieves its advanatge at the cost of lower song reward, or whether DJ-MC achieves higher song reward than does greedy, implying that DJ-MC's advantage cannot be attributed to its adaption to transition preferences. Towards that, we examine the transition rewards and song rewards obtained by each approach, seperately. Recall that during the first 25 episodes, both greedy and DJ-MC randomly explore the action space. For the first 25 episodes, Figures 3.9a and 3.9b show that the song and transition reward histograms; as expected, the two approaches yield comparable rewards at this phase. Figures 3.9c and 3.9d show the song and transition rewards during the exploitation stage, when the approaches differ in the songs and sequences they choose to play to listeners. As shown, both DJ-MC and greedy yield comparable song rewards. Thus, any differences in performance can be attributed to DJ-MC's modeling of listeners transition preferences. Importantly, The Figures presented in 3.9 demonstrate that DJ-MC achieves significantly higher transition reward than the greedy algorithm. This difference is statistically significant according to an unpaired t-test (at $p << 0.01$). [6]. The differences in transition rewards demonstrate that DJ-MC's learning and online adaptation to human listeners' transition preferences yields more enjoyable transitions and that this does not come at a cost of a lower song rewards. Note that, while inferior to DJ-MC's transition reward,

[6]Testing whether the difference in mean reward is greater than 0.25 is statistically significant as well.

the greedy approach's average transition reward also improves in the exploitation phase relative to the rewards in the initialization (random exploration) phase. It is possible therefore that listeners are more likely to enjoy a transition if they enjoy the songs involved. It is also possible that the set of songs a listener enjoys tends to be more homogeneous in character, thereby having the effect of reducing the "risk" of producing poor transitions. Importantly, the higher reward achieved by DJ-MC suggests that, in the case that transition between favorite songs are more enjoyable, then some transition between favorite songs are more enjoyable to listerens than others. Consequently, learning these transition preferences yield a bettter listening experience.

Overall, our results with human listeners simultaneously demonstrate that the existence of patterns in transition preferences which DJ-MC captures and uses for the selection of transitions implies that listeners indeed have preferences for transitions and that learning these patterns of preferences yields an overall better experience. Importantly, DJ-MC's design to promote generalization in an online setting allows effective learning and adaption to human listeners preferences, which are both unknown and can be arbitrarily complex relative to DJ-MC's representation of listeners. As demonstrated in our simulation results as well, DJ-MC simple representation of listeners and representation of songs and transitions facilitate DJ-MC's learning from limited online experiences.

3.7 Planning Extensions To DJ-MC

Thus far in the description of the DJ-MC architecture, I have have predominantly focused on the *learning* aspect of modeling user preferences, rather than the *planning* aspect of utilizing learned knowledge effectively to generate good sequences. However, given the complexity of the playlists space, more sophisticated planning approaches could prove extremely useful. In this section I address this gap by applying the adaptation of an advanced planning approach, Upper Confidence Bound in Trees (UCT) [185], to generate better song sequences. This extension further serves to satisfy Contribution 2 of this thesis, solidifying both the conceptual and practical value of a sequential decision making approach to music recommendation, and the utility of ideas belonging to the reinforcement learning literature in domains in which they have not been applied thus far. To further illustrate the usefulness of our approach, we also created a new variant of DJ-MC that looks at a different recommendation object, namely novelty search, and showed that our proposed approach was well suited for this setting as well.

3.7.1 Upper Confidence Bound in Trees (UCT)

In this section, I discuss how DJ-MC planning can be improved using the more sophisticated UCT algorithm. There are several approaches to solving an MDP optimally, assuming the transition

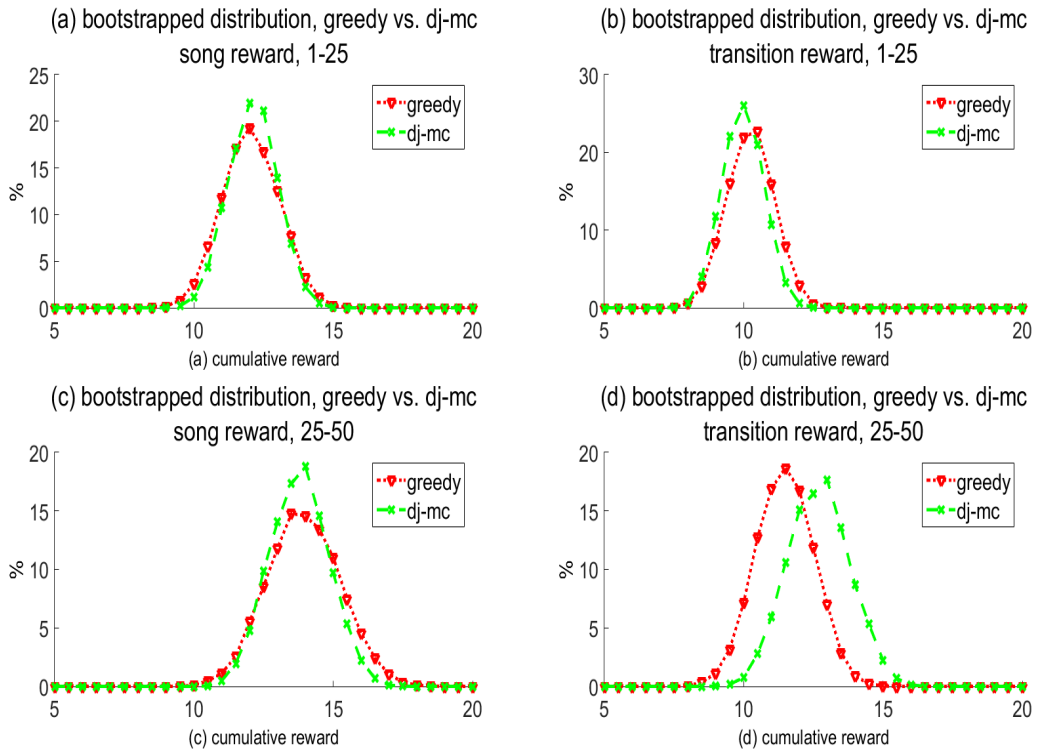

Figure 3.9: (a) Histogram of cumulative song rewards for the first 25 songs (b) Histogram of cumulative song rewards for the songs 25-50. (c) Histogram of cumulative transition rewards for the first 25 songs. (d) Histogram of cumulative transition rewards for the songs 25-50. Histograms computed via bootstrap resampling the original data 250,000 times.

function P and reward function R are known. For example, Value Iteration [374] solves discrete MDPs directly with dynamic programming. However, in many domains, including the playlist generation problem described in the previous sections, the state-action space is so large that optimally solving the MDP is infeasible. In these situations, it is necessary to use approximate solvers that restrict search to more relevant regions of the state action space. One family of such approaches is Monte Carlo Tree Search (MCTS). In this chapter, we focus on one variant of this approach called Upper Confidence Bound in Trees (UCT) [185].

In UCT, planning is performed by simulating a number of state-action trajectories from the current MDP state, i.e. Monte Carlo rollouts. In the playlist generation domain, the current MDP state reflects the songs that have already been played by the system. For each state-action pair encountered within this trajectory, UCT stores the number of visits for that pair as well as the long term expected reward of choosing that action at that state in a tree structure. Each node represents a state, with edges representing actions leading from one state to another.

Given information collected in prior simulations, UCT uses the UCB1 algorithm [17] for ac-

tion selection, allowing the algorithm to spend more time in areas of the state-action space that seem more promising. The UCB1 decision criterion is defined as $a = \text{argmax}_a \left(Q(s,a) + c_p \sqrt{\ln(n_s)/n_a} \right)$, where n_s is the number of visits to the state, n_a is the number of times action a was selected in previous simulations at this state, $Q(s,a)$ is the current expected long term reward for taking action a at this state, and c_p is tuned empirically to better balance exploration versus exploitation.

In this work, we apply a parametrized UCT variant called MaxMCTS(λ). MaxMCTS(λ) employs a more complex Q-value backpropagation strategy than that used in the original UCT algorithm. This variant was previously used for multi-robot coordination problems [178], and is studied extensively along with other variants by [179]. In MaxMCTS(λ), Q-values are estimated using an eligibility trace mechanism used in Peng's $Q(\lambda)$ reinforcement learning algorithm. This estimation process is summarized in Algorithm 6.

Algorithm 6 Eligibility trace backpropagation

1: **Input:** *trajectory* - Stack of ⟨state, action, reward⟩, populated during planning simulation.
2: $q \leftarrow 0$ {Backpropagated value}
3: **for** $\langle s, a, r \rangle = \textit{trajectory.pop}()$ **do**
4: $\quad q \leftarrow q + r$, $n_s \leftarrow n_s + 1$, $n_a \leftarrow n_a + 1$
5: $\quad Q(s,a) \leftarrow Q(s,a) + (q - Q(s,a))/n_a$
6: $\quad q \leftarrow (1-\lambda) \max_{a' | n_{a'} \neq 0} [Q(s,a')] + \lambda q$
7: **end for**

In Algorithm 6, q is a value backpropagated up the tree, and used to update Q-value estimates in Line 5. The key update rule for this backpropagation strategy is the update rule on line 6, which uses parameter λ to interpolate between the current backpropagated value and the maximum Q-value estimate at that state. Intuitively, when λ values are close to 0, even when exploratory actions are taken further down in the tree, the value of the action with the highest expected reward is propagated higher up in the tree. This technique minimizes the risk of exploratory actions taken further down the tree, but increases the likelihood of finding suboptimal policies. The value of λ must be selected empirically, and intermediate values between 0 and 1 can often provide significantly better performance in some domains.

3.8 UCT for Playlist Generation

As pointed out above, while the original DJ-MC architecture as described in Sections 3.2 - 3.6 puts a great deal of emphasis on effectively learning user preferences from limited information, when it comes to leveraging the learned model to select the next song, a relatively naive planning heuristic was employed. However, especially given the complex nature of generating playlists using a large song database, it stands to reason that a stronger heuristic, better suited for balancing the exploration-exploitation tradeoff with limited information, is a more appropriate choice. On the other hand, more sophisticated methods hold the risk of requiring more experience to be effective,

which may be a problem if only a limited number of simulations is possible.

Two main adaptations need to be made to make UCT-based approaches applicable in the playlist recommendation setting. First, as mentioned above, the more sophisticated parameterized backup strategy of MaxMCTS(λ) has to be used, since given the difficulty of the search problem pure Monte Carlo backups aren't likely to find good enough trajectories. Second, to make DJ-MC with MaxMCTS(λ) applicable to huge song corpora, we introduce hierarchy into the song selection step. One of the key determining factors in the efficiency of MCTS methods is the branching factor induced by the domain at each node. The branching factor is determined by the number of available actions at each node of the UCT tree. Since by default the set of actions available at each state of the music playlist MDP is the entire set of songs M, the default branching factor for this domain is prohibitively high for even moderately sized music databases. To mitigate this issue, we use the structure of the song space to cluster songs into subsets. Each subset represents an abstract song type. Then, we alternate between choosing song types and choosing specific songs at each step of the trajectory, dramatically reducing the branching factor (the lowest branching factor in expectation is achieved when the number of clusters is $\sqrt{M}$).

Algorithm 7 MaxMCTS(λ) for music applications starting at playlist s

1: *input: current playlist s, song corpus* $\mathcal{M}$
2: Cluster $\mathcal{M}$ to obtain song types C and mapping to concrete songs $S_{\mathcal{M}}(C)$
3: *rootNode* $\leftarrow$ *initNode*(s) {Root node represents the current playlist state}
4: **for** *sim* $\in \{1, \ldots,$ *numSimulations*$\}$ **do**
5: *node* $\leftarrow$ *rootNode*
6: *trajectory* $\leftarrow$ **new** Stack
7: **while** *notTerminal*(*node*) **do**
8: **if** *node.parent* $\in \mathcal{M}$ **then** $a \in$ *Song Types* {I}f last song in the trajectory is instantiated, plan over abstract songs.
9: **else** $a \in S_{\mathcal{M}}$(*node.parent*)
10: **if** $node.n_s = 0$ **then** $a \leftarrow$ *default song selection*
11: **else** $a \leftarrow$ *selectNextActionWithUCB1*(*node*)
12: $\langle ns, reward \rangle \leftarrow$ *simulate*($node, a$)
13: *nextNode* $\leftarrow$ *getOrInitNode*($node, a, ns$ {N}ext node represents the playlist after selecting a new song.
14: *trajectory.push*(*node*, *a*, *reward*)
15: *node* $\leftarrow$ *nextNode*
16: **end while**
17: BACKPROPAGATE(*trajectory*) using Algorithm 6
18: **end for**

The pseudocode for MaxMCTS(λ) adapted to the playlist generation domain is presented in Algorithm 7. Line 2 preprocesses the song set by clustering the song corpus to obtain abstract song types and a mapping from each song type to a set of concrete songs that comprise that type. Clustering is done via the canonical K-Means algorithm [236]. Then line 3 initializes the root of

the search tree to be the current playlist state. The main loop in lines 4-18 generates one MCTS simulation. Lines 7-12 descend down the tree using either the UCB1 criterion for action selection or a default song exploration policy if not all actions have been tried at least once. Lines 9-10 decide whether we're currently selecting a song type or a concrete song of the song type specified. Line 13 runs a simulation from the new song selected to obtain an estimated user reward this song choice will accrue. Once the rollout is complete line 17 backpropagates the overall reward up the search tree, using Algorithm 6. For clarity, in the rest of this chapter, we will refer to the DJ-MC architecture proposed in Sections 3.2 - 3.6 as "standard DJ-MC" and to the modifications proposed in Sections 3.7 - 3.10 simply as "MaxMCTS(λ)".

3.9 Planning for Personalization

In this section we empirically evaluate the benefit of adapting MaxMCTS(λ) to the DJ-MC framework, which aims to maximize playlist personalization. We compare the performance of the proposed algorithm, DJ-MC + MaxMCTS, with that of "vanilla" DJ-MC, measuring user reward over 30-song sequences. We also compare it to the benchmarks DJ-MC was originally compared against: a greedy heuristic which selects the favorite song not played yet irrespective of sequence, thus mimicking a more traditional music recommendation algorithm, and a random baseline. the results are presented in Figure 3.10.

Results indicated that using MaxMCTS(λ) with $\lambda = 0.5$ statistically significantly outperforms the standard DJ-MC, and that with all chosen λ values MaxMCTS(λ) did as well as or better than the standard planning technique used in DJ-MC.

3.10 Planning for Diversity

DJ-MC is an interesting and robust framework for learning playlist preferences and generating playlists efficiently on the fly. However, it is not the only playlist recommendation framework where planning is useful. Indeed, any playlist recommendation framework which takes transitions into account should benefit from the usage of advanced planning techniques such as MaxMCTS(λ). To illustrate this point, we study the application of MaxMCTS(λ) with various λ values in a different playlist recommendation setting, with different transition mechanics and a different reward function - diversity (or novelty) based playlist generation.

The idea of algorithmic novelty search in playlist generation has been proposed before [409, 207, 229, 378] and it seems intuitive - even once we gain some knowledge user preferences, they wouldn't want to listen to very similar songs one after another in sequence, leading to a potentially tedious experience.

To this end, we propose a novel diversity generation framework. Given a song database $\mathcal{M}$ and some similarity metric D between songs, we are tasked with finding a sequence of songs

the listener enjoys, but penalize for the amount of similarity between each song chosen and the songs which preceded it. Similar to the temporal discounting approach adopted by [218], we can assume the penalty for similarity across songs in the sequence decays as songs progress. Formally, in the novelty detection setting, we assume the same general playlist problem MDP as in DJ-MC, but propose an alternative approach for modeling R. Beyond its usefulness as a valid playlist recommendation objective, this variation on the DJ-MC reward model serves to illustrate the robustness and generalizability of the approach presented in this chapter, as it can be utilized in more than just one specific formulation of the playlist recommendation problem.

Concretely, instead of the decomposition proposed in Eq. 3.1 (see p. 15), we propose an alternative formulation, as presented in Eq. 3.2. Assuming a utility function for individual songs R_s and a similarity measure between songs D, the reward function representing a listener u for a song sequence $Seq = \{s_0, \ldots, s_t\}$ is modeled as:

$$R^u_{novelty}(Seq) = \sum_{i=0}^{t} (R_s(s_i) + \sum_{j=0}^{i} \frac{1}{i-j} D(s_i, s_j) \tag{3.2}$$

Given this setting, and assuming the user's individual song preferences are known, the planning problem - generating a good sequence of songs - is purely a combinatorial search problem.

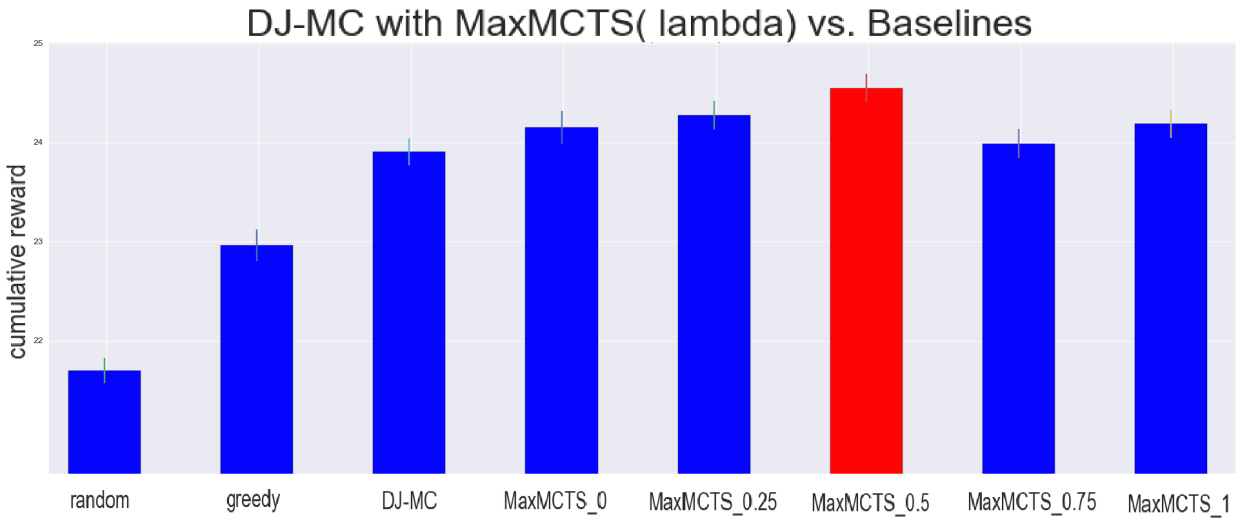

Figure 3.10: Average reward for 30-song sequences, comparing standard DJ-MC to DJ-MC with MaxMCTS(λ) planning using varying values of λ, a greedy system which attempts to maximize song enjoyment regardless of sequence, and a random sequence generator. Results are obtained over 30 repetitions with a corpus size of 5000 and 70 song types (leading to an average branching factor of 70). Songs are randomly taken from the Million Song Dataset [33]. Best performing algorithm marked red. Monte-Carlo approaches ran 5000 simulations.

However, it is intractable for even moderately sized song sets. Indeed, it can be shown to be NP-hard via a reduction from the weighted max-clique problem [301]. However, using MaxMCTS(λ) with the same two-stage song type abstraction suggested in Section 3.8 lends itself directly to this setting as well.

As in the DJ-MC extension case, we wish to empirically test MaxMCTS(λ) in the novelty maximization setting described above. We choose an experimental setting similar to that employed in the previous section, but in order to isolate the planning aspect, we assume the individual song preferences are roughly known - the listener provides a list of 100 liked songs and preferences are inferred from these songs based on the similarity function D which is also assumed to be known. For the purpose of this section, D is assumed to be the Euclidean distance between the 34-dimensional song representation vectors used by DJ-MC (this is similar in spirit to [229]), i.e. for two songs s_1, s_2, $D(s_1, s_2) = \sqrt{\sum_{i=1}^{34} (s_{1i} - s_{2i})^2}$.

The results in the diversity-based playlist recommendation domain are provided in Figure 3.11.

As evident from the results, in this setting, MaxMCTS(λ) with $\lambda = 0$ significantly outperforms the other approaches. We note that the relative improvement here is higher than that observed for the DJ-MC setting, but unlike that case, this result is sensitive to an apt choice of

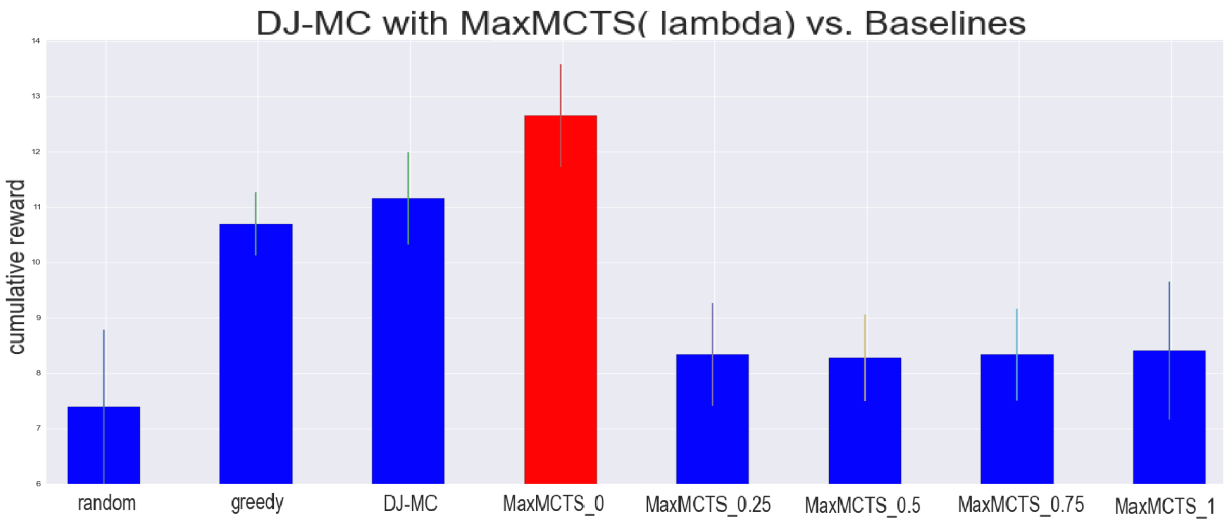

Figure 3.11: Average reward for 20-song long sequences, comparing a naive Monte-Carlo search approach to MaxMCTS(λ)planning using varying values of λ, a greedy system which attempts to maximize song enjoyment regardless of sequence, and a random sequence generator. Results are obtained over 30 repetitions with a corpus size of 1000 and 30 song types (leading to an average branching factor of 30). Songs are randomly taken from the Million Song Dataset. Best performing algorithm marked red. Monte-Carlo approaches ran 5000 simulations.

λ. It can nonetheless be seen as a positive result, especially given that [179] suggest a straightforward approach for roughly tuning λ. This outcome yet again illustrates how RL applied to playlist recommendation can improve performance compared to our baselines. Looking comparatively at the two different playlist generation settings, the novelty detection setting induces a considerably sparser search problem. This provides a relative advantage to planning methods which tend to retain relatively good observed trajectories, which is effectively what MaxMCTS($\lambda = 0$) does.

3.11 Summary

To recap, in this section I described our work on a sequential decision making approach for playlist recommendation. The proposed framework, named DJ-MC, formulates music recommendation as a Markov Decision Process, in which songs played at timesteps $1 \dots t$ affect the listener's experience of songs played at time $t + 1$. The system learns a model not only of the listener's individual song preferences, but also of the listener's preference over song transitions, and uses that knowledge to adaptively plan song playlists while updating its models online based on listener feedback. I further discuss various extensions to the core framework which are meant to more effectively plan playlist sequences, a potentially intractably complex problem given the size of playlist space. These extensions are useful both to help make the proposed framework more broadly applicable and to further demonstrate the value in a music recommendation domain as a testbed for reinforcement learning research, and to highlight the utility of ideas from the sequential decision making research in domains in which they haven't been yet applied. This work constitutes the core of Contribution 2 of this thesis.

This chapter focused on reinforcement learning as a framework for adapting individualized playlists on the fly in a single session. However, reinforcement learning can play a significant role in content recommendation over a much longer timeframe. In the next chapter I will describe the work done towards the completion of Contribution 5, i.e. providing algorithms for the adaptation of recommendation models over time.

4. Algorithms for Tracking Changes In Preference Distributions

As described in Section 1.1, one of the core problems in music recommendation over time, and in content recommendation in general, is that the distributional properties of music, and people's musical tastes, change over time in ways that are nontrivial to track and predict. This informational challenge is a special case of concept drift – the change, either abrupt or gradual, in the underlying structure of data. Concept drift is a common and fundamental problem in machine learning in general, so any solutions designed to combat drift in content recommendation, even ones specific for that setting, are broadly applicable to ML research in general. In this chapter I focus on the application of reinforcement learning approaches to handle concept drift, specifically through model retraining, both in its general context and directly with respect to tracking people's temporal listening habits as reflected by a real world dataset. This chapter fulfills Contribution 5 of this thesis, as well as 6, highlighting how an approach useful for a music related dataset has applications in other domains as well.

As automation grows, more and more industry control systems around us make decisions autonomously, ranging from image understanding [351] to movie recommendation systems [122] and network and service virtualization [58]. Underneath their hoods, many of these systems rely on models. These models can be descriptive (capturing the properties of data) or predictive (using known data to predict other, latent properties). However, such systems are often susceptible to the nonstationary, ever-changing dynamics of data in the real world. In recommendation systems, tastes and fashion change. In climate prediction, the properties of the environment change constantly. These changes, either gradual or abrupt, are commonly referred to as *concept drift*. Such shifts in the feature distribution and underlying label correspondence constitute a significant challenge to learning systems. In the face of concept drift, we consider the problem of how to generically and adaptively adjust models to mitigate the risks of drift. We refer to this challenge as *model retraining*, or model management. In this chapter I propose a novel *reinforcement learning* (RL) approach, framing the model retraining problem as a sequential decision making task, and harnessing ideas from the RL literature to learn a robust policy for model update.

The issue of concept drift has been the focus of many works in the past 20 years [417, 395, 436, 117]. Widmer and Kubat proposed a framework for continually deciding which samples to add and which to throw away (or "forget") as new data comes in [417]. Klinkenberg and Joachims presented a windowed support vector machine formulation meant to countermand drift [183], whereas Gamma et al. started a research thread of detecting drift by considering the error rate over time (I refer to this technique as DDetM in this chapter [7])[116].

[7] In the literature this method is typically abbreviated as "DDM", but to avoid confusion with the Drift-Diffusion Model used in Chapter 5, which is also abbreviated as DDM, I use the abbreviation DDetM for the Gamma et al.

More current examples include the work of Brzezinski and Stefanowski, who actively update an ensemble of classifiers weighted by their current accuracy, and combine them using Hoeffding trees [47], and HDDM, a method proposed by Frias-Blanco et al., that uses Hoeffding bounds to identify whether drift has occurred [114]. In another related work, Minku and Yao proposed a diversity based approach for adapting an ensemble to a drifting data stream[265]. These works are also connected to the large subfield of online learning [182]. The model retraining problem is also related to the notion of continual, or lifelong learning, both in the context of general machine learning [341], and in RL [361, 334]. However, lifelong learning is not the same as drift adaptation, since in the case of learning under drift, the learner is engaged in the same ongoing task, whereas in lifelong learning, the learner is expected to adapt to new tasks presented sequentially. Both lifelong learning, and our proposed task of autonomous model management can also be perceived as part of the transfer learning literature, a rich problem domain studied extensively both in the context of RL [379], and machine learning in general [391, 295].

A key difference between these methods and the research described in this chapter is that I do not focus on the specifics of the underlying models or even the data itself. Instead, we propose an RL meta-learner that decides when and how the model should be updated, thus enabling the models themselves to be simple and generic. Previous methods for drift detection such as DDetM and HDDM (which we compare against) focus on the error rate, which contains less information than the core distributional properties of the data that our method uses. In addition, rather than trying to adaptively find the optimal decision threshold given current information, methods such as DDetM employs hard-coded parameters, which are harder to tune appropriately.

Therefore, in fulfillment of Contribution 5, I adopt a sequential decision-making approach towards robust model management for content recommendation. According to this approach, rather than incorporate built-in drift resistance into the learned models, an agent can adaptively learn a meta-policy of when models should be updated. The core insight is that the problem of deciding when and how to retrain models can be suitably formulated as a Markov Decision Process. In such a formulation, the proposed system will be an agent, which, given the current model, observations of new data, and knowledge of past data, needs to routinely decide whether to update its model, and in what fashion. Incoming data modeled by the system can be divided into batches of varying size. For each batch, the system needs to decide how to best process the data. In the process, the agent needs to balance performance and cost (as they are determined for a given domain).

In Section 4.1, I formally define the model management (or model retraining) problem as a sequential decision-making task. In Section 4.2, I propose a formal approach for learning a policy for the model retraining Markov Decision Process via Approximate Value Iteration (AVI). In Sections 4.4 - 4.5 I consider the application of this approach in the context of models that track data distributions. Subsequently, in Section 4.6 I consider the harder problem of prediction models - models that do not simply reduce data to its sufficient statistics, but predict the value of hidden

algorithm instead.

variables based on this data. Finally in Section 4.7 I present a more data-efficient variation on the learning approach presented in the paper, that makes it possible to apply the proposed approach in higher-dimensional data domains. This chapter is based on published work carried out jointly with Eric Zavesky [224].

4.1 Model Retraining as a Markov Decision Process

Updating the current model of a given system affects not only the ability to act upon the data currently observed, but data observed in the future as well. Update a model too quickly and you may keep getting sidetracked by outlier occurrences, or waste resources on needlessly retraining too frequently. Wait too long to update, and the performance of your system might deteriorate drastically. Given this property, it makes sense to frame autonomous model management as a sequential decision-making task. As such, this problem is suitably formulated as a Markov Decision Process (MDP) [373]. In this formulation, the system is an agent, which, given the current model, observations of new data, and knowledge of past data, needs to routinely decide whether to update its model, and in what fashion.

Let us first consider a concrete example. Suppose you are managing a recommender system with a learned model that maps individual profiles to song recommendations. Song requests are received from multiple individuals over time. For quality assurance purposes, requests are grouped by hours. To decide whether the model is still useful, every k hours, the system aggregates all the songs played in a predetermined time window (say, a single hour), along with people's rating of the songs recommended to them (such that the aggregated information is supervised). Given this supervised sample of the data, how can the system decide best whether its model is up to date or not, and if not, how it should be updated?

Accordingly, in our formulation, incoming data modeled by the system can be divided into batches of varying size (most typically, representing some time window e.g. days or hours). For each batch, the system needs to decide how to best process the data. In the process, the agent needs to balance performance and cost (as they are determined for a given domain). We assume that it is infeasible for an agent to observe the entire batch before making a decision on how to best adapt the model. For this reason, the agent relies on subsampling the data before making a decision. We assume this subsample is unbiased. After observing the subsample, the agent then decides how to best update the model prior to handling the entire set of samples in the batch. This process is repeated in the next timestep indefinitely.

In the next two subsections, we first formally recap what a Markov Decision Process is, then proceed to describe the model retraining problem as an MDP.

4.1.1 Markov Decision Processes

As a reminder to the reader, an MDP M is represented as $\langle S, A, P, R\rangle$ where S is the set of states an agent can be in, A is the set of actions that the agent can take at a state, P is the transition function that gives the transition probability of reaching a particular next state s' from state s after taking action a ($P : S \times A \times S \to \mathbb{R}$; $\sum_{s'} P(s, a, s') = 1$), and R is the reward received given a transition ($R : S \times A \times S \to \mathbb{R}$). In continuing (non-episodic) MDPs it is often customary to specify a discount factor $\gamma \in [0, 1]$, which specifies the uncertainty (and therefore diminishing importance) of future rewards compared to rewards observed right now. To perform optimally in a task that an MDP represents, an agent must find a policy $\pi^* : S \to A$ such that executing action $\pi^*(s)$ from any given state s would yield the highest expected sum of rewards over the sequence of states and actions $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$. This is traditionally referred to as "solving" an MDP.

4.1.2 Formulation of the Model Retraining Problem

Formulated as a Markov Decision Process, the general structure of the model retraining problem is the same, regardless of domain or model purpose. In this subsection I describe how model retraining reduces to a Markov Decision Process.

The state is defined as a tuple $\langle M_t, obs_t\rangle$, with M_t the current model and obs_t the new observed data at time t. Data at time t contains multiple instances, and we refer to it as a *batch*. Batch sizes may vary from one timestep to the next. Furthermore, in this paper we assume not all batch data obs_t can be practically observed in real time to make a decision. Instead, some realistically observable and unbiased subsample of it, $\widehat{obs}_t$, is used by the agent, making the MDP partially observable. As $|\widehat{obs}_t| \to |obs_t|$, the properties of the sampled data converge to the exact values of the true world data at time t and the MDP becomes fully observable. Therefore, the state space S is defined as the Cartesian product of the model and observation spaces.

At each time step, the agent has the choice of either *retraining* a model from scratch based on the new data, *adapting* (or transfering) the current model to the new data, or leaving the current model as is. This defines an action space of three possible actions, $a \in \{\textsc{retrain}, \textsc{adapt}, \textsc{keep}\}$. For convenience of notation, we define these actions as functions of the current state, $a : M \times OBS \to M, a_t(\langle M_t, obs_t\rangle) \to M_{t+1}$, where M is the model space and OBS is the observation space.

S and A induce a stochastic transition function P. Specifically, we can use the following shorthand notation:

$$P(\langle M_t, obs_t\rangle, a_t) = \langle a_t(\langle M_t, obs_t\rangle), obs_{t+1}\rangle$$

Note that stochasticity originates from the fact that we do not know what observations the agent will observe next, nor do we know how well the data sample corresponds to the actual data the model needs to process at time t (though we assume the sample is unbiased).

As a reward function, we factor in the utility of correctly handling observed data vs. the cost of each action. A predefined utility *acc_reward* is associated with every single observation $o \in obs$

that is accurately modeled at time t.

Furthermore, there is an inherent cost associated with each action. While keeping an existing model costs nothing, adapting and retraining are both associated with a computational penalty per processed sample. The resultant reward function takes the form of $r_t = |O_{modeled}| \cdot acc\_reward - action\_cost$, where $O_{modeled}$ is the set of all samples in the batch correctly handled by the model at time t, where the specific notion of "correctly handled" changes from one domain to the next. Note that this formulation satisfies the Markov property, as the reward and transition functions are entirely dependent on the current state and the action taken.

The cost of each action represents the computational cost of training a model, as well as the cost of collecting and annotating data. For instance, assume, in our music recommender system example previously discussed, that while the sample obtained every k hours is sufficient to determine whether the model is still useful, it is not enough to train a sufficiently reliable model for future timesteps, and therefore additional samples need to be obtained. In the case of transfer, fewer samples are needed, and the computational cost of updating the model is lower than in the case that a new model is trained from scratch. There is also the opportunity cost of potential system downtime while models are being updated. In such a case, it is less expensive to train a new model from scratch during a "slow" timestep when there aren't that many observations to handle, compared to a very busy timestep when the number of handled observations is very high. The specifics of such cost considerations are determined specifically for each environment, but the framework is sufficiently flexible to represent a wide range of such considerations.

To complete the MDP formulation, we also specify a discount factor $\gamma \in [0, 1]$, a constant which reflects how future costs and utilities should count towards present estimates. Given that the agent maximizes the sum of discounted rewards $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$, setting γ to 0 only weights the immediate reward at each timestep, whereas setting γ to 1 gives equal weight to rewards obtained now or at any time in the future.

In the following section, we discuss how this formulation lends itself to a reinforcement-learning based approach to finding an optimal policy for model management that is domain agnostic in structure.

4.2 Learning a Policy through Approximate Value Iteration

Once the model retraining problem is formally defined as an MDP, a suitable policy needs to be learned for this MDP. As mentioned in Section 4.1.1, in certain cases the MDP can be explicitly "solved", resulting in the optimal policy. If the state and action spaces are finite and sufficiently small, solving the MDP can be done through a dynamic programming procedure called *value iteration* [373]. However, in the case of continuous state spaces this option is no longer feasible. A variety of approximation strategies exist to mitigate this problem. Consider for instance the simplest approach of partitioning a continuous state space via a grid, then applying value iteration

on the discretized state space (now rendered discrete and finite). As further discussed in Section 4.3, if the state space is Lipschitz-continuous [60], the error induced by this approximation can be bounded, and converges to 0 as the grid becomes tighter.

Let us assume our model can be parameterized as $M = \langle m_1, \ldots, m_k \rangle$. Given a reasonable model for observation distribution, it too can be parameterized, either by its sufficient statistics or using a nonparametric representation $obs_t = \langle d_1, \ldots, d_l \rangle$ (examples for such a representation include the weights of an artificial neural net or the coefficients of a regressor). This parameterization lends itself to a straightforward value iteration procedure based on a discretized representation of the parameter space.

For simplicity of notation, let us consider the following shorthand $[M - C, M + C]_x$ as representing a grid of width $2C$ around current model parameters M with stepsize x. For instance, for a simple two-dimensional model $M = \langle m_1, m_2 \rangle$, $[M - C, M + C]_x = M_1 \times M_2 = \{ (a, b) \mid a \in M_1 \text{ and } b \in M_2 \}$, where $C = \langle c_1, c_2 \rangle$, $M_1 = \{m_1 - c_1, m_1 - c_1 + x, m_1 - c_1 + 2x, \ldots, m_1 + c_1 - x, m_1 + c_1\}$ and $M_2 = \{m_2 - c_2, m_2 - c_2 + x, m_2 - c_2 + 2x, \ldots, m_2 + c_2 - x, m_2 + c_2\}$.

By discretizing both the model and observation spaces we have produced a discrete (and therefore finite) representation of the state space as the Cartesian product of $\mathcal{M} \times \text{OBS}$ where $\mathcal{M} = [M - C_m, M + C_m]_{x_m}$, and $\text{OBS} = [\text{obs} - C_{\text{obs}}, \text{obs} + C_{\text{obs}}]_{x_{\text{obs}}}$. This finite state representation enables us to perform the dynamic programming procedure of value iteration. Intuitively, the result of applying this procedure is the answer to the question of "if my current model parameters are M and the observation data is drawn from obs, should I RETRAIN, ADAPT or KEEP?". Lastly we note that the expected reward is given by $\int P_s(x) \cdot M_s(x) dx$ where $M_s(x)$ specifies whether in state s the model M (which is part of the state) correctly services request x. An intuitive strategy for deciding on the discretization range is to set C values to be a factor of the estimated observation covariance Σ. For observation distribution parameters, under certain smoothness assumptions, modeling the deviation range as a multiplier of the variance manifested in the data makes it very likely that your learned policy will be well-defined for the next batch of observations.

Intuitively, the policy learned by the approximate value iteration (AVI) process, given a current model and an estimate of the data distribution, specifies deviation ranges in which keeping the current model is best, deviation ranges where it is best to do model transfer, and ranges where it is best to learn a new model entirely. The challenge (and power) of this approach lies in finding the near-optimal cutoffs for these decisions at each timestep in an adaptive fashion. As we later show, relying on a fixed rule of thumb to accomplish the same does worse than this value iteration approach. In the following subsection we briefly discuss how far off from optimal this policy can be.

4.3 Theoretical Intuition

If we were able to use the unapproximated value iteration procedure (for instance, in a case where the state and action spaces are sufficiently small), and to fully observe the data at each step, the resulting policy would be optimal as a guaranteed property of the value iteration algorithm. However, given that both the estimate and the approximation are imperfect, error in the resultant policy comes from two sources: (1) the sampling error in the estimate of the current distribution; and (2) the state-action approximation error. Item (1) is manageable due to the fact that as $|\widehat{obs}|$ increases, the statistical estimates converge to the true values of OBS. Furthermore, it is reasonable to assume that in many cases $\widehat{obs}$ can be an unbiased sample, meaning the expectation over the bias is 0 [63]. As for Item (2), it has been shown that if the approximated space is a non-expansion then the approximation error with respect to the optimal policy is bound as well [125]. Specifically, if the space is Lipschitz-continuous, which is a reasonable assumption for many data domains, including the ones we examine in this paper, then the error can be bound by $\frac{h}{1-\gamma}(K_1 + \gamma K_2||V^*||_Q)$ where the $||V^*||_Q$ quasi-norm is the span of the value function, $||V||_Q = sup_s V(s) - inf_s V(s)$, h is the grid width, γ is the discount factor, and K_1, K_2 are constants [60].

4.4 Distribution Model Retraining

In Section 4.2 we discussed the abstract formulation of the model management problem as an MDP. In this section, we discuss autonomous model management in the concrete case where the model needs to capture statistical properties of observed data.

One type of model maintenance problem that has multiple uses and serves as a good illustration for a framework is that of distribution tracking. Given a constant stream of multidimensional data, we need to generate a model that tracks the properties of the observed data. In this setting, we are interested in modeling the observed data as a multivariate Gaussian, with (μ, Σ) as the sufficient statistics. We consider observations in the current batch $o \in obs_t$ as modeling success cases, or "hits", if they are within a certain factor of our estimated μ. Similarly, we consider it a modeling failure, or a "miss", if an observation is outside that confidence range. At each timestep, the agent is presented with a set of observations obs_t, drawn from some unknown distribution. The agent is then allowed to sample some subset $\widehat{obs_t}$ of the observations, and decide whether it needs to update the model, and how. As described in the previous section, in this study we consider three options at each timestep - keep the model unchanged, retrain a new model from scratch based on the new observations, or use transfer learning to update the old model with the new model. In the next subsection, I show how this setting is formally modeled as a Markov Decision Process.

4.4.1 MDP Representation for the Distribution Tracking Problem

Following the formulation presented in subsection 4.1.2, we define the state as a tuple $\langle M_t = \langle \mu_t, \Sigma_t \rangle, obs_t \rangle$, with M_t and obs_t the model and the new observed data at time t. In each timestep the agent has the choice of either *retraining* a model from scratch based on the new data, *adapting* (or transfering) the current model to the new data, or leaving the current model as is. In the distribution tracking setting, updating the model from scratch simply means taking the maximum likelihood estimate of μ and Σ based on the sampled observations at time t, $\widehat{obs}_t$. In this study, adapting the model means taking the average between the current model and the new model estimates $\mu_{t+1} = (\mu_t + \overline{\widehat{obs}_t})/2$, $\Sigma_{t+1} = (\Sigma_t + var(\widehat{obs}_t))/2$.

The transition function $P(\langle M_t, obs_t \rangle, a_t) = \langle a_t(\langle M_t, obs_t \rangle), obs_{t+1} \rangle$ is defined concretely as:

$$a_t(\langle M_t, obs_t \rangle) = \begin{cases} = \langle M_t, obs_{t+1} \rangle \text{ if } a_t = \textsc{keep} \\ = \langle \langle (\mu_t + \overline{\widehat{obs}_t})/2, \Sigma_t + var(\widehat{obs}_t))/2 \rangle, obs_{t+1} \rangle \text{ if } a_t = \textsc{adapt} \\ = \langle \langle \overline{\widehat{obs}_t}), var(\widehat{obs}_t) \rangle, obs_{t+1} \rangle \text{ if } a_t = \textsc{retrain} \end{cases} \tag{4.1}$$

Finally, the reward function is formulated as $r_t = |O_{modeled}| \cdot acc\_reward - action\_cost$. An observation is successfully modeled if $o \in [(1-\delta)\mu, (1+\delta)\mu]$, where δ is the slack or accuracy factor. This concept is illustrated by example in Figure 4.1. We note that given a current model and an assumed distribution, the expectation of the reward can be computed analytically, a property we use in the value iteration process. The action cost is determined according to the concrete domain of application, and needs to reflect both the computational increased incurred by the sample size and the difficulty of obtaining samples.

4.4.2 AVI for Distribution Model Retraining

In this section we concretely adapt the general value iteration approach proposed in Section 4.2 to the distribution tracking problem, based on the MDP formulation described above (Section 4.4.1).

Recall that in the distribution tracking case, the model tracks the mean and the variance of an observed data stream as a multivariate Gaussian. For simplicity, let us assume that we are only interested in tracking the mean, that the data is one-dimensional, and that our current estimate of the mean is μ. Now we observe a sample set from the next data batch, $\widehat{obs}$. If we assume that new observations can only deviate by a certain range $C = C_{\text{obs}} \cdot \text{var}(\widehat{\text{obs}})$ from the current mean, we can assume a putative range for the expected mean of future observation batches, $\overline{\widehat{obs}} \in [\mu - C, \mu + C]$. Since the observations represent the same distribution we are trying to model, C may be used for both model and observation space discretization.

As described in Section 4.2, we discretize the range $[\mu - C, \mu + C]_x$ and perform value iteration on the grid values, with all the properties known. At every step of the dynamic programming

process, we are essentially asking the following: if I observe k samples drawn from a distribution with mean a and variance b, and my current model is with mean c, what is the expected reward? Assuming $\mu_{t+1} \sim \mathbf{N}(\mu_t, var_t)$ and var_{t+1} remains the same, this question can be answered analytically for each state, making the AVI process well-defined, as described in Algorithm 8.

4.5 Distribution Tracking - Empirical Evaluation

In this section we study our model retraining agent architecture in three different domains: one synthetic and two real world datasets. In all the experiments, $\gamma = 0.95$. While adjusting for different discount factors is outside the scope of this paper, our experimentation with other γ values in the range $[0.9, 0.99]$ indicates that results are qualitatively robust in that respect. In our experiments, we assume a utilty of 10 for each serviced request and a cost of 5 for using samples in training. In the case of transfer, a smaller model is trained, using only 50% of the amount of

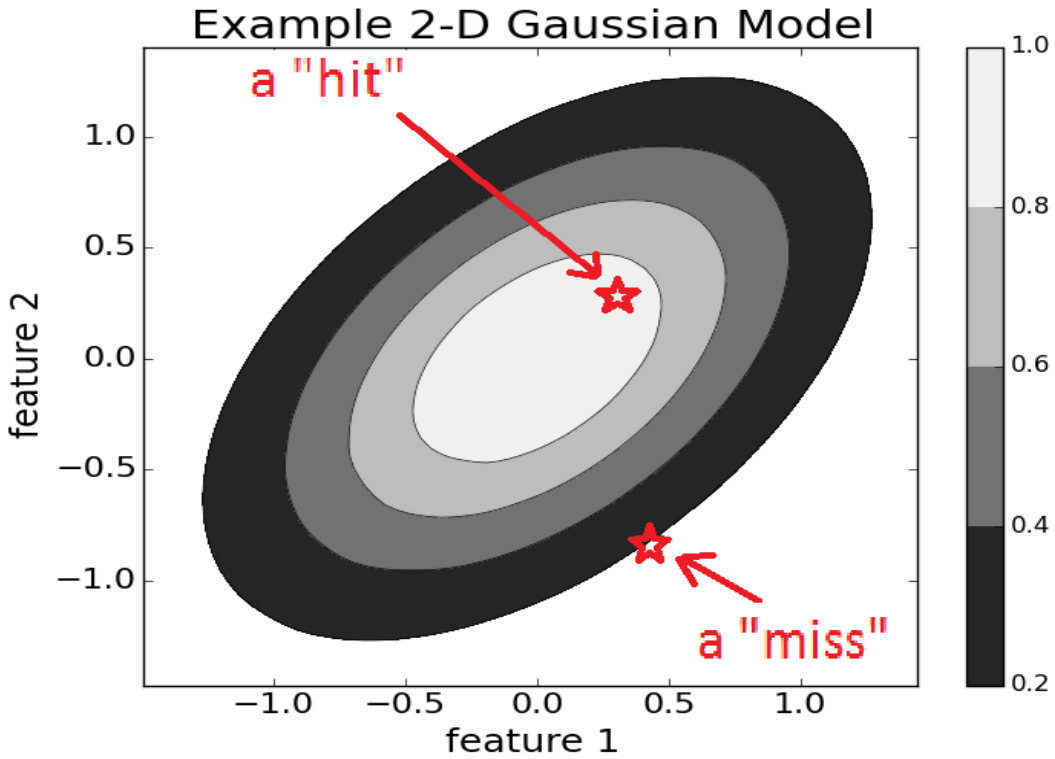

Figure 4.1: An example 2-dimensional model for a data distribution. A new observation that is sufficiently "close" to the mean estimate (as determined by a parameter) is considered a success (or a "hit"), whereas an observation that is not is a failure (or "miss"). X and Y axes are in 2-dimensional feature space, color represents estimated distribution frequency.

Algorithm 8 Distribution Tracking Approximate Value Iteration

Input: observation sample $\widehat{\text{obs}}$, current model mean μ, current variance estimate Σ, variation fraction C, discretizing factor k, discount factor γ.
Set $C = C_{\text{obs}} \cdot \text{var}(\widehat{\text{obs}})$
Discretize range $\widehat{M} = [\mu - C, \mu + C]_k$, yielding an approximate state space $\mathcal{S} = \widehat{M} \times \widehat{M} = \{\langle m, \overline{obs} \rangle \mid m, \overline{obs} \in \widehat{M}\rangle$.
while *not converged* **do**
    $V_{i+1}(s) := \max_a \{\sum_{s'} P_a(s, s') (R_a(s, s') + \gamma V_i(s'))\}$
end while
return policy:

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s, s') \left(R_a(s, s') + \gamma V(s')\right) \right\}$$

samples needed to retrain from scratch. Throughout this chapter we use a sampling rate of 10% per batch. These parameters were chosen for simplicity, as they seem to reflect reasonable tradeoffs. Based on informal experimentation, the results do not seem qualitatively sensitive to the specific parameter configuration.

4.5.1 Proof of Concept - Synthetic Data

As a first step, we concretely illustrate our approach using a synthetic domain. This step is useful since controlling the process which generates the data gives us the freedom to both test the validity of our approach and test its limitations.

Our synthetic domain is simple - data is drawn from a 2-dimensional Gaussian distribution with unknown $\langle \mu, \Sigma \rangle$ parameters. At each timestep, a number of observations drawn from a Poisson distribution $|obs_t| \sim \textbf{Poi}(\lambda)$. Additionally, the distribution shifts in a random walk process at each timestep - μ and Σ drift by factors drawn from a different unknown distribution is added. To make the distribution meaningful, an upper bound is placed on the value of the true underlying variance.

We compare our value iteration approach to seven baseline policies: (1) a "do nothing" policy which always stays with its current model; (2) a "retrain always" policy, which retrains a model, paying the cost associated with this action, at each timestep; (3) an "adapt always" policy, which always updates the existing model using new data; (4) a random policy, which chooses actions uniformly irrespective of the current state at each timestep; (5) a fixed policy, which adapts the model if the observed data has deviated by more than 25% of the current estimate, and retrains if it has deviated by more than 50%. (6) DDetM [116], a drift detection method that identifies potential drift based on the distributional properties of the error rate. DDetM identifies two levels of drift risk - "warning" level and "drift" level, so these two levels correspond naturally to "transfer" and "retrain" actions, respectively (default parameters were used). (7) HDDM [114], a more recent approach to drift detection that relies on Hoeffding bounds. HDDM also identifies "warning" and "drift" modes for the data, which again correspond with "transfer" and "retrain" (default

parameters were used).

We analyze the performance of each model reuse strategy over 30 timesteps with randomly drawn batches in this synthetic domain. The experiment is repeated over 10 iterations at each step to provide a measure of statistical significance to the reported results. The results are provided in Figure 4.2. As can be observed, our system significantly outperforms the other baselines. Figure 4.2(a) shows the overall average reward per step, whereas (b) and (c) show the accuracy and the costs, respectively. In Figure 4.2(a), one can observe that the AVI approach (labeled here and elsewhere in this chapter as "value_iter" for greater visual convenience) gets a reward on par or better than the other approaches in timesteps $\{1 \ldots 30\}$. Breaking down these results in Figures 4.2(b) and 4.2(c), one can observe that the success rate (i.e. accuracy) of the AVI-produced models is by and large on par with the best performing alternative approach at each timestep, and obtains good performance while incurring considerably lower retraining costs. This outcome illustrates how the AVI approach carefully balances these two considerations in model retraining.

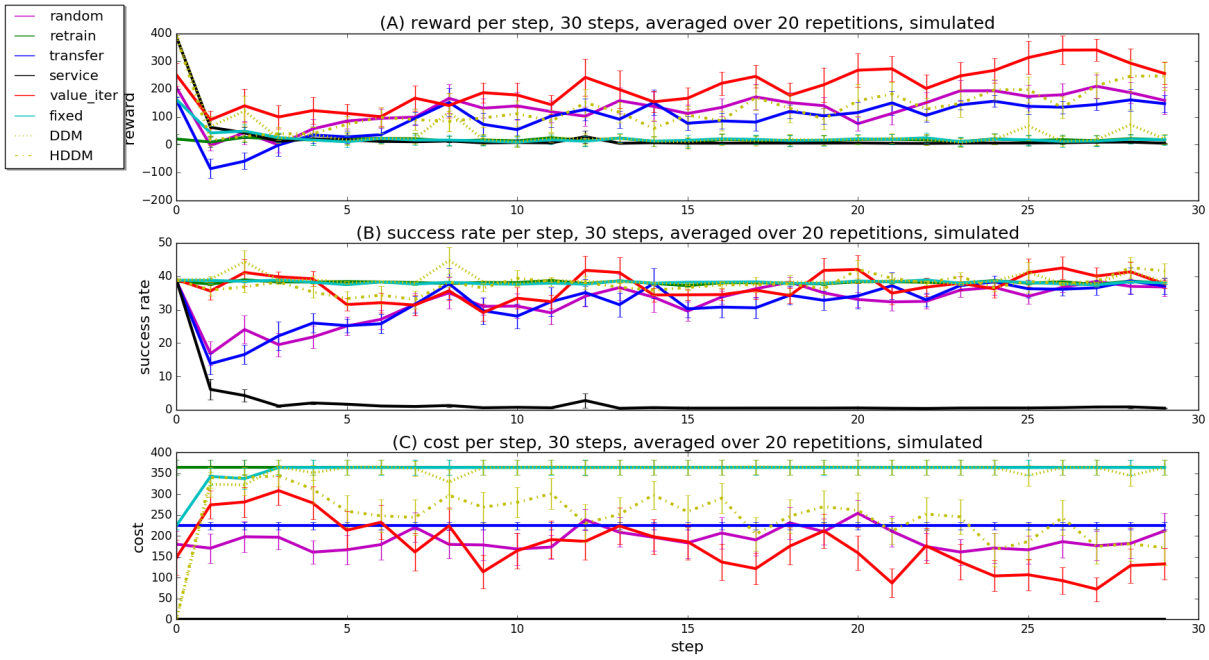

Figure 4.2: Reward per step over 30 time steps for our approximate value iteration (AVI) system compared to the other model retraining policies. Results are averaged over 20 simulations per step and are statistically significant (using a paired t-test[a]). Results are in the synthetic domain. (a) avg. reward (b) average success rate (c) avg. cost. Figure is best viewed in color.

[a] **In all places where mentioned, statistical significance was determined for cumulative reward at the end of the 30 steps for all agents.**

4.5.2 Real World Domain I - ThisIsMyJam Dataset

It is important to ascertain whether our approach holds merit with real world data that may not (and is indeed unlikely to) behave like a constantly shifting Gaussian distribution. For this purpose we look at an interesting real world domain that specifically involves music: the ThisIsMyJam archive, which curates people's reported song preferences (AKA "jams") [168]. This Is My Jam (2011-2015) was an online social music network where users could post one song at a time, their current "jam". A jam lasted for up to a week, emphasizing the ephemeral nature of jams as favorite songs at that particular point in time. By cross-referencing the attested "jams" with the Million Song Dataset [33], one can extract many auditory features of songs liked by multiple people over varying time spans between the years 2011-2015. As a proof of concept, we set out to track the distribution of tempo and loudness of preferred songs on a weekly basis, using the same assumptions used in the synthetic domain. This task makes a good real-world test case for handling drift, since aggregated song preferences can vary wildly over short periods of time, and being able to track distributional properties of musical preferences is a valuable aspect of any music recommendation system that operates over extended periods. In the following experiments, the grid granularity we used was 50, and $C = 3$.

Figure 4.3(a) illustrates the results in this domain. The results indicate that, as before, the value-iteration-based approach we propose consistently outperforms the baselines in terms of balancing performance and cost. Note that the AVI approach does well despite the fact that the underlying distribution of the data is unknown and non-trivial. Furthermore, it succeeds even in the face of considerable fluctuations and high levels of noise. These fluctuations in the data can be observed by the correlated shifts in performance all the algorithms demonstrate as the sequence of batches unfolds.

4.5.3 Real World Domain II - ECOMP

Lastly, we study the effectiveness of the proposed framework for tracking distributions on another, fundamentally different real world dataset, one that is not related to music or content recommendation per se. This *chats*-based dataset, extracted from AT&T's ECOMP Framework, is a dataset looks at real world anonymized data derived from a customer care scenario, where a model must determine if a customer's needs were fully satisfied after a digital chat with a representative. This domain is of interest because success in tracking the profile of customer requests over time suggests overall applicability of the proposed drift management approach for many real world online services.

In this study, digital chats are pooled from 9/1/15 to 4/1/16 (214 days, over 1.8M samples) and transformed into term frequency (1363 features) with annotations as either true (the issue was resolved) or false (a subsequent chat was received) - this last annotation serves as the label we wish to predict. In this experiment, however, we are only interested in tracking the request distribution, reflecting the frequencies of certain terms. At each experiment we build a joint model on the top

two principal components of the word count features. The results are presented in Figure 4.3. As before, we compare to the other baseline policies described above, and show our RL approach significantly outperforms the other ones. The grid granularity used was 50, and $C = 5$. Since data is provided in day ranges, in our experiments we treat all requests on a given day as a single batch.

Figure 4.3(b) presents the results in this domain. As in the previous domain, our results indicate that in this environment our proposed method does particularly well, outperforming the other baselines by a large margin. This increase in relative performance can be attributed in part to the structure of the data, in which the variance is lower but the data stream is still erratic enough that having fixed decision thresholds (either with respect to accuracy or with respect to the observed data distribution) does worse than adaptively deciding on the threshold.

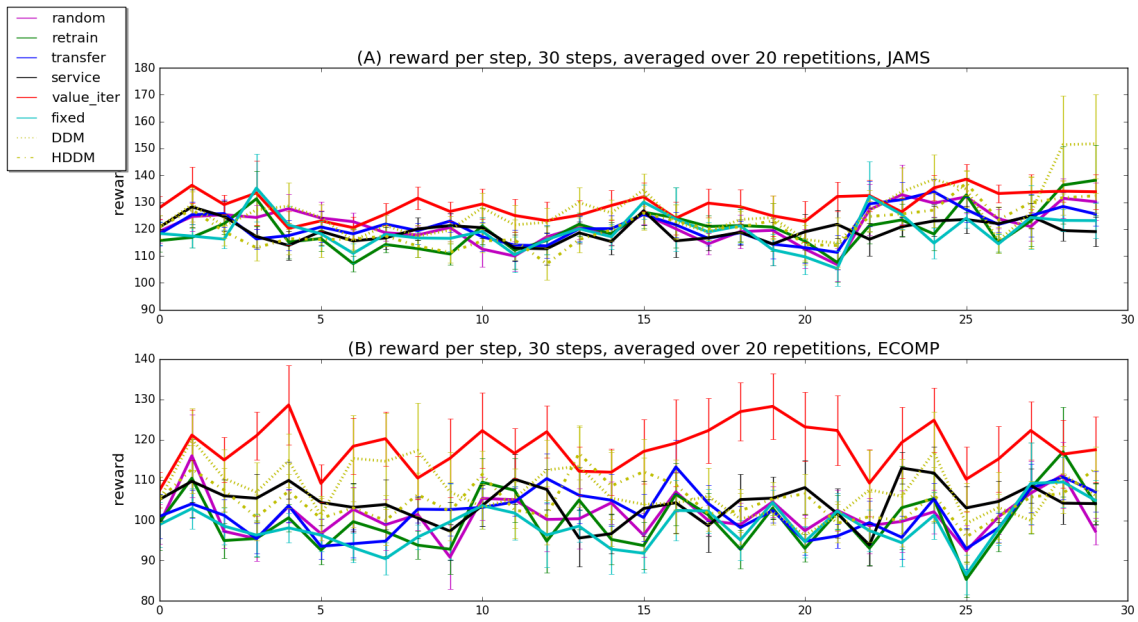

Figure 4.3: Reward per step over 30 time steps for our approximate value iteration (AVI) system compared to the other model retraining policies. Results are averaged over 20 simulations per step. (a) Results are in the song preference domain. (b) Results are in the AT&T ECOMP domain. Figure is best viewed in color. In both cases results are statistically significant (using a paired t-test).

4.6 Prediction Model Retraining

In this section we discuss another instantiation of the autonomous model management problem, that of maintaining prediction models.

While tracking feature distributions is a valuable task, in many real world scenarios the purpose of a trained model is to use the incoming data to predict other, hidden properties. In this section we study this problem, focusing on binary label classification. Given a constant stream of multidimensional data, we generate models that are meant to track the distributional properties of this data, *as well as* its correspondence to a predicted variable (or label).

4.6.1 MDP Representation for the Prediction Problem

The case of tracking a *function* of the data may seem inherently different, but conceptually, it is in fact a generalization of the previous scenario. From the state space perspective, recall that in the distribution tracking setting described in Section 4.4 the states are defined as the tuple $\langle M_t, obs_t \rangle$, with M_t and obs_t the model and the new observed data at time t. In the prediction model retraining case, the formulation is exactly the same, but the internals of M are different - instead of storing sufficient statistics about the distributions, the model now contains a parameterization of the mapping from the observed data to the predicted label. However, in general structure, this task can also be seen as a concrete instantiation of the general problem described in Section 4.4.1. If in the distribution tracking case we were only interested in having new observations be sufficiently close to our model estimation, we now wish to explicitly predict labels for observations. In this case, a success (or a "hit") is a case when we predict the label of a new observation correctly, and a failure (or a "miss") is the case we didn't. It is easy to see now why this setting strictly generalizes the distribution tracking case - one can construct a prediction task where the predicted label is "1" if the observation is within a certain ellipse around the distribution mean, and "0" otherwise (where the hidden decision boundary needs to be learned by the classifier). This model prediction formulation exactly mimics the distribution tracking case.

In cases where the model can be explicitly parameterized, almost the exact approach as that taken in Algorithm 8 can be taken, with only a few modifications, namely to the grid partitioning of the state space, which is now defined as $\mathcal{S} = \{\langle m, \overline{\text{obs}} \rangle \mid m \in \widehat{M}, \overline{\text{obs}} \in \text{OBS}\}$.

While not all modeling approaches are equally amenable to this type of model parameterization, many useful families of classifiers, such as artificial neural networks and logistic regression (which we use in this paper), expose their internal parameters. This property makes such models particularly suitable to be used in conjunction with the model management framework proposed in this study.

4.6.2 Real World Domain - ECOMP

As a proof of concept for the predictive model update case, we once again use the AT&T's ECOMP data. In our experiment, we use the two features most correlated with the predicted label. As an underlying predictive model we use logistic regression. As described above, this model is straightforwardly usable within the context of our framework since it explicitly exposes its parameters.

Recall that in the logistic regression setting we fit coefficients β_0, β_1 such that they capture the log odds ratio $ln(\frac{Pr(label=1|x)}{1-Pr(label=1|x)}) = \beta_0 + \beta_1 \cdot x$. Given that this is the case, an arithmetic mean of the coefficients is a reasonable analogue to the notion of transfer defined in the distribution tracking case. We also note that the expected utility of servicing requests adhering to one decision rule based on another decision rule can again be computed analytically. In this case, we used a grid granularity of 10 per dimension, $C = 5$, and days as timesteps. Using the ECOMP dataset to monitor a predictive model's ability to classify user requests as solved vs. unsolved is a real world use case for the proposed framework. We show that despite this setting being significantly harder, the AVI approach still does a better job of balancing cost and accuracy over time compared to the seven baselines we tested.

The results in this experiment are presented in Figure 4.4. Even in this considerably more complicated case, the RL approach we present in this chapter is significantly superior to the other model update policies examined. Mirroring the observations made observing Figure 4.2, we note that the AVI approach achieves overall accuracy on par with that of the best method at each timestep (4.4(b)), while doing so at a consistently lower average cost (4.4(c)), thus yielding an overall better score by balancing accuracy with model retraining costs (4.4(a)).

4.7 Scaling Up Model Retraining with Fitted Value Iteration and Deep Neural Nets

As encouraging as the results in sections 4.5 and 4.6 are, the concrete implementation used in those sections is not without limitations. Perhaps most important is the issue of scale - any reinforcement learning paradigm reliant on grid value iteration is heavily subject to the curse of dimensionality. Fortunately, alternative function approximators, more amenable to high-dimensional input, exist. In this section, we adapt a different approximate value iteration approach, Fitted Value Iteration [125], to the model retraining problem, using a Convolutional Neural Network [200] as the approximator. We subsequently demonstrate that with this modification our framework scales to much higher dimensional input.

4.7.1 Fitted Value Iteration

Sampling-based Fitted Value Iteration, or FVI [125], is an off-policy approximate dynamic programming algorithm that computes an approximation of the optimal value function, $\hat{V}(s)$ by repeatedly sampling a finite subset of the state space, $S_{FVI} = \{s^1, s^2, \ldots, s^m\}$, and using it to perform value iteration, while also refining a function approximator that predicts $\hat{V}(s)$ for $s \in S$ that haven't been observed yet. More concretely, in our general case:

$$\forall i \in \{1 \ldots m\}$$

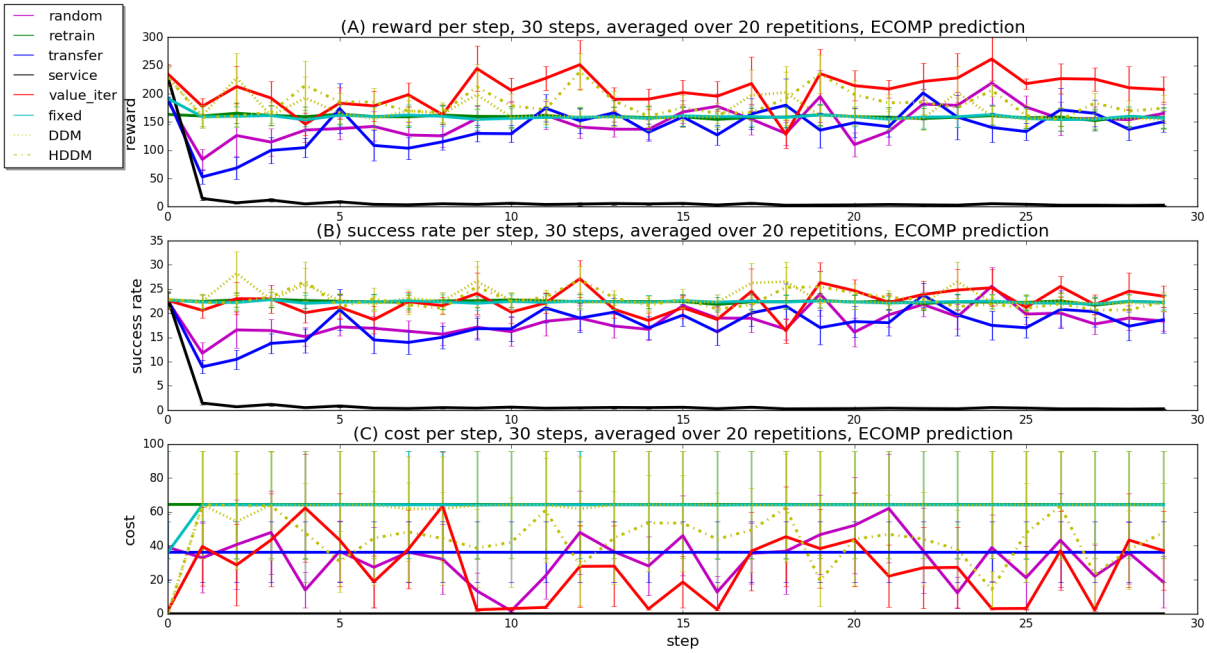

Figure 4.4: Reward, success rate and cost per step over 30 time steps for our approximate value iteration (AVI) system compared to the other model retraining policies in the AT&T ECOMP domain. In this experiment, the models do not simply track a distribution but rather attempt to make predictions over the data. Results are averaged over 20 simulations per step and are statistically significant (using a paired t-test). Figure is best viewed in color.

$$y^i = R(s^i) + \gamma \cdot max_a(E_{(s'|s^i,a)}[\hat{V}(s')])$$

$$\hat{V}(s) := SL(\{(s^1, y^1, (s^2, y^2), \ldots, (s^m, y^m)\})$$

$\hat{V}(s)$ is initialized arbitrarily (in our case, most conveniently, to all zeros). After each update scan, a supervised learning algorithm SL is used as a function approximator that approximates the value function over the complete state-space, based on the "labeled" examples generated with by the previous version of the approximator, $\{(s^1, y^1, (s^2, y^2), \ldots, (s^m, y^m)\}$. While FVI is not guaranteed to converge to the optimal policy, it often performs well in practice, and is theoretically well-behaved [376].

4.7.2 Generalizing the Model Retraining Framework with Neural Fitted Value Iteration

In this subsection we consider how to concretely extend the distribution tracking approach described in Section 4.4 to an arbitrary number of dimensions using an FVI approach. Instead of sampling a grid representation of the entire state space, we wish to iteratively sample from the state space

Algorithm 9 Distribution Tracking Fitted Value Iteration

Input: observation sample $\widehat{\text{obs}}$ which is a union of observation samples from the last k timesteps, max number of iterations q, initial approximate value function $\hat{V}_0$, discount factor γ.
$i = 0$
while $\hat{V}$ *not converged* and $i < q$ **do**
  draw states $S_{i+1} = \{\langle M^i, obs^i \rangle\}$
  compute approximate reward for each state $\{r_i\}$
  $y_{i+1}(s) := r_i(s) + \max_a \{\sum_{s'} P_a(s, s')\gamma\left(V_i(s')\right)\}$
  Train new approximator $\hat{V}_{i+1} = CNN(S_{i+1}, y_{i+1})$
  $i = i + 1$
end while
The resultant policy:

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s, s') \left(R_a(s, s') + \gamma \hat{V}(s') \right) \right\}$$

$\langle M^i, obs^i \rangle$. For this purpose, we can use the sampled batches of observations to sample $\langle$ model, true observation distribution $\rangle$ configurations. Because in the multivariate case computing the expected utility of a model given a true distribution analytically may be expensive, we opt to approximate this utility using Monte Carlo sampling. Specifically, we draw from the "real distribution" and estimate how many of these requests are adequately handled by the model, giving us an immediate reward signal for the state which fits the analytical estimate in expectation.

For the purpose of function approximation we use a 5-layer convolutional neural net regressor. The training input is a tuple of both the model and true distribution coefficients, and the supervised signal is $R(\langle M^i, obs^i \rangle) + \gamma \cdot max_a(E_{(s'|s^i,a)}[\hat{V}(a(\langle M^i, obs^i \rangle)])$. We iteratively draw a new subset and update the function approximator using FVI until either convergence or a predetermined maximal number of iterations is reached. The full procedure is described in Algorithm 9.

4.7.3 Empirical Evaluation

We illustrate the effectiveness of the FVI approach using a synthetic dataset of a 20-dimensional multivariate Gaussian that shifts randomly in each dimension in each timestep. As before, using a synthetic dataset for testing is useful since controlling the process which generates the data gives us the freedom to fully test the validity and robustness of our approach. Data is drawn from a 100-dimensional Gaussian distribution with unknown $\langle \mu, \Sigma \rangle$ parameters. As before, at each timestep, a number of observations (or requests) is drawn from this distribution (the number of observations drawn is itself a random variable drawn from a Poisson distribution $|obs_t| \sim \textbf{Poi}(\lambda)$). Additionally, the distribution shifts in a random walk process at each timestep - μ and Σ drift by factors drawn from a different unknown distribution. We use the same cost parameters as in Section 4.4, with the same baselines, all generalizable to the 20-dimensional case. The results are presented in Figure 4.5. As one might observe from the results, despite the overall challenge, the value-iteration approach

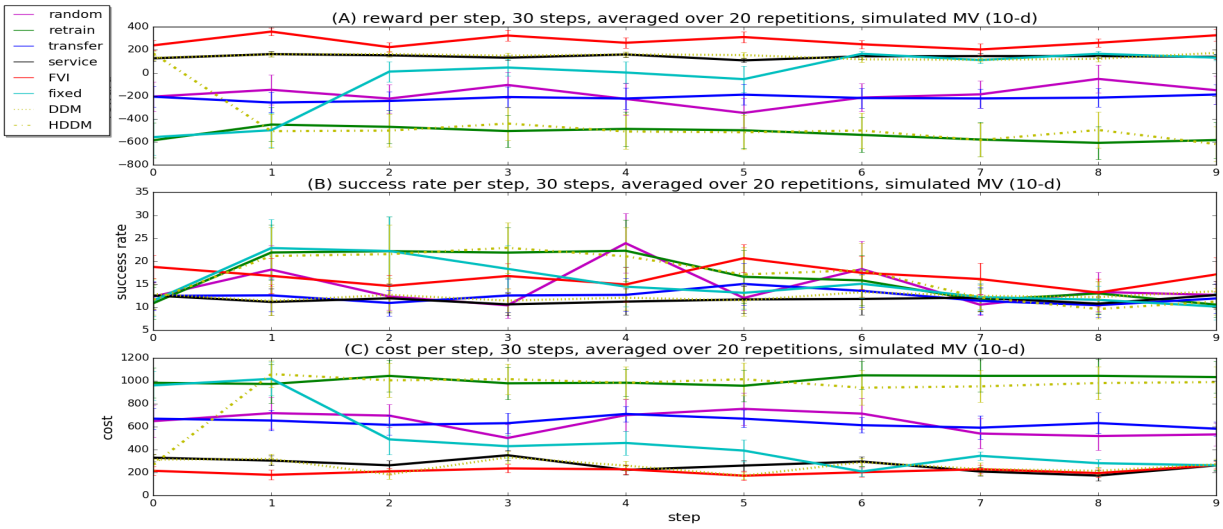

Figure 4.5: Reward, success rate and cost per step over 30 time steps for our fitted value iteration (FVI) system compared to the other model retraining policies on simulated 10-dimensional data. Results are averaged over 20 simulations and are statistically significant (using a paired t-test). Figure is best viewed in color.

still does best in terms of balancing retraining costs and performance.

4.8 Summary & Discussion

The risk of concept drift has a potentially devastating effect on many real world systems that involve modeling, including many content recommendation systems, and music recommendation platforms in particular. To this end, in this chapter I presented a reinforcement learning approach for continual model updating. Rather than building concept drift resistance into the learned model, we frame the model update problem as a sequential decision making task, and adapt an approximate value iteration approach to learn a policy for when to update the model, and how. This framework is generic, has theoretical grounding, and can be easily applied to many different real world systems. We empirically evaluate our approach on three different datasets, and show it outperforms other baseline update policies. The approach presented in this chapter was able to handle music preference data as reflected by the ThisIsMyJam dataset, thus providing strong evidence on the utility of this approach in the context of combating concept drift in long-term content recommendation, in fulfillment of Contribution 5. The applicability of the proposed approach in the vastly different real world ECOMP dataset illustrates the generalizability and overall utility of algorithms and frameworks initially designed in music-related contexts, in line with Contribution 6 of this thesis.

In this chapter we discussed online adaptation of models over time, in recommendation

settings or otherwise. In the next two chapters, I will connect the issue of reasoning about individual preferences with explicitly modeling the impact of music on human decision-making, and how that, in turn, may alter person-agent interaction.

5. Modeling the Impact of Music on Human Decision-Making

In the previous chapter, I discussed the problem of music recommendation as a sequential decision-making problem, as people's preferences and expectations are informed by what has been played up to any given point in time. However, there are other ways in which people's preferences and expectations are influenced by music. A relevant question in the context of studying interactive processes between people and automated systems is how background information, and music in particular, impact the way people make decisions. This chapter focuses on this question in two distinct contexts which engage different decision-making processes.

Numerous studies have demonstrated that mood can affect emotional processing (as discussed in Chapter 2.3). In this chapter I describe two human study experiments we conducted and propose models to study the impact of music on human decision-making in these contexts. The first experiment focused on emotional processing; participants decided whether words were emotionally positive or negative while listening to music that was chosen to induce positive or negative mood. The second experiment involved a quantitative task; participants decided whether to accept or reject multiple bets with different risk vs. reward ratios while listening to music. The results of each experiment in itself are of interest, but when juxtaposed, they reveal interesting insights regarding how music affects people differently in different cognitive contexts.

This research thread establishes that music does indeed impact people's decision-making in nontrivial ways, that this impact can be modeled efficiently using a stochastic model of evidence accumulation, and that various concrete aspects of the music can be correlated with specific aspects of corresponding human behavior. This thread constitutes Contribution 3 of this thesis by proposing a concrete approach for explicitly modeling the impact of music on human decision-making in two different settings.

In the next subsection I discuss one of the main technical tools used in this chapter to model and analyze decision-making behavior: the Drift-Diffusion Model. In Sections 5.2 - 5.5 I describe the first experiment. In Sections 5.6 - 5.9 I describe the second experiment. In the next chapter (6), in fulfillment of Contribution 4, I further study the impact of music on human decision making, and how an agent which actively models this aspect of people's behavior could alter its learning to engender better interaction with people. Therefore, this chapter, in addition to serving to satisfy Contribution 3 of this thesis, also provides the necessary conceptual foundation for satisfying Contribution 4. This chapter is based on published research conducted jointly with Corey White [221, 222, 415].

5.1 The Drift-Diffusion Model

For the research presented in this chapter, a prerequisite is a computational model of human decision making. A useful model in this context is one that breaks down the complex cognitive process that involves the intake of information, its analysis and the resulting decision into a relatively manageable subset of parameters that can be fitted to the data, so that they could predictively characterize a given person's decision-making. For instance, is a person slow to process the nature of a given input, or is he quick? Is that person cautious, taking longer time to think about which option fits the input best, or perhaps a cursory consideration of the evidence is sufficient?

For this purpose, in this chapter we employ the Drift-Diffusion Model (DDM) of simple decisions, to relate observed decision behavior to the underlying decision components [328]. The DDM, shown in Figure 5.1, belongs to a broader class of evidence accumulation models that posit simple decisions involve the gradual sequential accumulation of noisy evidence until a criterial level is reached. In the model, the decision process starts between the two boundaries that correspond to the response alternatives. Evidence is accumulated over time to drive the process toward one of the boundaries. Once a boundary is reached, it signals a commitment to that response. The time taken to reach the boundary denotes the decision time, and the overall response time is given by the decision time plus the time required for processes outside the decision process like encoding and motor execution. The model includes a parameter for this nondecision time (Ter), to account for the duration of these processes.

The primary components of the decision process in the DDM are the boundary separation, the starting point, and the drift rate. Boundary separation provides an index of response caution, or rather, the speed vs. accuracy trade-off; wide boundaries indicate a cautious response style where more evidence needs to be accumulated before the choice is made. The need for more evidence makes the decision process slower, but also more accurate as it is less likely to hit the wrong boundary by mistake. The starting point of the diffusion process (z), indicates whether there is a response bias. If z is closer to the top boundary, it means less evidence is required to reach that boundary, so "positive" responses will be faster and more probable than "negative" responses. Finally, the drift rate (v) provides an index of the evidence from the stimulus that drives the accumulation process. Positive values indicate evidence for the top boundary, and negative values for the bottom boundary. Further, the absolute value of the drift rate indexes the strength of the stimulus evidence, with larger values indicating strong evidence and leading to fast and accurate responses.

In the framework of the DDM, there are two mechanisms that can drive behavioral bias. Changes in the starting point (z) reflect a response expectancy bias, whereby there is a preference for one response even before the stimulus is shown [271, 416]. Experimentally, response expectancy bias is observed when participants have an expectation that one response is more likely to be correct and/or rewarded than the other. In contrast, changes in the drift rate (v) reflect a stimulus evaluation bias, whereby there is a shift in how the stimulus is evaluated to extract the decision

evidence. Experimentally, stimulus evaluation bias is observed when there is a shift in the stimulus strength and/or the criterion value used to classify the stimuli. Thus response expectancy bias, reflected by the starting point in the DDM, indicates a shift in how much evidence is required for one response relative to the other, whereas stimulus evaluation bias, reflected by a shift in the drift rates in the DDM, indicates a shift in what evidence is extracted by the stimulus under consideration. Importantly, both mechanisms can produce behavioral bias (faster and more probable responses for one choice), but they differentially affect the distribution of response times (or RTs for short). In brief, response expectancy bias only affects fast responses, whereas stimulus evaluation bias affects both fast and slow responses (see [416]). It is this differential effect on the RT distributions that allow the DDM to be fitted to behavioral data to estimate which of the two components, starting point or drift rates, is driving the bias observed in the RTs and choice probabilities.

In short, given its representational power, the DDM can differentiate two types of bias:

- Expectancy bias - bias due to an a priori preference for one response over the other; and
- Stimulus bias - bias due to a shift in how the stimuli are evaluated for decision making.

The DDM has been shown to successfully differentiate these two bias mechanisms from behavioral data in both perceptual and recognition memory tasks [416]. However, to my knowledge, it has never been used to investigate effects of music on emotional classification or on risk-related decision-making.

The studies discussed in this chapter use the DDM approach described above to investigate how music-induced mood affects the different decision components under different conditions, examining different types of decision-making. In both cases the DDM was fitted and used to measure how music indeed affects people's decision processes in nontrivial ways, and is useful in highlighting the differences in underlying mechanisms impacted in each experiment.

5.2 First Experiment: Impact on Emotional Classification

There is robust evidence that one's mood can affect how one processes emotional information. This phenomenon is often referred to as mood-congruent processing or bias, reflecting the finding that positive mood induces a relative preference for positive emotional content (and vice versa). The goal of the study discussed in this section was to use the Drift-Diffusion Model (DDM) to explore how music-induced mood affected the different components of the decision process that could drive mood-congruent bias.

Studies that induce mood, either through listening to happy/sad music or having participants write passages or see pictures based on a particular emotion, have shown mood-congruent bias across a range of tasks. Behen et al.[169] showed participants happy and sad faces while they listened to positively- or negatively valenced music and underwent fMRI. Participants rated the happy faces as more happy while listening to positive music, and the fMRI results showed that

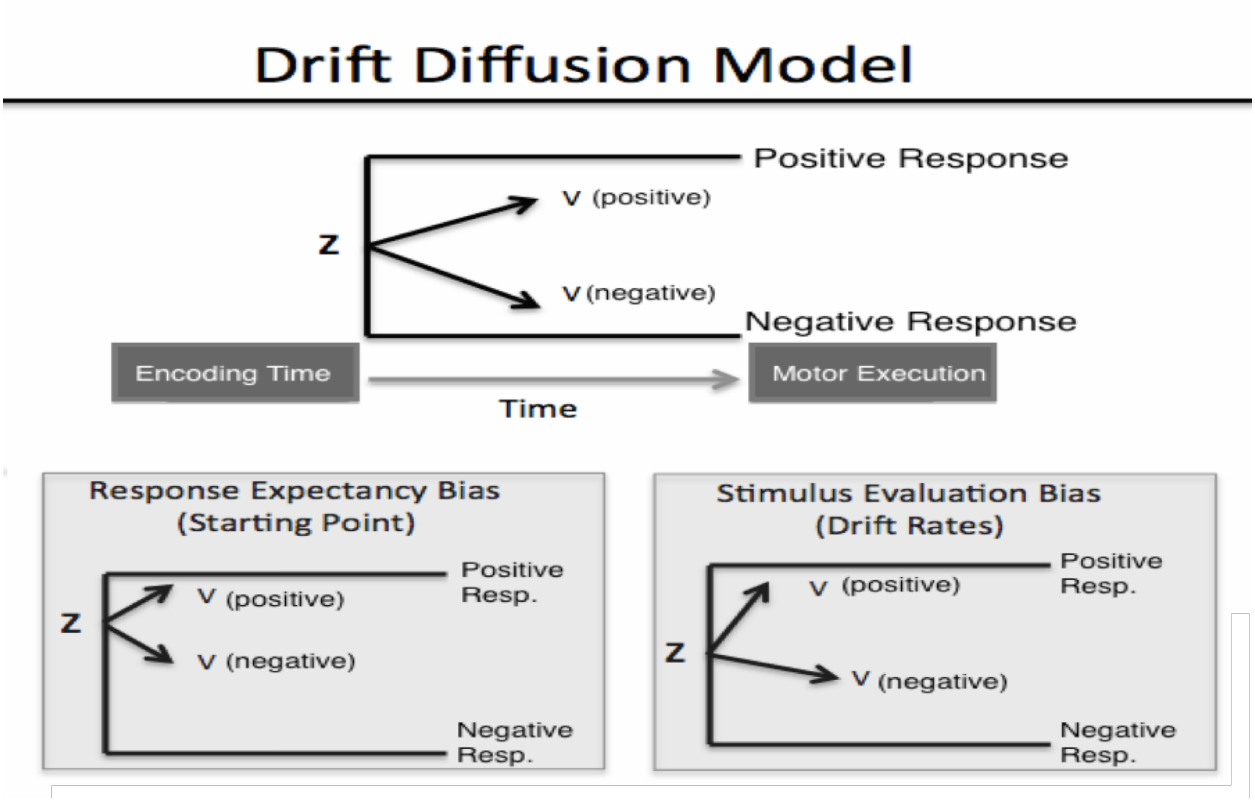

Figure 5.1: An Illustration of the Drift-Diffusion Model.

activation of the superior temporal gyrus was greater when the face and music were congruent with each other. In a study of mood and recall, De l'Etoile [82] found that participants could recall significantly more words when mood was induced (through music) at both encoding and retrieval. Similarly, Kuhbandner and Pekrun [193] had participants study emotional words that were printed in either black, red, green, or blue, with the hypothesis that congruent words (e.g., negative words in red, positive words in green) would show enhanced memory at test. Their findings supported the hypothesis, as memory was better for negative words shown in red and positive words shown in green.

Previous work at the intersection of musicology and cognitive science has also studied the connection between music and emotion. As Krumhansel points out [192], emotion is a fundamental part of music understanding and experience, underlying the process of building tension and expectations. There is neurophysical evidence of music being strongly linked to brain regions linked with emotion and reward [40], and different musical patterns have been shown to have meaningful associations to emotional affectations [300]. Similarly, studies have indicated that mood also affects the perception of music [406]. Not only is emotion a core part of music cognitive processing, it can also have a resounding impact on people's mental state, and aid in recovery, as shown for instance by Zumbansen et al. [437] in the case of people suffering from Broca's aphasia. People regularly use music to alter their moods, and evidence has been presented that music can alter the strength of emotional negativity bias [54]. All this evidence indicates a deep and profound two-way connection between music and emotional perception. In this study, we set out to explicitly model this connection and study its robustness. In the next section, we discuss the concrete design of the first experiment discussed in this chapter.

5.3 Methods of the First Experiment

Participants were shown words on the computer screen and asked to classify them as emotionally positive or negative while listening to music. After a fixation cue was shown for 500 ms, each word was presented in the center of the screen and remained on screen until a response was given. If no response was given after 3 seconds, the trial ended as a "no response" trial. Responses were indicated with the "z" and "/" keys, and mapping between the key and response was counterbalanced across participants. The task consisted of 4 blocks of 60 trials with 20 stimuli from each word condition (positive, negative, neutral). A different song was played during each block, alternating from positive to negative music across blocks. The order of the songs was counterbalanced across subjects. The entire experiment lasted less than 30 minutes. To ensure that the results were not specific to the particular choice of songs, the entire experiment was replicated with a new set of music.

The stimuli (i.e. words to be classified as positive or negative by the participants) consisted of emotionally positive (e.g., success, happy), negative (e.g., worried, sad), and neutral words (e.g.,

planet, sipped) taken from a previous study [414]. There were 96 words for each stimulus condition, which were matched for word frequency and letter length. From each word pool, 80 items were randomly chosen for each participant to use in the task. Words were randomly assigned to appear in the positive or negative music blocks with the constraint that 20 of each word type appeared in every block of trials.

Publicly available music was surveyed to isolate two clear types - music that is characterized by slow tempo, minor keys and somber tones, typical to traditionally "sad" music, and music that has upbeat tempo, major scales and colorful tones, which are traditionally considered to be typical of "happy" music. Our principal concern in selecting the musical stimuli, rather than their semantic categorization as either happy or sad, was to curate two separate "pools" of music sequences that were broadly characterized by a similar temperament (described above), and to show they produced consistent response patterns.

To ensure that the selected music was effective for inducing the appropriate mood, a separate set of participants rated each piece of music on a 7-point Likert scale, with 1 indicating negative mood and 7 indicating positive mood. There were 21 participants that rated the songs for Experiment 1, and 19 participants for Experiment 2. This mood assessment was done outside of the main experiment to eliminate the possibility that the rating procedure would influence the participants' classification behavior in the primary task. The ratings showed that the music choices were appropriate. The positive songs in Experiment 1 led to more positive ratings (song 1: mean = 5.14, SD = 1.24; song 2: mean = 5.00, SD = 1.22) than the negative songs (song 1: mean = 2.24, SD = 1.00; song 2: mean = 2.33, SD = 0.97). Similar results were found for the songs in Experiment two, with higher ratings for the positive songs (song 1: mean = 5.15, SD = 1.29; song 2: mean = 5.42, SD = 1.17) than the negative songs (song 1: mean = 2.26, SD = 1.24; song 2: mean = 2.11, SD = 0.99). The differences between the positive and negative song ratings were highly significant for both experiments (p-values $< .001$).

The DDM was fitted to each participant's data, separately for positive and negative music blocks, to estimate the values of the decision components. The data entered into the fitting routine were the choice probabilities and RT distributions (summarized by the .1, .3, .5, .7, and .9 quantiles) for each response option and stimulus condition. The parameters of the DDM were adjusted in the fitting routine to minimize the χ^2 value, which is based on the misfit between the model predictions and the observed data (see [329]). For each participant's data set, the model estimated a value of boundary separation, nondecision time, starting point, and a separate drift rate for each stimulus condition. Because of the relatively low number of observations used in the fitting routine, the variability parameters of the full DDM were not estimated (see [328]). This choice of estimation procedure resulted in two sets of DDM parameters for each participant, one for the positive music blocks and one for the negative music blocks.

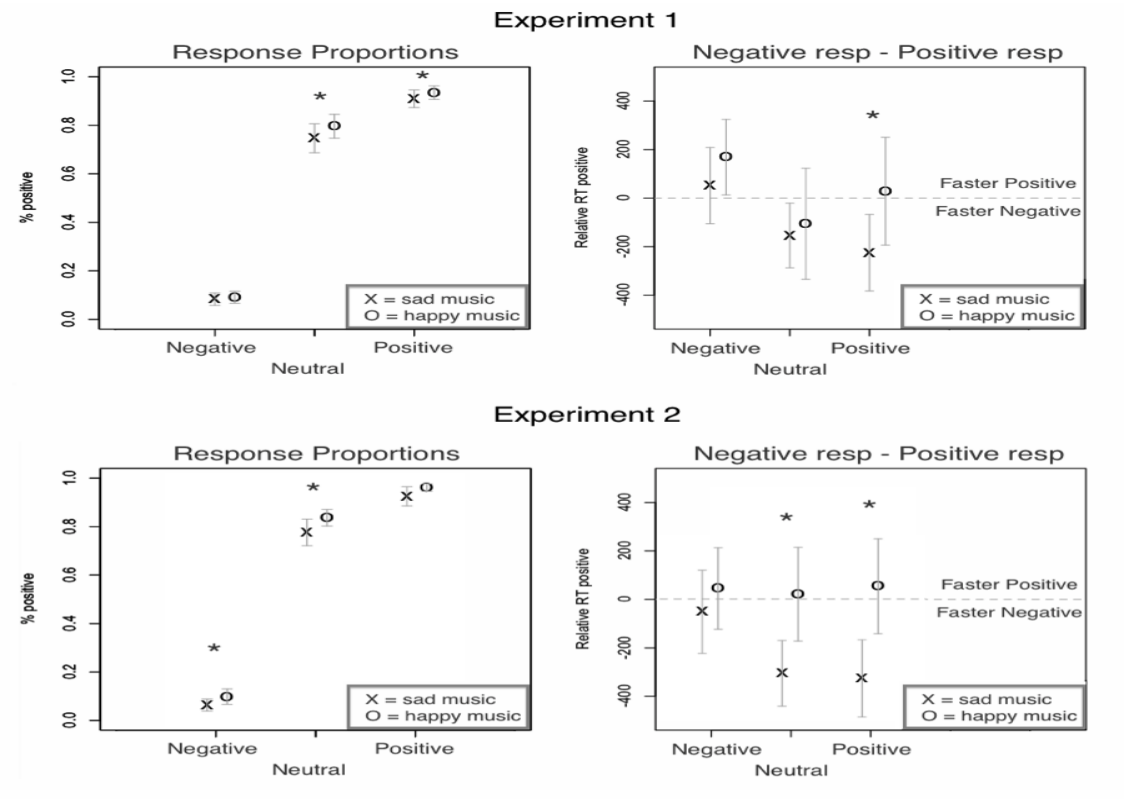

Figure 5.2: Response patterns for the two experiments.

5.4 Results of the First Experiment

The RTs and choice probabilities in Figure 5.2 show that the mood-induction successfully affected emotional bias. The left column shows the response probabilities, and the right column shows an RT-based measure of bias, which is taken as the median RT for negative responses minus the median RT for positive responses for each condition. Thus RT values above 0 indicate faster positive than negative responses for that condition, and vice-versa. In Experiment 1 (top row), happy music led to more "positive" responses overall. This difference was significant for neutral words and positive words, but not for negative words. For RTs, positive responses were generally faster than negative responses in the happy compared to sad music conditions, though the difference was only significant for positive words . The results from Experiment 2 largely mirror those from Experiment 1. Participants were more likely to respond "positive" in the happy music condition. This difference was significant for the negative and neutral words, but not the positive words (though there is a trend in that direction). Likewise, positive responses were relatively faster than negative responses in the happy compared to sad music conditions, though the difference was only significant for neutral and positive words.

Overall, the behavioral data show that the mood induction was effective in influencing participants' emotional classification: positive responses were more likely and faster in the happy

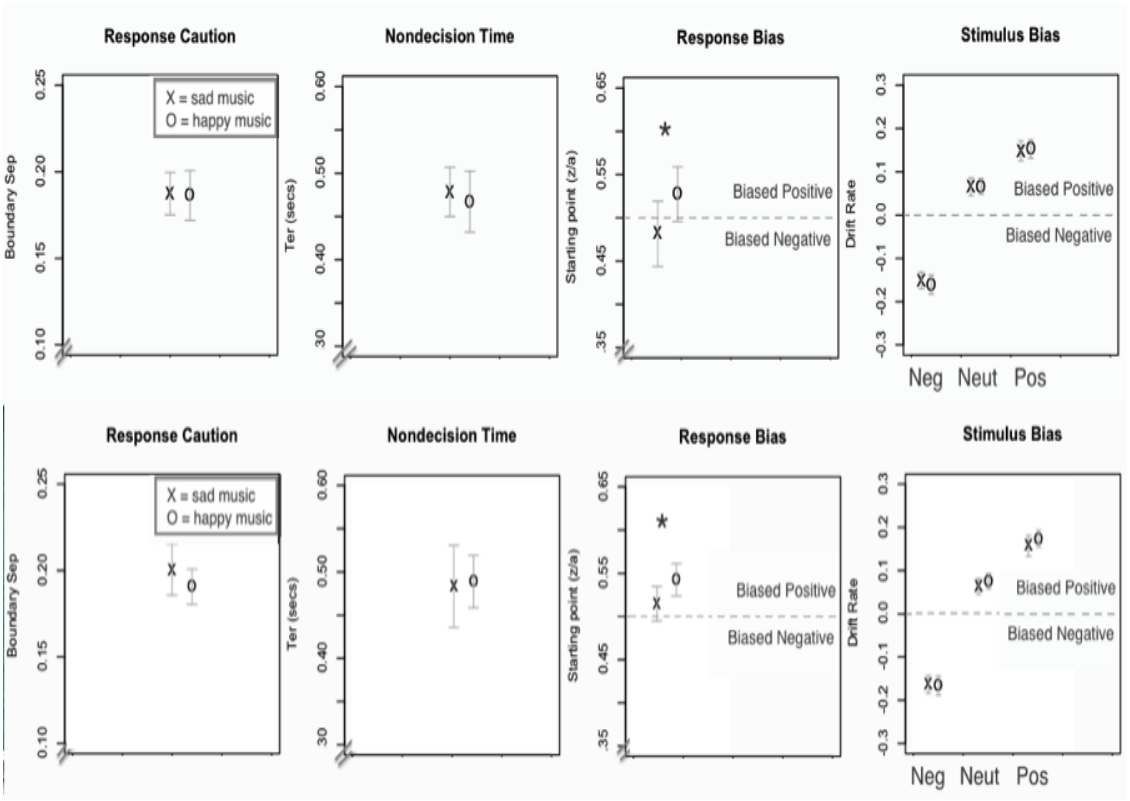

Figure 5.3: DDM fitted parameters for the two music conditions.

compared to sad music condition. These behavioral data are next decomposed with the DDM.

Figure 5.3 shows the DDM parameters for each experiment. Although the two bias-related measures (starting point and drift rates) are of primary interest, all of the DDM parameters were compared across music conditions. It is possible that the different music conditions could affect response caution and nondecision time. For example, the slower tempo of the sad songs could lead participants to become more cautious and have slower motor execution time. Thus all parameters were investigated. As the left columns of Figure 5.3 show, the music conditions did not differentially affect response caution or encoding/motor time, as neither boundary separation nor nondecision time differed between happy and sad music blocks. Of primary interest were the starting point and drift rate parameters, which provide indices of response expectancy and stimulus evaluation bias, respectively. For starting point, there was a significant shift in response bias for both experiments, with participants favoring the "positive" response more heavily in the happy compared to sad music. This result indicates that the music induced an a priori bias for one response over the other. In contrast, the music conditions had no reliable effect on the drift rates for positive, negative, or neutral words. Thus music did not influence the stimulus evaluation of the items. The DDM results show that the music-based manipulation of mood had a targeted effect on the starting point measure, which reflects an a priori response expectancy bias. There were no effects of music

on response caution, nondecision time, or drift rates (stimulus evaluation bias). Thus the results show that the mood-congruent bias was driven by a change in participants' expectancy about the appropriate response, rather than a change in how the emotional content of the words was evaluated.

5.5 Correlating Responses and Musical Features in the Context of Emotional Classification

The partition between "positive" and "negative" mood-inducing songs is intuitively understandable, and is indeed sufficient in order to observe the effects discussed in the previous section. This partition, however, is still somewhat arbitrary. It is of interest then to identify, on a more fundamental auditory level, how specific aspects of music affect response patterns. To this end, we considered the 8 musical segments used in this experiment, extracted key auditory features which we assume are relevant to the mood partitioning, and examined how they correlate with the participant responses we observed.

5.5.1 Extracting Raw Auditory Features

We focused on three major auditory features: a) overall tempo; b) overall "major" vs. "minor" harmonic character; c) average amplitude. Features (a) and (c) were computed using the Librosa library [332]. To compute feature (b), we implemented the following procedure. For each snippet of 20 beats an overall spectrum was computed and individual pitches were extracted. Then, for that snippet, according to the amplitude intensity of each extracted pitch, we identify whether the dominant harmonic was major or minor. The major/minor score was defined to be the proportion of major snippets out of the overall song sequence. We can easily confirm that these three features were indeed associated with our identification as "positive" vs. "negative". Having labeled "positive" and "negative" as 1 and 0 respectively, we observed a Pearson correlation of $0.7 - 0.8$ with p-values ≤ 0.05 between these features and the label. Significance was further confirmed when we applied an unpaired t-test for each feature for positive vs. negative songs (p-values $< .05$[8]).

5.5.2 Processing Participant Responses

For each observed song we first aggregated all relevant subject responses. We focused on three measurements - time delay for classifying positive words as positive, time delay for classifying negative words as negative, and likelihood of classifying neutral words as positive. Time delays were normalized to a z-score per user. This alternative perspective helps verify the robustness of the effects observed in the previous section. Following this analysis step, we proceeded to fit the

[8] In this chapter and elsewhere in this thesis, p-values for correlation are results obtained by analysis of the distribution of correlation values given the null hypothesis.

DDM parameter decomposition as we did in Sections 5.3 and 5.4, but rather than for each song condition ("sad"/"happy"), to each song separately.

5.5.3 Observed Correlations

In this section we consider the effects observed when analyzing response patterns with respect to each of the three auditory features discussed in the previous subsections. Only statistically significant correlations are reported, though it's worth noting that with a relatively small sample size in terms of songs, potentially meaningful effects might be missed due to outliers.

Correlation with Response Times and Bias

When we consider how the three auditory features correspond with the normalized delays when classifying positive or negative words as such, we see an interesting pattern. For all three features, there was a statistically significant negative correlation (p-value ≤ 0.05) between the average normalized response time and the feature values. Intuitively speaking, the faster the song was, the louder it was, or the more it was major in mode overall, the faster people classified positive words as positive (see Figures 5.4a-5.4c). However, no such clear correlation was observed for negative songs. This observation supports our key finding when using the drift-diffusion model, that participants were biased to label words positive in the positive music condition. When we analyzed the likelihood of associating neutral words as positive with respect to each auditory feature, the only effect that is borderline significant (p-value ≤ 0.1) is the correspondence between major mode dominance and the likelihood of associating a neutral word as positive (the more major-mode oriented the song is, the more likely people are to associate neutral words as positive) - see Figure 5.4d.

Correlation with DDM Decomposition

We analyzed the correlation between the extracted auditory features and the DDM parameters fitted for each song separately: nondecision time, response caution, response bias, and stimulus evidence (drift rate) for each word type. We found a statistically significant correlation ($r = 0.7 - 0.8, p < 0.05$) between the major dominance feature and the bias and positive drift rate parameters (see Figures 5.5a, 5.5b). A borderline correlation ($r = 0.62, p < 0.1$) was observed between major dominance and the neutral drift rate. These findings support the previous observations in the chapter. Interestingly, we have also observed a borderline significant negative correlation ($r = -0.67; p < 0.1$) between mean amplitude and response caution, implying people are less cautious the louder the music gets (see Figure 5.5c).

It is worth noting that the purpose of this analysis is observational. That is to say, we wish to establish there's a meaningful correspondence between auditory features and the observed differences in decision-making behavior. This analysis in its current form does not account for endogeneity, or co-dependencies among the auditory features themselves. In fact, it stands to reason

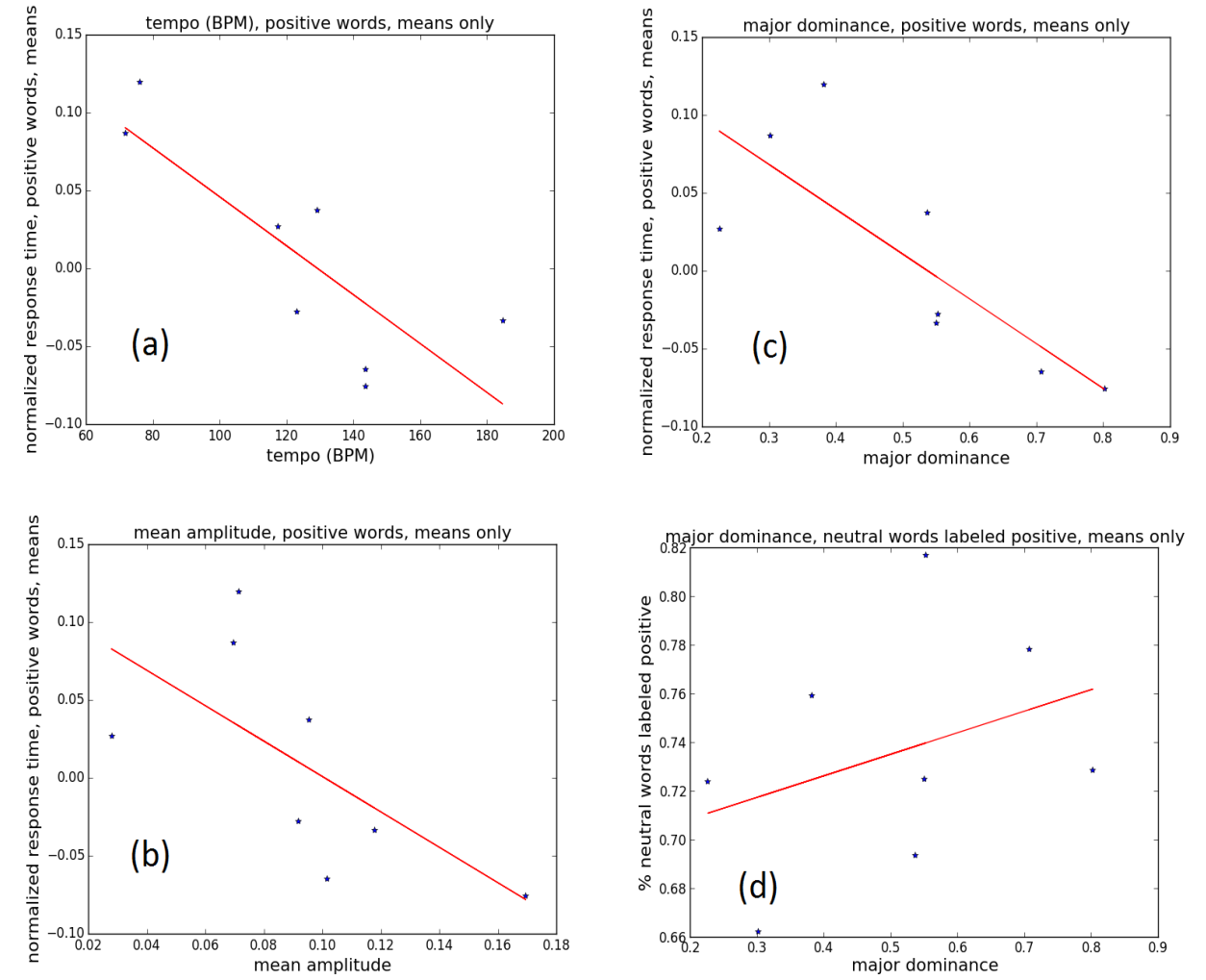

Figure 5.4: (a) Scatter plot of the correlation between the average tempo (as BPM) in a song and the normalized average delay in classifying positive words as such. (b) Scatter plot of the correlation between the average amplitude in a song and the normalized average delay in classifying positive words as such. (c) Scatter plot of the correlation between the percentage of major-mode harmonies (major dominance) in a song and the normalized average delay in classifying positive words as such. (d) Scatter plot of the correlation between the percentage of major-mode harmonies (major dominance) in a song and the likelihood of associating neutral words as positive.

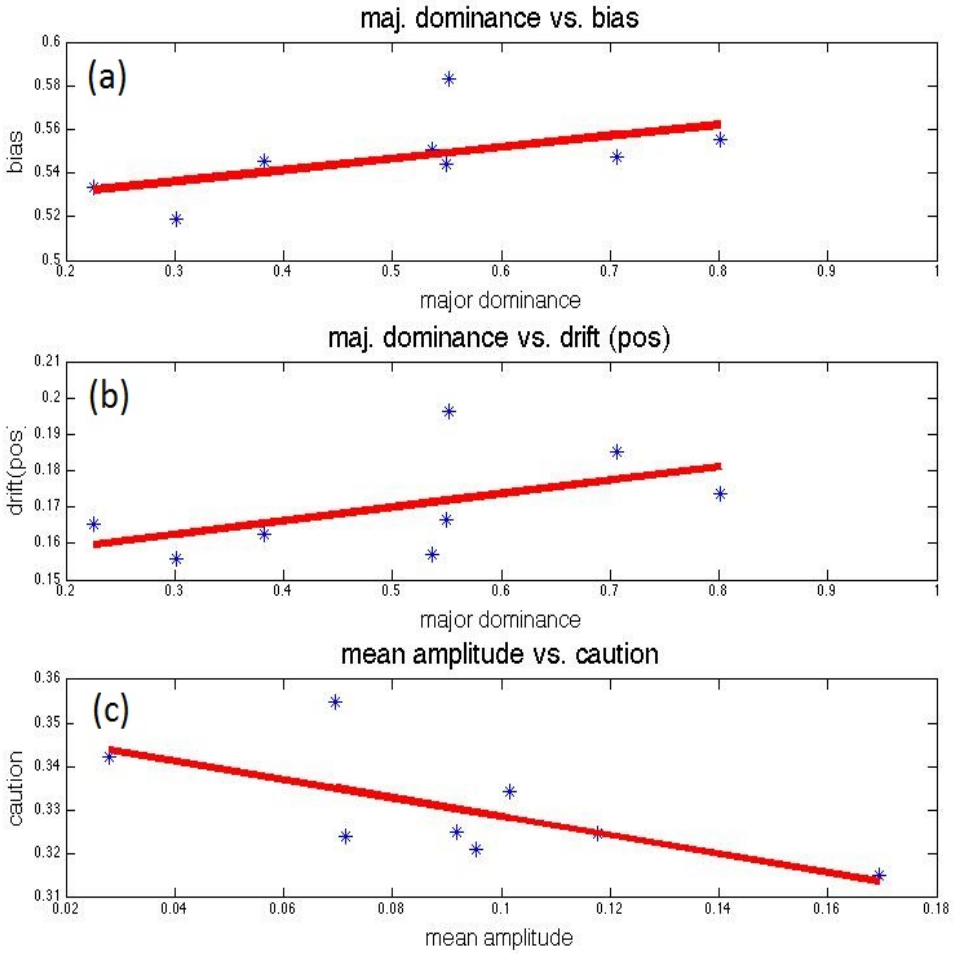

Figure 5.5: (a) Scatter plot of the correlation between the percentage of major-mode harmonies (major dominance) in a song and the bias component of the DDM. (b) Scatter plot of the correlation between the percentage of major-mode harmonies (major dominance) in a song and the stimulus evidence component (drift rate) for positive words in the DDM. (c) Scatter plot of the correlation between the average amplitude of a song and the response caution component of the DDM.

that various facets of the music, such as tempo and scale, do not directly correlate with behavior, but rather affect it through the effect of latent aspects of the music (capturing mood itself). While the results reported in this section strongly suggest a correlative relationship between lower-level facets of music and human decision-making, a larger sample size (with respect to both participants and songs) is needed in order to do a full multivariate analysis of variance on the data.

In the next section we consider a different experiment, which captures the influence of music on a completely different decision-making process, one which involves quantitative, rather than emotional, reasoning.

5.6 Second Experiment: Impact on Quantitative Decision-Making

To recap, there's plentiful evidence indicating one's mood can affect how one processes information. As described in the previous sections, when the information being processed has emotional content,[9] this phenomenon is referred to as mood-congruent processing, or bias, and it's been found that positive mood induces a relative preference for positive emotional content and vice versa.

However, what effect does music have on non-emotional decision making? Numerous studies have demonstrated that mood can affect emotional and cognitive processing, but the exact nature of the impact music has on decision making in quantitative tasks has not been sufficiently studied. In a follow-up experiment designed to specifically focus on the impact of music on quantitative, rather than emotional, decision-making, participants were tasked with deciding whether to accept or reject multiple bets with different risk/reward ratios while listening to music that was chosen to induce positive or negative mood.

Previous work in this area shows robust effects of loss aversion, whereby participants put more weight on potential losses than potential gains. Loss aversion in this context manifests as subjects being unwilling to accept gambles unless the potential gain significantly outweighs the potential loss (e.g., only accepting the gamble if the gain is twice as large as the loss [397, 388]). The present study focuses on whether and how emotional music influences such risky decision behavior.

Not much work has studied the direct connection between music and risky decision making. Some previous work has studied the general connection between gambling behavior and ambiance factors including music [368, 127, 86] in an unconstrained casino environment. Additionally, Noseworthy and Finlay have studied the effects of music-induced dissociation and time perception in gambling establishments [286]. In this section, we take a deeper and more controlled look at how music impacts decision making in this type of risk-based analytical decision making. To this end, we once again use the drift-diffusion model to explore how music-induced mood affects the different components of the decision process in such tasks. Our results indicate that music indeed has a nontrivial and unexpected effect, and that certain types of music led to better decision making than others.

5.7 Methods of the Second Experiment

Participants were presented with simple binary gambles and were asked whether to accept (bet) or reject them (fold). Each gamble had a 50%-50% chance of success, with varying win to loss ratio, reflecting how much was to be gained vs. lost. For example, a 15:5 win-loss ratio reflect a 50% chance to win 15 points and a 50% chance of losing 5 points. After a fixation cue was shown for 500 ms, each gamble was presented in the center of the screen and remained on screen until a response

[9] An example for information which may carry emotional content is words, a fact we have leveraged in the previous experiment.

was given. If no response was given after 3.5 seconds, the trial ended as a "no response" trial. Responses were indicated with the "z" and "/" keys, and mapping between the key and response was counterbalanced across participants.

The gamble stimuli were partitioned to very negative (win-loss ratio in range $[0.33, 0.66)$, average proportion of $1:2$) , negative (win-loss ratio in range $[0.66, 1)$, average proportion of $5:6$), positive (win-loss ratio in range $[1, 2)$, average proportion of $3:2$), and very positive (win-loss ratio in range $[2, 3]$, average proportion of $5:2$). The actual values of the bets were randomized in the range of $[3, 60]$. Each experiment comprised 20 batches of 20 gambles, such that in each batch each stimuli condition was repeated 5 times (gamble order was randomized). Subjects were not shown the outcome of their gambles immediately as that would have been distracting. Instead, between each batch subjects were shown the overall score they accrued for the previous batch (whereas each batch score starts as 0). To encourage competitive behavior, they were also shown the expected score for that batch.[10] A different song was played during each block of 5 batches, alternating from positive to negative music across blocks. The order of the songs was counterbalanced across subjects. The entire experiment lasted less than 30 minutes. To ensure that the results were not specific to the particular choice of songs, the entire experiment was repeated with a large sample of participants ($N = 84$), and two separate sets of songs to assess result reliability.

The music used for this experiment is the same as that used in the previous experiment described in Sections 5.2 - 5.5, as we've already established experimentally that the selected music was effective for inducing the appropriate mood.

As in the previous experiment, the DDM was fitted to each participant's data, separately for positive and negative music blocks, to estimate the values of the decision components. The data entered into the fitting routine were the choice probabilities and RT distributions (summarized by the .1, .3, .5, .7, and .9 quantiles) for each response option and stimulus condition. The parameters of the DDM were adjusted in the fitting routine to minimize the χ^2 value, which is based on the misfit between the model predictions and the observed data (see [329]). For each participant's data set, the model estimated a value of boundary separation, nondecision time, starting point, and a separate drift rate for each stimulus condition. Because of the relatively low number of observations used in the fitting routine, the variability parameters of the full DDM were not estimated (see [328]). As in the previous experiment, this choice of estimation procedure resulted in two sets of DDM parameters for each participant, one for the positive music blocks and one for the negative music blocks.

[10] The expected score for a batch is simply the sum of expected wins (or losses) over trials. For instance, for an individual trial, a betting proportion of $8:5$ has the expected score of $0.5 \cdot 8 - 0.5 \cdot 5 = 1.5$. Aggregated over trials, this value provides a baseline for how a person would perform by betting randomly.

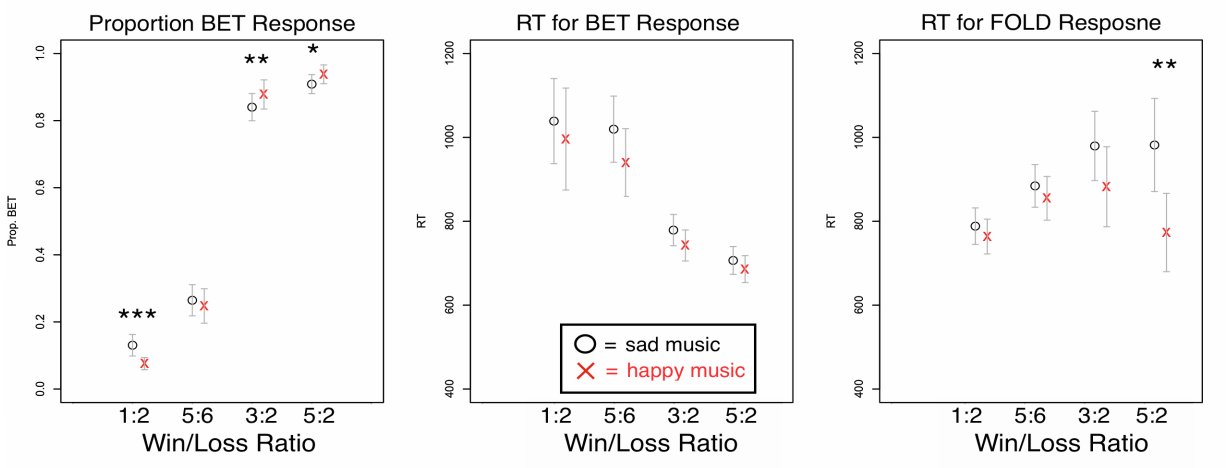

Figure 5.6: Response patterns in terms of response times and bet-fold proportions for the behavioral experiment. A statistically significant difference between the happy song and the sad song conditions is evident for betting proportions given the four clusters of betting ratios (very negative, negative, positive and very positive). There is also a large statistically significant difference between response times for folding in the two different conditions. Error bars reflect 95% confidence intervals. * $= p < .05$; ** $= p < .01$; *** $= p < .001$.

5.8 Results of the Second Experiment

The response times and choice probabilities shown in Figure 5.6 indicate that the mood-induction successfully affected the decision making behavior observed across participants. The left column shows the response proportions, the center column shows normalized response times for betting decisions, and the right panel shows normalized response times for the folding decisions. The betting proportions and response time (or RT) measures for the two conditions - the happy songs and the sad songs - indicate a clear difference between the conditions. Generally speaking, happy music led to more "correct" behavior - participants were more likely to accept good bets and reject bad bets under the happy song condition than the sad song condition. These trends are evident across all gamble proportions and bet-fold decisions, but were only shown to be statistically significant for some of the settings; the difference in betting proportions is shown to be significant for very negative, positive and very positive gambles, whereas the difference in response times is only shown to be significant for folding decisions in very positive gambles. Significance was evaluated using a paired t-test with $p \leq 0.05$.

Figure 5.7 shows the DDM parameters fitted for the experiment. Although the two bias-related measures (starting point and drift rates) are of primary interest, all of the DDM parameters were compared across music conditions. It is possible that the different music conditions could affect response caution and nondecision time. For example, the slower tempo of the sad songs could lead participants to become more cautious and have slower motor execution time. Thus all parameters were investigated. As the top-left and top-center panels of Figure 5.7 show, the music conditions did

not differentially affect response caution or encoding/motor time, as neither boundary separation nor nondecision time differed between happy and sad music blocks. Of primary interest were the starting point and drift rate parameters, which provide indices of response expectancy and stimulus evaluation bias, respectively. Interestingly, as apparent in the top-right and bottom-right panels of Figure 5.7, overall, we did not observe any stimulus (evidence processing) bias nor starting point (response expectancy) bias in the two music conditions. However, the key difference lied in the drift rates themselves. Fitting parameters for the drift rates for the four gamble types indicate an overall change in evidence processing in the happy vs. the sad music conditions, which is statistically significant for all gamble proportions. This outcome is shown in the bottom-left panel of Figure 5.7. In other words, people were faster to process the evidence and make betting decisions for good gambles and folding decisions for bad gambles in happy vs. sad music. This difference is summarized in the bottom-center panel of Figure 5.7, which presents the discriminability factor in the happy vs. the sad condition. Discriminability is defined as the sum of the drift rates for good bets minus the sum of the drift rates for the bad bets, $(d_{positive} + d_{very-positive} - d_{negative} - d_{very-negative})$. This measure represents the "processing gap" between good evidence (good bets) and bad evidence (bad bets). The discriminability was dramatically higher for happy songs compared to sad songs.

The DDM results show that the music-based manipulation of mood affected the overall processing of information in the quantitative task of deciding when to bet and when to fold, rather than any single bias component. There were no effects of music on response caution, nondecision time, or response or stimulus bias, meaning that people weren't more likely to accept bets or reject them in one condition or the other, but rather the change impacted the entire decision process. In other words, the mood change induced by music neither affected the a-priori inclination of people to bet or to fold, nor has it led to a relative difference in processing one type of bet vs. the other, but rather simply made people make better decisions (more likely to accept good bets and reject bad ones).

5.9 Correlating Responses and Musical Features in the Context of Gambling Behavior

Similarly to the analysis done in Section 5.5, we are interested in finding a deeper connection between the behavior observed in the experiment and the different characteristics of music. More exactly, we are interested in finding the correspondence between various musical features, which also happen to determine how likely a song is to be perceived as happy or sad, and the gambling behavior manifested by participants. To this end, we considered the 8 songs used in this experiment, extracted key characterizing features which we assume are relevant to their mood classification, and examined how they correlate with the subject gambling behavior we observed.

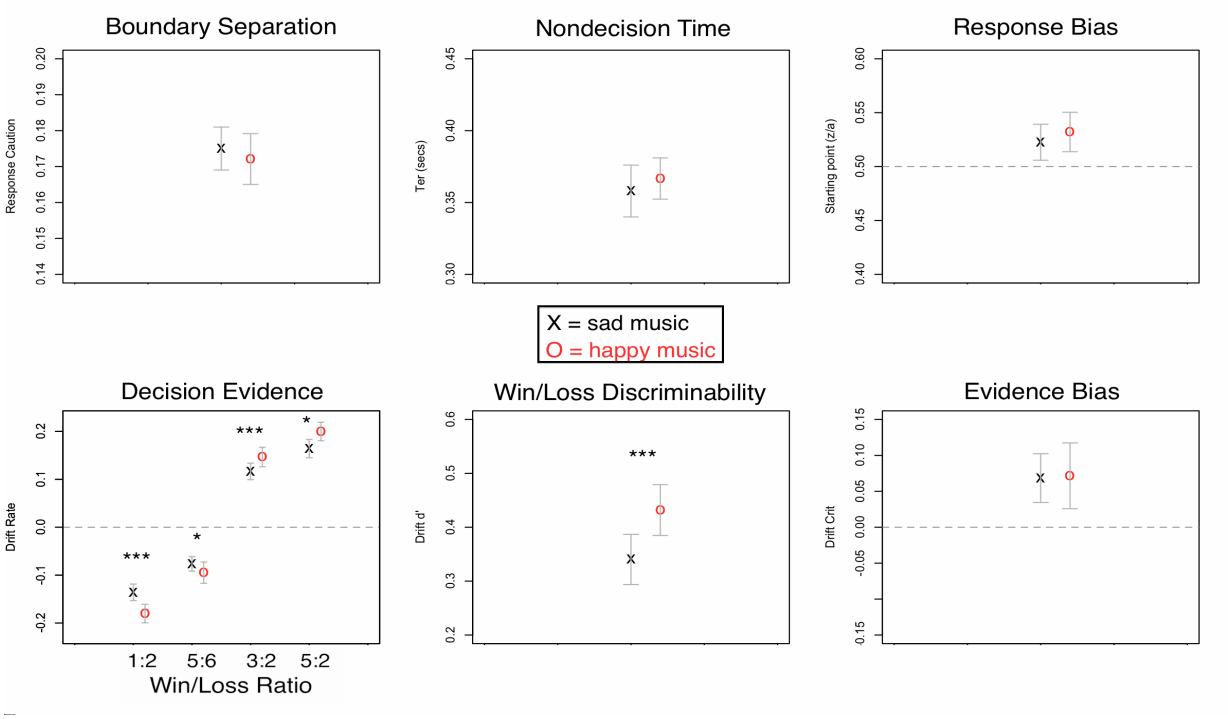

Figure 5.7: Drift-Diffusion Model parameters fitted for the behavioral experiment. A statistically significant difference between the happy song and the sad song conditions is evident for betting proportions given the four clusters of betting ratios (very negative, negative, positive and very positive). There is also a large statistically significant difference between response times for folding in the two different conditions. Error bars reflect 95% confidence intervals. * = $p < .05$; ** = $p < .01$; *** = $p < .001$.

5.9.1 Extracting Raw Auditory Features

Since the same songs were used in the first and second experiments, the raw auditory features of the songs were reused from the analysis done in Section 5.5.1.

5.9.2 Processing Observed Gambling Behavior

Given the complexity of the behavioral experiment discussed in Sections 5.6 - 5.7, several behavioral breakdowns of participant behavior were extracted. Normalizing the response times (RTs) for each participant, we separately considered the average response times for betting and for folding for all four gamble types and songs (64 values overall). Subsequently, we aggregated these average response times per decision (bet or fold), per gamble type (very negative, negative, positive and very positive), per song (4 happy songs, 4 sad songs overall), to obtain 64 average response times and response time variance per ⟨decision, gamble type, song⟩ configuration. Then we could correlate these values per ⟨decision, gamble type⟩ setting with the features extracted for each song. Similarly, we extracted the average bet-fold ratio and bet-fold variance across all participants for each ⟨decision, gamble type, song⟩ configuration as well. As a result we were also able to examine the relationship between bet-fold ratios per ⟨decision, gamble type⟩ setting with the features extracted for the songs.

5.9.3 Observed Correlations

In this section we discuss how the auditory features corresponded with the normalized response time and bet-fold ratio information extracted from the behavioral experiment. We proceed to analyze the more exact correspondence between the DDM parameters as extracted per song individually and the auditory features of the songs. We note that since we are correlating continuous scalar aggregates across users with continuous auditory features, using the assumptions implicit in a standard Pearson correlation is reasonable.

Correlation with RTs and Bet-Fold Ratio

Examining the relationship between the features extracted per song and the response time and bet-fold ratio data discussed in 5.9.2 reveals a compound and interesting picture.

Tempo was consistently and in most cases statistically significantly inversely correlated with response times. This was true for all gamble types and decision combinations. Tempo also tended to be inversely proportional to the observed response time variance. Again, this result was consistent across all gamble type and decision combinations. In other words, generally speaking, not only people responded faster (lower response times) the faster the music was, the variance in response

Decision	Gamble	RT Avg	RT Var.	Avg. p-val	Var. p-val
bet	v. negative	**-0.73**	**-0.61**	**0.03**	**0.1**
bet	negative	**-0.65**	0.48	**0.07**	0.22
bet	positive	**-0.77**	**-0.64**	**0.02**	**0.07**
bet	v. positive	**-0.78**	-0.59	**0.02**	0.11
fold	v. negative	**-0.81**	**-0.65**	**0.01**	**0.07**
fold	negative	**-0.78**	-0.56	**0.02**	0.14
fold	positive	-0.45	-0.45	0.25	0.25
fold	v. positive	**-0.77**	**0.76**	**0.02**	**0.02**

Table 5.1: Correlation values between tempo and response times (average and variance). Results with p-value ≤ 0.1 are marked in bold.

times also tended to be reduced. The observed Pearson correlations for average normalized response times and response time variances across the 8 gamble type and decision combinations is provided in Table 5.1.

Tempo was also inversely correlated with the average bet-fold ratio for very negative gambles ($r = -0.74, p = 0.03$). This also manifested in the correlation with the bet-fold variance ($r = -0.66, p = 0.06$). However, it was linearly correlated with the bet-fold ratio in the very positive gambles case ($r = +0.71, p = 0.04$). Furthermore, in the very positive gambles case, the variance was still reduced, leading to a negative correlation ($r = -0.71, p = 0.04$). In other words, the faster the music, the more people are likely to bet on very good bets, and more consistently (reducing variance). Furthermore, the faster the music, the more likely people are to fold on bad bets, and more consistently (reducing variance). This is a strong signal for how tempo improves the quality of decision making in quantitative tasks.

There is evidence that the major dominance feature (determining the major to minor chord proportion in each song) is inversely correlated to the average bet-fold ratio and the bet-fold ratio variance in the very negative gambles case (average: $r = -0.6, p = 0.11$, variance: $r = -0.61, p = 0.10$). Similarly, there is some evidence that major dominance is linearly correlated with the average bet-fold ratio and inversely correlated to the bet-fold variance in the strong-positive case, but this result wasn't as convincing (average: $r = +0.42, p = 0.29$, variance: $r = -0.58, p = 0.14$). This result, though inconclusive, hints at the possibility that the more major chords there are in a song, the better the analytical decision making that subjects manifest.

Interestingly, the major dominance feature (determining the major to minor chord proportion in each song) was inversely proportional to the variance in response times when folding on a very positive bet ($r = -0.71, p = 0.04$). Major dominance was also inversely proportional to variance in response times betting on a very negative bet ($r = -0.65, p = 0.07$). In other words, the more major chords appeared in a song, the less variability people displayed in the time it took them to make a poor decision. This could be a side effect of people making fewer such mistakes in these gamble - decision combinations, as was documented in previous sections.

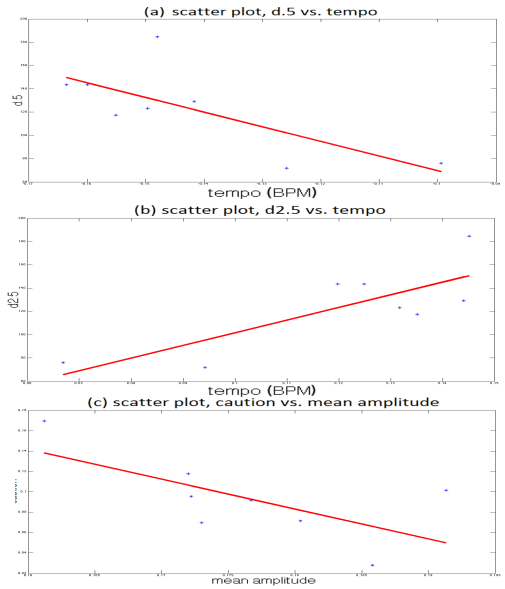

Figure 5.8: (a) Correlation between tempo and the drift rate for very negative gambles. (b) Correlation between tempo and the drift rate for very positive gambles. (c) Correlation between mean amplitude and the overall response caution (boundary separation).

The average amplitude was inversely correlated to the average bet-fold ratio and the bet-fold ratio variance for negative and very negative gambles. These observations seem borderline significant (average: $r = -0.59, p = 0.12$ for negative, $r = -0.53, p = 0.17$ for very negative, variance: $r = -0.58, p = 0.13$ for negative, $r = -0.7, p = 0.05$ for very negative). This implies that the louder the music, the less likely people are to make betting decisions on bad gambles. In addition, the louder the music, the lower the variance in user behavior, indicating more decisive behavior overall.

Correlation with DDM Decomposition

Finally, we were also interested in examining how the individual DDM parameters fitted for each song separately corresponded with the song features. Comparing the DDM parameters per song with the tempo, major dominance and amplitude data, we observed a statistically significant correlation between the tempo and the drift rate for very positive gambles (Figure 5.8(a), $r = -0.72, p = 0.04$), tempo and very negative gambles (Figure 5.8(b), $r = +0.79, p = 0.01$), and, interestingly, between the mean amplitude and the response caution, a connection that was also suggested in [220] (Figure 5.8(c), $r = -0.67, p = 0.06$). These observations corroborate the earlier observations made in this section, as well as those made in Section 5.8.

5.10 Summary & Discussion

In this chapter, I discussed how music-induced mood affects human decision making in the contexts of both emotional classification and risky quantitative tasks. In the first set of experiments, subjects were tasked with quickly classifying words as either positive or negative while listening to different kinds of music. In the second set of experiments, subjects were presented with gambles and needed to decide whether to accept or reject these gambles as different types of music were played to them. The results clearly indicate marked effects of music on people's decision making, which we can successfully model through the Drift Diffusion Model. This work constitutes the core of Contribution 3 of this thesis.

In the first set of experiments, discussed in Sections 5.2 - 5.5, most of the effect can be associated with the expectancy bias of the listener, implying that people's a-priori expectations regarding the emotional content of words, before observing the stimulus, are influenced by the emotional content of the music. Considering the second set of experiments, discussed in Sections 5.6 - 5.9, our results show that while there is no evidence for music-induced bias in the decision making process, music does have a differential effect on decision making behavior. Participants who listened to music categorized as happy were faster to make decisions than people who listened to music categorized as sad. Moreover, the decisions participants made while listening to happy music were consistently better than those made while listening to sad music, implying increased discriminabilty. Further analysis indicated there is a correlation between tempo and the speed and quality of decision making in this setting. Interestingly, previous work on gambling behavior has found a connection between the tempo and the speed of decision making, but was unable to isolate further impact on the quality of decision making, due to a fundamentally different design and different research questions [86]. Given that both the first and second sets of experiments use the exact same songs, the fact that such different effects were observed in the context of different tasks implies that while music has a broad effect on decision-making, the psychological mechanisms involved in emotional classification and risky analytical decision making are inherently differently affected by such external conditions.

This chapter establishes that music does indeed impact people's decision making processes in nontrivial ways, and furthermore provides evidence that this impact can be successfully modeled. Important follow-up questions, that are an integral part of this thesis and are deeply connected to Contribution 4, are whether this phenomenon extends to more complicated settings than binary decisions, and whether knowledge of how music affects people's behavior could be used by an autonomous agent to better interact with a person. The next chapter deals with both these questions towards satisfying Contribution 4 of this thesis.

6. Impact of Music on Person-Agent Interaction

In the previous chapter I established, in fulfillment of Contribution 3 of this thesis, that music indeed alters people's decision-making process in a nontrivial way, and that this effect can be modeled computationally. In this chapter I take this research thread two steps further: first to provide evidence that the impact of music on human decision-making goes beyond binary decisions; and second to show results suggesting that this effect can be modeled in real-time by a learning agent in order to induce better human-agent interaction. One of the core contributions of this thesis, described in Section 1.1, is the development of approaches and techniques for multiagent collaboration in musical environments, and specifically human-agent collaboration, constituting Contribution 4. Providing evidence that an agent can represent music as part of its world state and utilize this knowledge for better interaction is a substantial component of fulfilling Contribution 4. The other aspect of this contribution, which is the study of how multiple agents with varying tastes (which are not directly known to the other agents) can collaborate jointly, is studied in Chapter 7.

The purpose of this chapter is to study the impact of music on human decision-making and human-agent interaction in a task that requires taking into consideration the agency and intentions of other entities. To this end I present two experiments designed explicitly for this purpose. In both experiments, a person must control a car as it crosses a simulated intersection that is simultaneously being crossed by another autonomous agent, controlled by artificial intelligence. The results of the first experiment indicate that different types of music indeed have a differential effect on people's behavior in this setting. The second set of experiments provides evidence that taking the background music into account in the artificial agent's world state representation is beneficial. This chapter is based on published work done in collaboration with Corey White [223], as well as work towards an upcoming conference submission currently in preparation.

6.1 First Experiment - The Impact of Music on Cooperative Task Behavior

In this section I describe the details of the experiment conducted in this study. First, we describe the overall procedure. We proceed to describe the participants, the autonomous car behavior, the music selected for the experiment, and the data collected for analysis.

6.1.1 Procedure

In this study, participants were given control of a simulated vehicle crossing an intersection. They had three control options - speed forward, go in reverse, and brake. In addition to the human-

controlled vehicle, another vehicle, controlled autonomously by an artificial agent, was also crossing the intersection from a different direction. If the two cars collided, they would crash. Participants were instructed to safely cross the intersection without crashing. Participants were also instructed that the autonomous car would generally respect the laws of traffic but cannot be blindly relied upon to drive safely. Throughout this chapter, by "waiting at the intersection" we mean that either the person or the autonomous agent reached the intersection, then stopped for some nonzero wait time before proceeding. Each time both vehicles cleared the intersection and reached the end of the screen safely, the trial would end and a new trial would commence (a 2 second pause was introduced between trials). The experiment was divided into 8 blocks of 12 trials (for a total of 96 trials per participant). In each trial the behavior of the autonomous vehicle was randomized, determining its speed and the amount of time it would wait by default in the intersection if it had arrived to the intersection first. In each block, a different song was played, alternating between positive and negative music across blocks. The order of the songs was counterbalanced across subjects. A 3 second pause before the beginning of each block was introduced to make sure the new song had started before a new trial commenced. Each experiment lasted approximately 20 minutes. A snapshot of the experiment is presented in Figure 6.1.

6.1.2 Participants

For this experiment we originally collected data from 20 participants. All participants were graduate students who volunteered to participate in the study. Two participants were filtered out for behaving uniformly without paying attention to the experimental conditions (always going forward at the beginning of each trial without slowing, stopping or paying attention to the autonomous vehicle), leaving a total of 18 participants. Note that each participant performed the studied task under both conditions (happy vs. sad music), allowing for paired (rather than unpaired) t-test analyses. Thus, the sample size was sufficient to detect statistically significant differences in behavior between these conditions.

6.1.3 Autonomous Car Behavior

The key variability in stimuli in this experiment was presented through randomization of the autonomous car behavior. The three main aspects of the autonomous car behavior that were variable were its speed approaching the intersection, how long it would wait in the intersection before going forward if it arrived to the intersection first, and how fast it would move into the intersection and onward after entering the intersection. Participants were instructed not to blindly rely on the autonomous car's behavior, but in the scope of this experiment we opted to have the autonomous car always gave right of way if the human-controlled car made it to the intersection first. The speed of the autonomous vehicle was drawn uniformly in a range between .3 and .7 of the maximum speed allowable to the human (which was configured by screen size to guarantee that a vehicle driving continuously without stopping could reach the end of the road in 3 seconds).

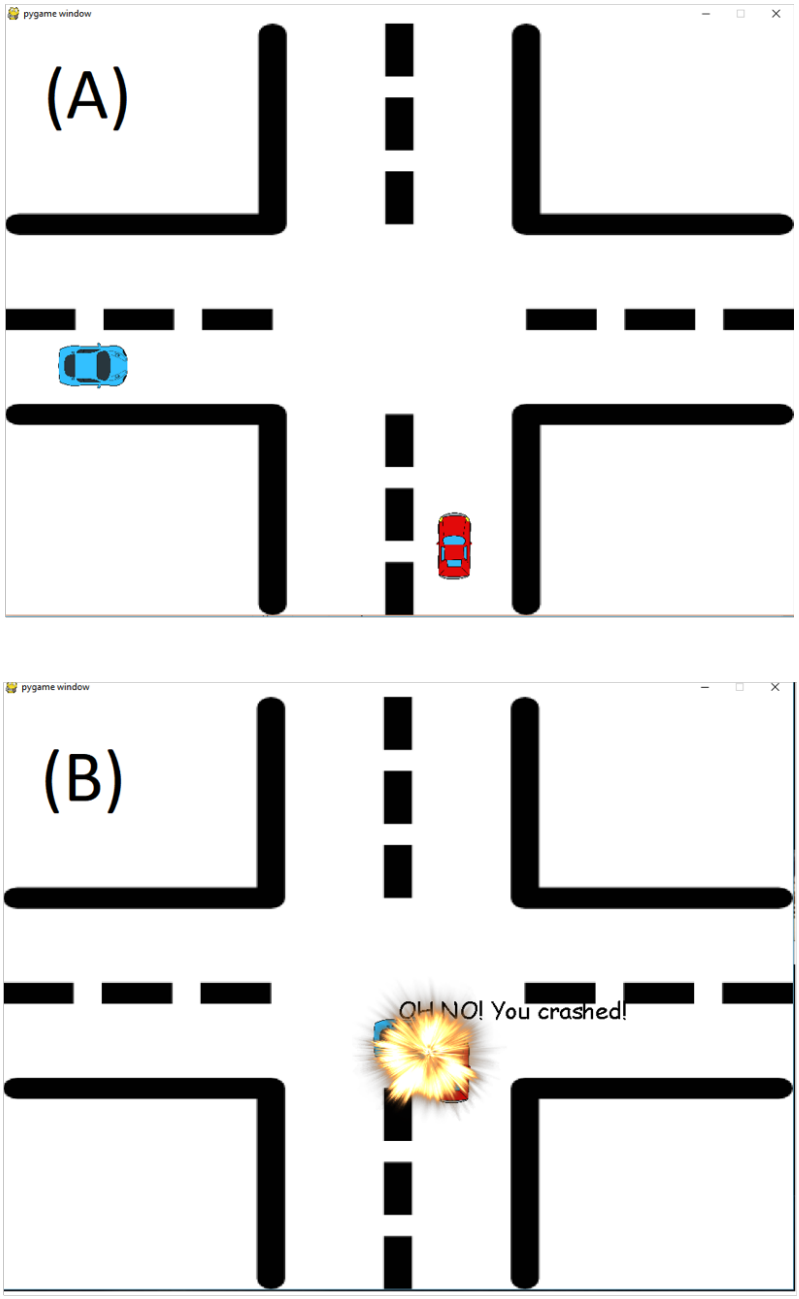

Figure 6.1: (A) A screen capture of the experiment. The red car was controlled by the participant. The blue car was controlled autonomously. (B) A collision would result in a crash, as demonstrated in this screen capture. After the crash, the trial terminates and the next trial begins.

The consequence of this behavior on the autonomous vehicle's side was that the decision whether to give right of way or move forward was almost always in the hands of the human participant. Indeed, one of the explicit goals of this study was to examine how different music-induced mood would affect people's aggressiveness vs. their inclination to give right of way.

6.1.4 Music

The music used for this experiment is the same as that used in the previous experiment described in Chapter 5, as we have established experimentally in that chapter that the selected music was effective for inducing the appropriate mood. To recap in brief, it is a set of 8 publicly available pieces curated to capture two clear music types: "sad" and "happy" music. In Chapter 5, it has been shown experimentally that the selected music was effective for inducing the appropriate mood.

6.2 Overview of Results for the First Experiment

In this section we survey the key findings of the study, examining the participants' behavior globally (that is, across all types of circumstances and autonomous vehicle behavior). We had multiple competing theories about the potential underlying impact of music-induced mood on decision-making in this context. Would happy music make people more easygoing and therefore collaborative? Or would sad music make people more apprehensive and cautious? Indeed, the results, which I will now describe at length, reveal a complicated picture, as it seems music impacts more than one component of decision-making in this setting.

6.2.1 Minimal Distance from Autonomous Car

The most statistically significant difference ($p < 0.05$ using a paired t-test) across all trials was that participants listening to sad music kept a lower minimal distance overall from the autonomous car compared to when they were listening to happy music. In other words, their behavior when listening to sad music was riskier and less considerate ("cutting it closer" with respect to how much margin for error they kept when entering the intersection). This result is presented in Figure 6.2.1.

6.2.2 Driving Speed

Participants also differed in their driving speed in the sad and happy music conditions (significant at $p < 0.05$ using a paired t-test). Overall, participants were more likely to go fast in the happy music condition compared to the sad music condition, as reflected in Figure 6.3.

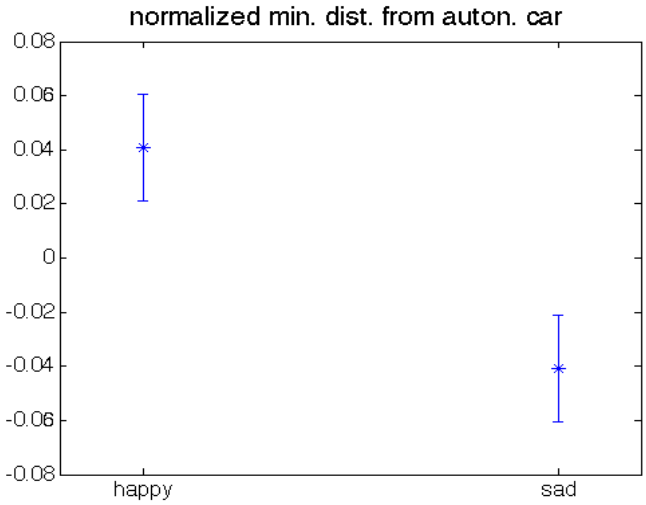

Figure 6.2: Normalized minimal distance kept from the autonomous car by the participants in the sad and happy music conditions (here and elsewhere, bars represent std. error). Participants tended to keep a lower minimal distance when listening to sad music.

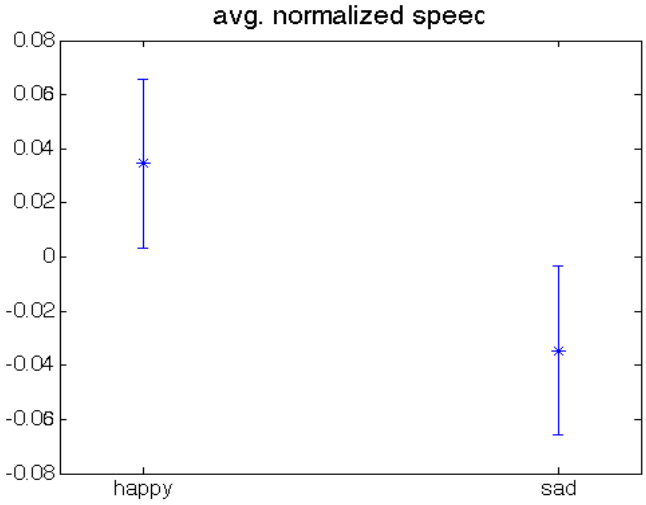

Figure 6.3: The average normalized speed of the participants in the happy and sad music conditions. Participants were more likely to go faster when listening to happy music.

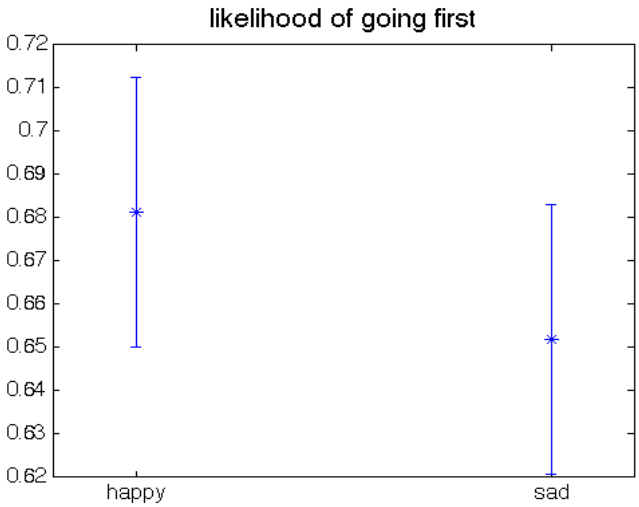

Figure 6.4: The likelihood of the participants to go first into the intersection in the sad and happy music conditions. Participants were more likely to go first when listening to happy music.

6.2.3 Right of Way

Another difference, which is strongly related to the previous observation, and is borderline significant[11] (at $p < 0.1$ using a paired t-test) was that participants listening to happy music were more likely to go into the intersection first compared to when they were listening to sad music, as depicted in Figure 6.4.

6.3 Breakdown of User Behavior Under Different Trial Conditions In the First Experiment

In this section we consider how different music induced different participant behavior when breaking down the trials by the different types of autonomous car behavior.

6.3.1 Behavior under Different Autonomous Car Intersection Wait Times

If we compare how participants behaved when the autonomous vehicle waited < 4 seconds at the intersection, compared to cases in which the autonomous vehicle waited 4 seconds or more, we observe different patterns of behavior. Specifically, the differences in speed and wait times between the happy and sad conditions are quite different in cases in which the autonomous vehicle waited < 4 compared to cases in which it waited > 4, suggesting that the impact of music cannot be decoupled

[11] A 0.1 threshold for testing the significance of p-values is accepted in the context of relatively small samples sizes. Nonetheless, we strive to use these measures responsibly in our choice of language, thus using the equally common term "borderline significance" to describe results with p-value < 0.1 but > 0.05

from the experimental circumstances, and specifically from autonomous vehicle behavior. While the differences between participant behavior in the case that the autonomous vehicle waited < 4 vs. when it waited the autonomous $>= 4$ are highly statistically significant ($p < 0.01$ with an unpaired t-test), the differences between the happy and sad conditions once imposing this partition are no longer statistically significant.

6.3.2 Behavior under Different Autonomous Car Average Speed

A similar related trend to that observed in the previous section was observed when considering the average speed of the autonomous car. In trials in which the average speed of the autonomous vehicle was above the median, people were slower to drive and took longer to wait at the intersection while listening to sad music, compared to when listening to happy music (again with $p < 0.05$ using an unpaired t-test).

6.4 Impact of Musical Parameters on User Behavior in the First Experiment

The partition between "positive" and "negative" mood-inducing songs is easy to understand intuitively, and in itself is enough to induce the different behavioral patterns discussed in the previous section. However, similarly to the analysis performed in Chapter 5, we are interested in finding a deeper connection between the behavior observed in the experiment and the different characteristics of music. More exactly, we are interested in finding the correspondence between various musical features, which also happen to determine how likely a song is to be perceived as happy or sad, and the driving decision-making manifested by participants. To this end, we considered the 8 songs used in this experiment, extracted key characterizing features which we assume are relevant to their mood classification, and examined how they correlate with the subject behavior we observed.

6.4.1 Extracting Raw Auditory Features

We focused on four major auditory features: a) overall tempo; b) overall "major" vs. "minor" harmonic character (I again refer to this feature as "major chord ratio" for simplicity); c) average amplitude, representing overall loudness; and d) maximum amplitude, representing peak loudness. Features (a), (c) and (d) were computed using the Librosa library [332]. To compute feature (b), we used the same procedure described in Section 5.5.1. As in Chapter 5, analysis done in [220] confirms these features are indeed associated with our identification as "positive" vs. "negative". Having labeled "positive" and "negative" as 1 and 0 respectively, a Pearson correlation of $0.7 - 0.8$ with p-values ≤ 0.05 was observed between these features and the label.

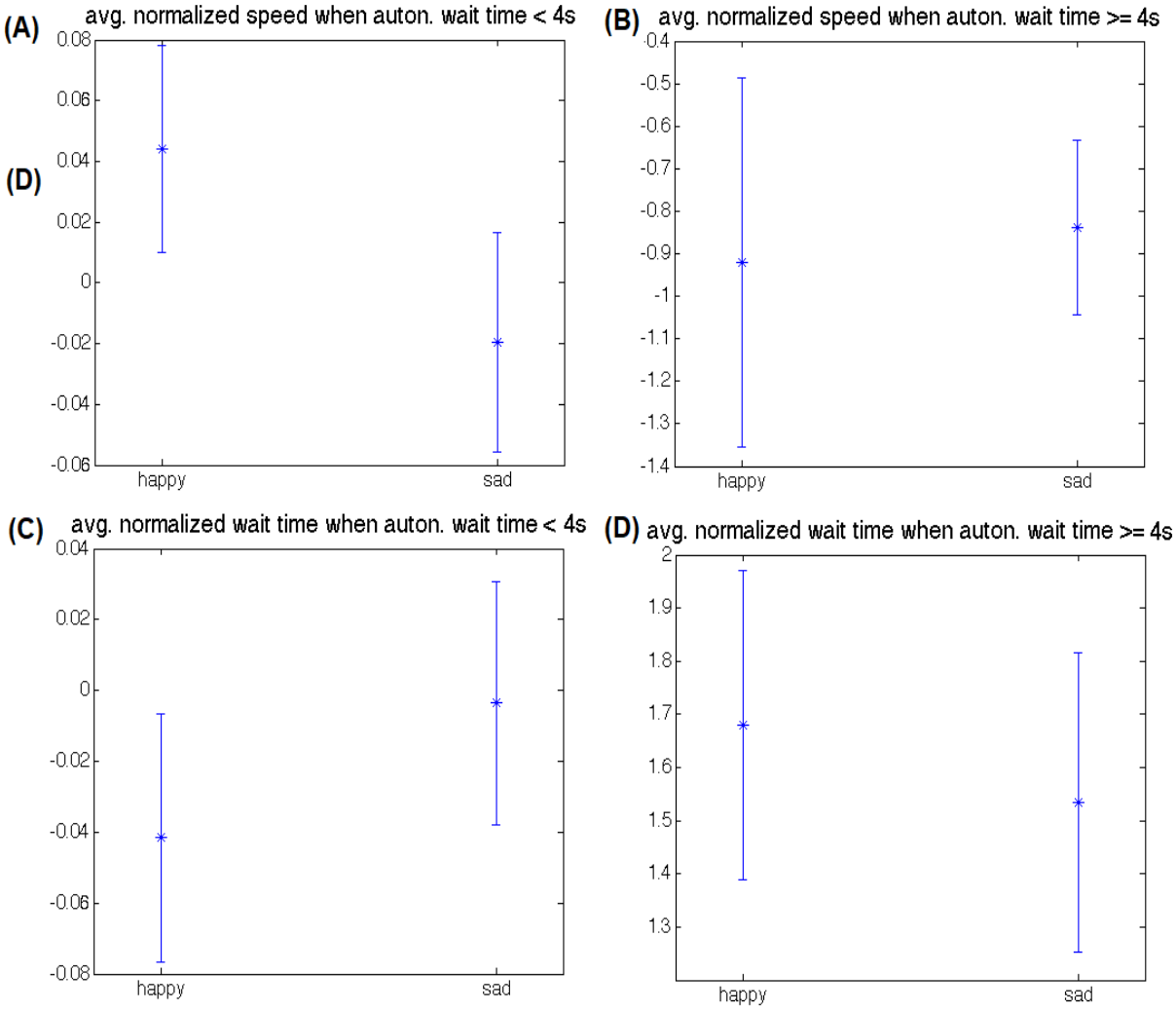

Figure 6.5: (a) Normalized average per-trial speed of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited less than 4 seconds. (b) Normalized average per-trial speed of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited more than 4 seconds. (c) Normalized per-trial time waiting at the intersection of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited less than 4 seconds. (d) Normalized per-trial time waiting at the intersection of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited more than 4 seconds.

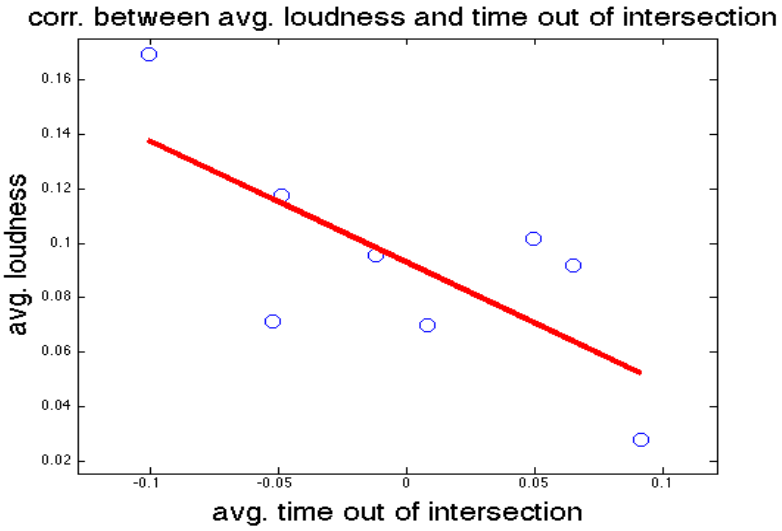

Figure 6.6: Correlation between the average loudness of the music and the normalized total time out of the intersection for the participants.

6.4.2 Results

Overall, the most prominently influential aspect of the music as observed by statistical analysis is the loudness of the music. Additional effects were observed relating to tempo and major chord ratio, but they did not meet the same criteria for significance.

6.4.3 Loudness and Overall Time Out of Intersection

The normalized overall time out of intersection is the total time it took the participant to drive up to the intersection, wait, and cross the intersection, normalized per subject. The normalized time out of the intersection was statistically significantly ($p < 0.05$) inversely correlated with both the average loudness ($r = -0.72$) and the maximum loudness ($r = -0.77$) of the music. The correspondence between the average loudness and the overall time out of intersection is presented in Figure 6.6 (the findings for the maximum loudness are similar). In other words, the louder the music was, the faster people were to complete the task.

6.4.4 Loudness, Speed, Time Stopped, and Minimal Distance

Loudness also impacted various aspects of participant behavior that are related to the participants' driving speed and overall aggressiveness. These results are borderline significant at $p < 0.1$ for all correlations reported in this subsection.

- Most straightforwardly, the average loudness was positively correlated ($r = 0.65$) with the normalized average speed of the participants, meaning that participants drove faster when

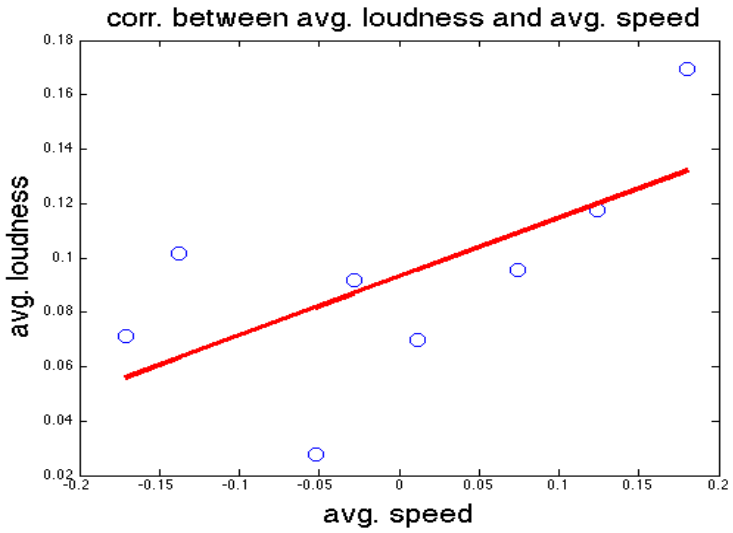

Figure 6.7: Correlation between the average loudness and the average speed of the participants.

listening to louder music. This result is depicted in Figure 6.7.

- Similarly, other metrics reflect overall speed, including the minimum speed, the median speed and the initial speed (speed after 1 second from the beginning of the trial) were positively correlated with $r > 0.6$.

- The overall normalized time the participants stopped at the intersection was inversely correlated at $r = -0.67$ with the average loudness, meaning people were faster to continue into the intersection when listening to louder music. This finding is presented in Figure 6.8

- Lastly, the minimal distance the participants kept from the autonomous car was positively correlated with the average loudness, meaning the louder the music was, the higher the minimal distance was. Considering the other findings in this section and the fact that the minimal distance and the average speed are positively correlated at $r = 0.75$ (and $p < 0.05$), it is reasonable to assume this relationship is a result of the impact of loudness on the participants' speed rather than an indication of how loud music increases people's risk aversion, for instance.

6.4.5 Tempo and Hesitancy

The total number of key presses per trial, normalized per participant, is a good proxy for hesitancy in decision making (speeding and slowing down, going forward and braking, etc). Interestingly, the key press count was inversely correlated to the tempo ($r = -0.59$ and $p < 0.1$), suggesting faster music reduced people's hesitancy. This result is presented in Figure 6.9.

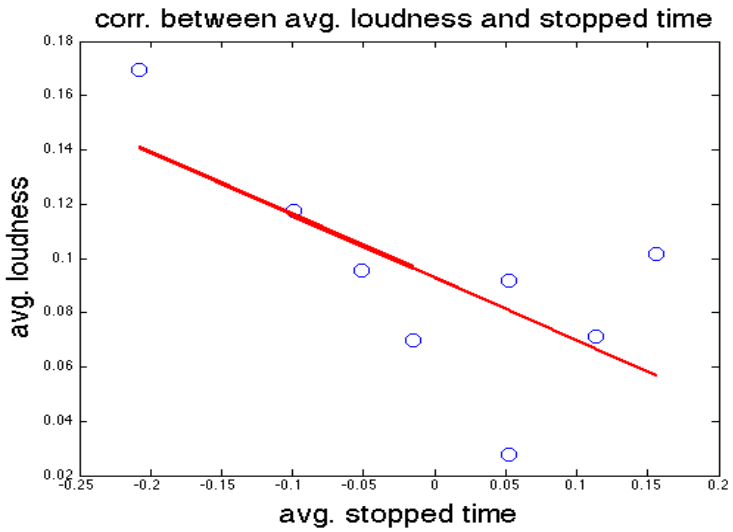

Figure 6.8: Correlation between the average loudness and the average time the participants stopped at the intersection.

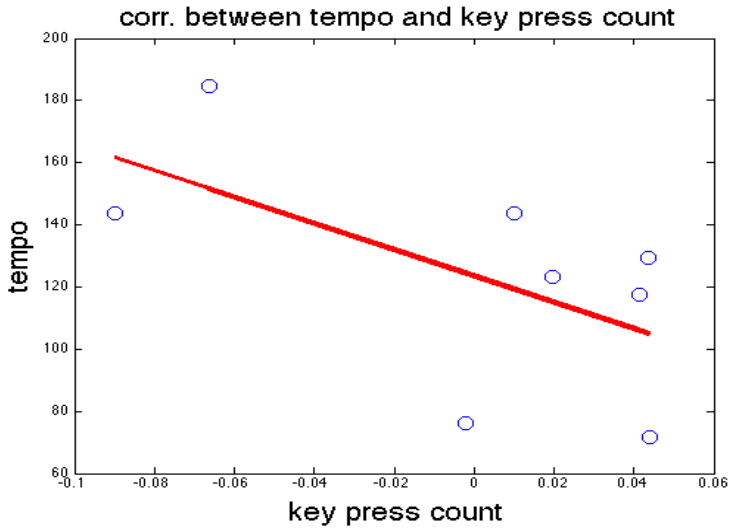

Figure 6.9: Correlation between the normalized key press count of the participants and the tempo.

6.4.6 Additional Observations

Beyond the results reported thus far in this section, several relationships between musical features and participant behavior were observed that did not meet the $p < 0.1$ criterion for significance, but came sufficiently close to merit mention:

- The normalized key press count was also inversely correlated with the major chord ratio (at r = -0.52), implying it is possible that music that leans heavier towards major harmonies also reduces hesitancy in the participants.
- The major chord ratio was also positively correlated with the maximum speed of the participants, and the minimal distance the participants kept from the autonomous car, at r = 0.54 and r = 0.52, respectively.
- The tempo was positively correlated with both the average and the max speed at r = 0.53 for both.

Thus far in this chapter I focused on measuring whether music has an effect on people's decision-making in the context of a task which requires reasoning about the intentions of another agent. Having gathered evidence that such an effect does exist, I will now discuss a follow-up experiment that was designed to study whether knowledge of background music can help the *autonomous agent* perform better in the context of a similar task.

6.5 Second Experiment - Introducing a Learning Agent

Indeed, the autonomous agent in the experiment described in Sections 6.1 - 6.4 did not, in fact, display all that much agency in general. It would always give right of way to the human driver if the person reached the intersection first, and deciding on its speed and intersection wait times uniformly at random in a rather conservative range implied that it was, on average, considerably slower and less aggressive than the participants. In the second experiment, the agent's behavior was fundamentally changed, and replaced with a learning agent that was motivated by a goal similar to that of the subjects - to reach the other side of the intersection as quickly as possible without crashing.

6.5.1 Procedure

The overall experimental design was very similar to that described in 6.1. As in the first experiment, participants were given control of a simulated vehicle crossing an intersection and had three control options - speed forward, go in reverse, and brake. Participants were instructed to do their best to reach the other side safely, and were warned the autonomous agent's behavior could be unpredictable. An additional modification from the previous experiment was that at the end of

each trial, the task completion times (i.e. time to reach the end of the intersection) of both the human participant and the autonomous agent were shown.

Each time both vehicles cleared the intersection and reached the end of the screen safely, the trial ended and a new trial commenced (a 3 second pause was introduced between trials). The experiment was divided into 16 blocks of 12 trials (for a total of 192 trials). In each block, a different song was played, alternating between positive and negative music across blocks. The order of the songs was counterbalanced across subjects and blocks. A 3 second pause before the beginning of each block was introduced to make sure the new song had started before a new trial commenced. Each experiment lasted approximately 30 minutes.

6.5.2 Participants

For this experiment we collected data from 22 participants. All participants were graduate students who volunteered to participate in the study. It is useful to note that in this experiment the focus was on the behavior of the autonomous agent, rather than the human participants, a fact which allowed us to aggregate data across participants, allowing for larger samples per condition compared to the previous experiment. In effect, the participants are considered in this section as draws from a stochastic environment, as far as the agent is concerned.

6.5.3 Music

The music used for this experiment is the same as that used in the previous three experiments described in Chapter 5 as well as in Sections 6.1 - 6.4 of this chapter. This choice was primarily justified by the fact that these experiments already established that the selected music was effective for inducing the appropriate mood, and was thus suitable for this experiment as well.

6.5.4 Autonomous Vehicle Behavior and Learning Architecture

The biggest departure from the experiments described in Chapter 5 and in sections 6.1 - 6.4 is the behavior of the autonomous counterpart. In this experiment, the focus was on whether an agent can learn a specialized policy that takes the background music into account in a way that helps it achieve its goals. To that end, during the first half (8 blocks of 12 trials, 96 trials total) the autonomous agent is randomly exploring while learning two models simultaneously - one that takes the background music into account (music-aware), and one that does not (music-unaware). The second phase of the experiment was partitioned in two. In one half of the exploitation phase (48 trials), the agent utilized the music-unaware model to make its decisions. In the other half, the agent utilized the music-aware model. The order of the two halves was counterbalanced across subjects.

State-Action Representation

Given that navigating through the intersection is an inherently sequential task, such that decisions at one timestep affect the agent's state in the next, the learning framework for the agent is that of reinforcement learning (RL). To recap in brief, sequential decision-making tasks are suitably formulated as a Markov Decision Process (MDP) [374]. Formally, an episodic MDP is a tuple (S, A, P, R, T) where S is the set of states; A the set of actions, $P : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function where $P(s, a, s') = r$ denotes the probability of transitioning from state s to state s' when taking action a. $R : S \times A \rightarrow \mathbb{R}$ is the state-action reward function, where $R(s, a) = r$ means that taking action a from state s will yield reward r. T is the set of terminal states, which end the episode. A more in-depth of reinforcement learning as a paradigm is presented in Chapter 2.

In the context of the intersection crossing car experiment, the state space S is comprised of the autonomous car position on the x axis; the participant-driven car position in the y axis; the speed of the two vehicles; and the in-trial elapsed time. On top of these 5 features, 3 more features were added to the state representation to simplify learning: a binary feature for whether the autonomous vehicle reached the end of the screen; a binary feature for whether the human participant has reached the end of the screen; and a binary feature indicating whether the two vehicles crashed. Ultimately, this state representation leads to an 8-dimensional vector. The music-aware model added one more feature, a binary flag denoting whether the background music was from the "happy" or the "sad" song pool, leading to an 9-dimensional state representation.

To simplify and standardize the decision-making process, the agent made decisions at 3 decision points: right at the start of the trial, once upon reaching the mid-point towards the intersection, and once upon reaching the intersection. If the agent stopped, it would make an additional decision after waiting for a randomized duration of 3-5 seconds. A visual illustration of the decision points is provided in Figure 6.10

The action space A comprised three possible actions at each decision point - $\{\textbf{FAST}, \textbf{SLOW}, \textbf{STOP}\}$. An episode began with both vehicles at the starting position and ended when both vehicles reached the end (a terminal state was added at the end of each trial).

Reward Function and Learning Architecture

As previously mentioned, the goal of the agent was to reach the other end as quickly as possible without crashing. For that reason, the reward function was designed to promote expedient completion of the task, but punish the agent severely for crashing. For a completion time t and a binary variable $crashed \in \{0, 1\}$, the reward r for an episode is $r = -t - crashed \cdot 100$ (effectively in this case maximizing reward is tantamount to minimizing penalty).

The learning model utilized was DQN, a deep-learning variant of Q-learning, a canonical reinforcement learning technique [267]. In Q-Learning, given a state-action transition $\langle s, a, r, s' \rangle$,

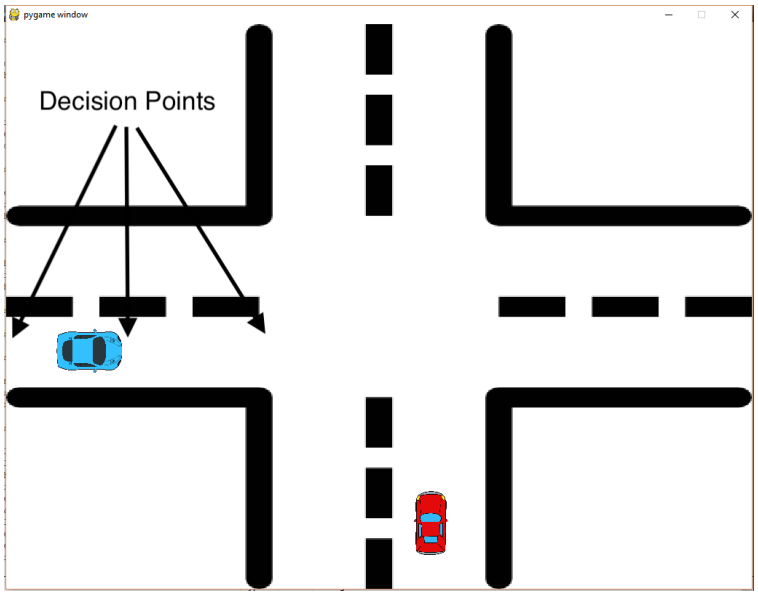

Figure 6.10: Decision points for the autonomous vehicle. At each point, the choices for speed are **FAST**, **SLOW**, and **STOP**.

meaning that the agent has taken action a from state s yielding reward r and leading to new state s', the expected utility of taking an action a from state s is learned through the update step $Q(s,a) = r + \gamma \cdot max_{a'}(s', a')$, with γ being the discount factor (the discount factor for the autonomous agent in this experiment was set to $\gamma = 0.9$). In DQN, the Q function is approximated using a deep feed-forward neural network in a fashion similar to that used by Maicas et al. [241]). In the implementation used in this experiment, a network used consisted of an input layer of sizes 8 or 9 depending on whether it was the music-aware or music-unaware model, two fully connected layers of 32 neurons each, and an output layer of size 3, with one output per action predicting its relative Q -value.

Since learning from limited experience in real time is impractical, the learning models utilized experience replay, a common practice in deep RL learning architectures [267]. At the end of each trial a random sample of 20 trials (with replacement) was drawn uniformly from the aggregated trial history thus far, and the two models were trained repeatedly for 100 iterations. To further speed up learning the discounted reward was back-propagated from the end of each episode (i.e. trial) to all the $\langle s, a, s' \rangle$ elements of an episode of length T, i.e. for the i-th $\langle s, a, s' \rangle$ tuple in the episode, $r = \gamma^T - i \cdot r^T$.

In the next section I will describe the results of this experiment.

6.6 Results of the Second Experiment

In this section I will describe the key finding of this experiment: utilizing a music-aware model reached better performance than the music-unaware model with respect to average task completion time, while maintaining a crash rate at least as low as that of the music-unaware model.

As a first step, we confirmed that the data collected in this study replicated the results reported in Section 6.2.

6.6.1 Establishing the Impact of Music on Human Behavior

Fortunately, in this experiment we were able to once again produce evidence that people's decision-making was differentially impacted by the type of music played. We base this conclusion on the data collected in the first half of all sessions in the second experiment, in which the agent was randomly exploring uniformly and independently of the music stimulus.

Even without any preprocessing, by just considering the aggregated completion times separated by the music condition and pooled together across all participants, one can observe a statistically significant difference between people's overall driving speed and hesitancy in the happy vs. sad music condition. This result is presented in Figure 6.11. These results are significant using an unpaired t-test with $p < 0.05$.

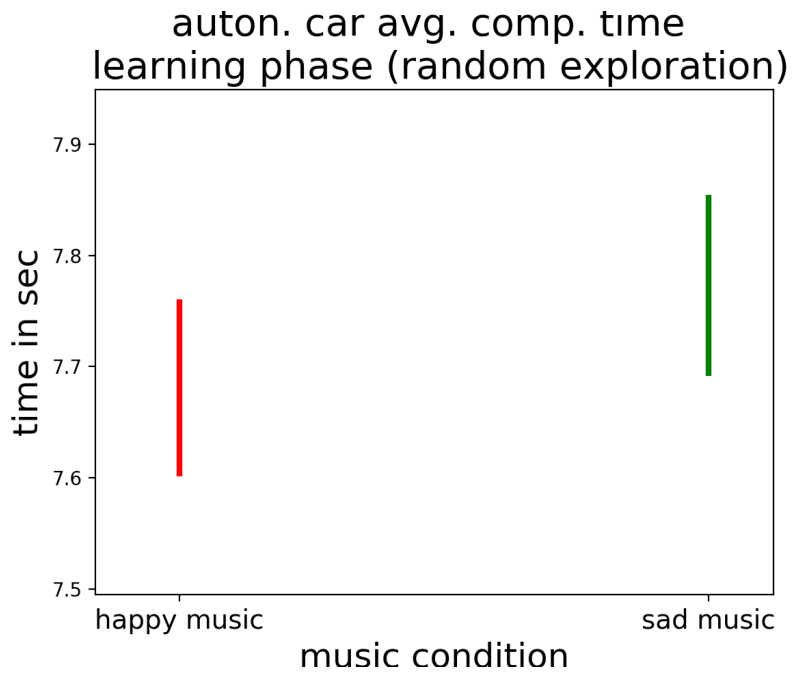

Figure 6.11: Unprocessed average completion times for the participants in happy music exploration vs. sad music exploration. Error bars represent the standard error. The results are significant using an unpaired t-test.

Once we normalize the completion times per subject, aggregate per-condition averages per condition subject, then aggregate across all users, we get an even more significant separation between the two music conditions, as indicated in Figure 6.12.

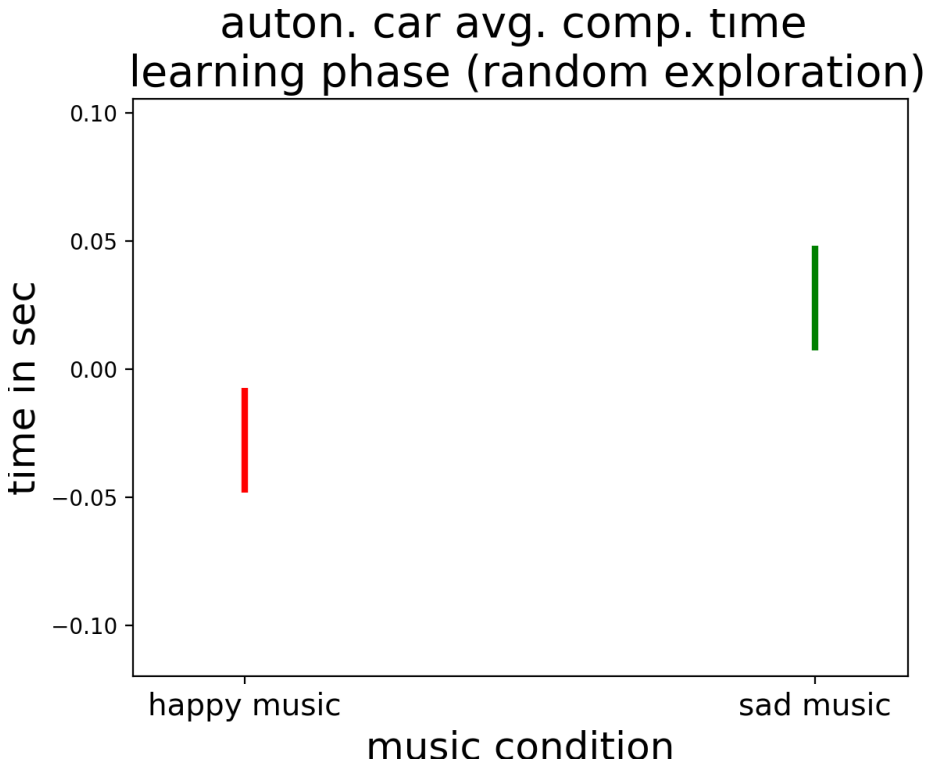

Figure 6.12: Average completion times for the participants in exploration vs. happy music exploration vs. sad music exploration, normalized per user and with means aggregated across all users. Error bars represent the standard error. Results are significant using a paired t-test.

This result confirms that indeed different types of music impacted how people approached the task in a why consistent with our findings in the first experiment described in this chapter 6.2. The question of whether the learning agent was able to leverage this differential effect is addressed in the next subsection.

6.6.2 Impact on Average Autonomous Agent Completion Time

The first objective was to compare the actual completion times of the learning agent in the three stages of the experiment - exploration, exploitation without music, and exploitation with music. The results of this analysis, presented in Figure 6.13, first show that both learned models improved over relative exploration, implying they had successfully learned some signal of the other person's intentions. Second, and more importantly, the results show that the music-aware model did significantly better than the music-unaware model, providing evidence that knowledge of the background music is indeed helpful to the learning agents. Interestingly, if we analyze the relative performance

of the two models in the two background music conditions, i.e., happy vs. sad music, we can clearly see the difference in performance is much greater in the sad case, implying that the music-aware model was able to take better advantage of the participants' slower-on-average driving under this condition. This observation is presented in Figure 6.14.

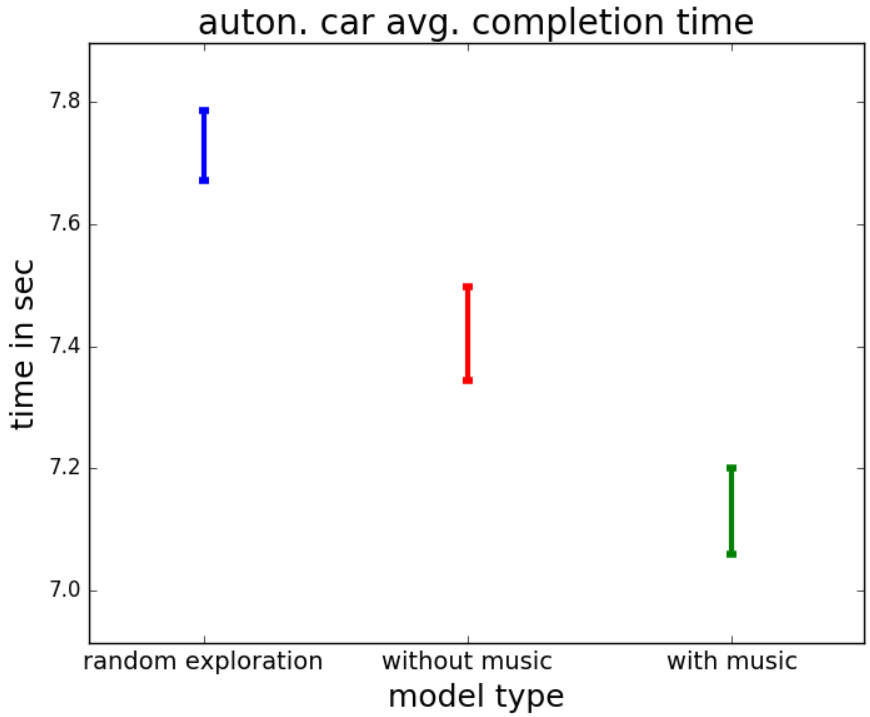

Figure 6.13: Average completion times for the autonomous vehicle in exploration vs. music-unaware expoitation vs. music-aware exploitation. Error bars represent the standard error.

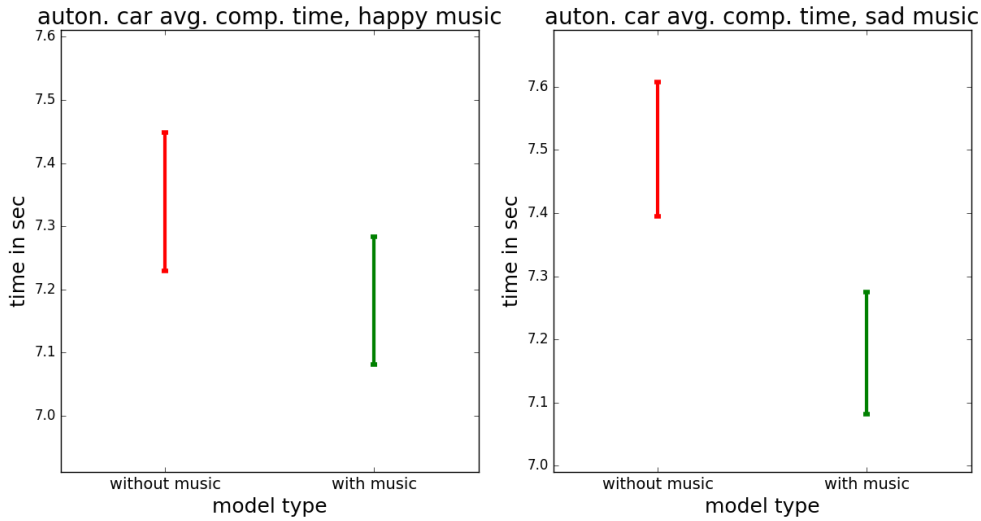

Figure 6.14: Average completion times for the autonomous vehicle in exploration vs. music-unaware expoitation vs. music-aware exploitation, split by music condition (happy music to the left, sad music to the right). Error bars represent the standard error.

Indeed, if we consider the differential effect of the background music on driving speed in the two learned models, we can see that while the music model leads to faster driving by the autonomous vehicle on average, the effect is statistically significantly greater in the sad music condition relative to the happy music condition. This result is highlighted in Figure 6.15.

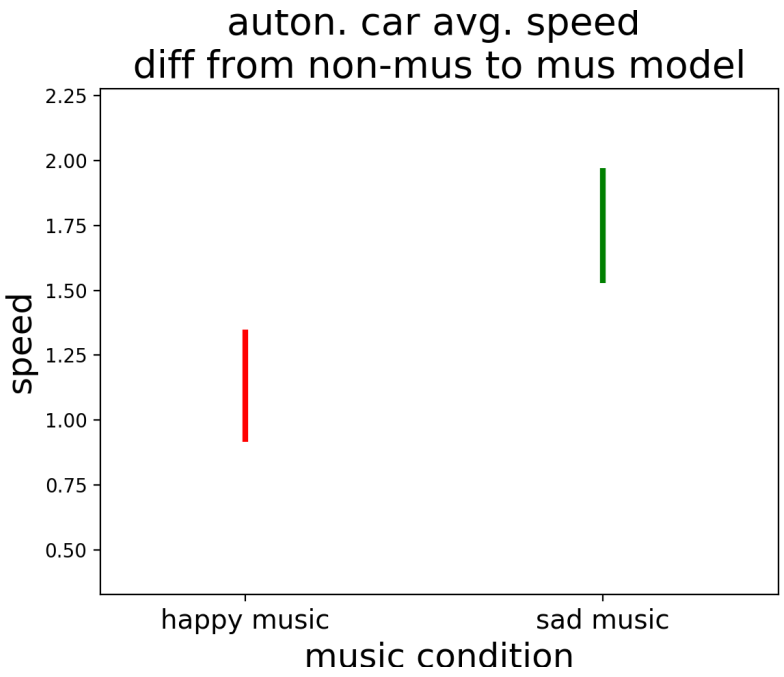

Figure 6.15: Difference in avg. autonomous driving speed in the music-unaware vs. music-aware case. The difference is significantly greater for sad music. Results were statistically significantly greater using an unpaired t-test.

6.6.3 Impact on the Crash Rate

One obvious caveat when claiming one model did better than another with respect to average completion time is that if one model behaves more recklessly, it would make it substantially more crash-prone. The goal was to have the agents learn how to reach the other side quickly, but also *safely*, meaning it had to take the human participant into account. For this purpose we also examined the average crash rate in the three phases of the experiment, i.e. exploration, exploitation without music, and exploitation with music. Though the results are not statistically significant, they are sufficient to assuage the suspicion that the music-aware model did better than the music-unaware model with respect to average completion time merely by driving recklessly (if anything, the results seem to suggest an opposite trend). The comparison of crash rates in the three phases is presented in Figure 6.16.

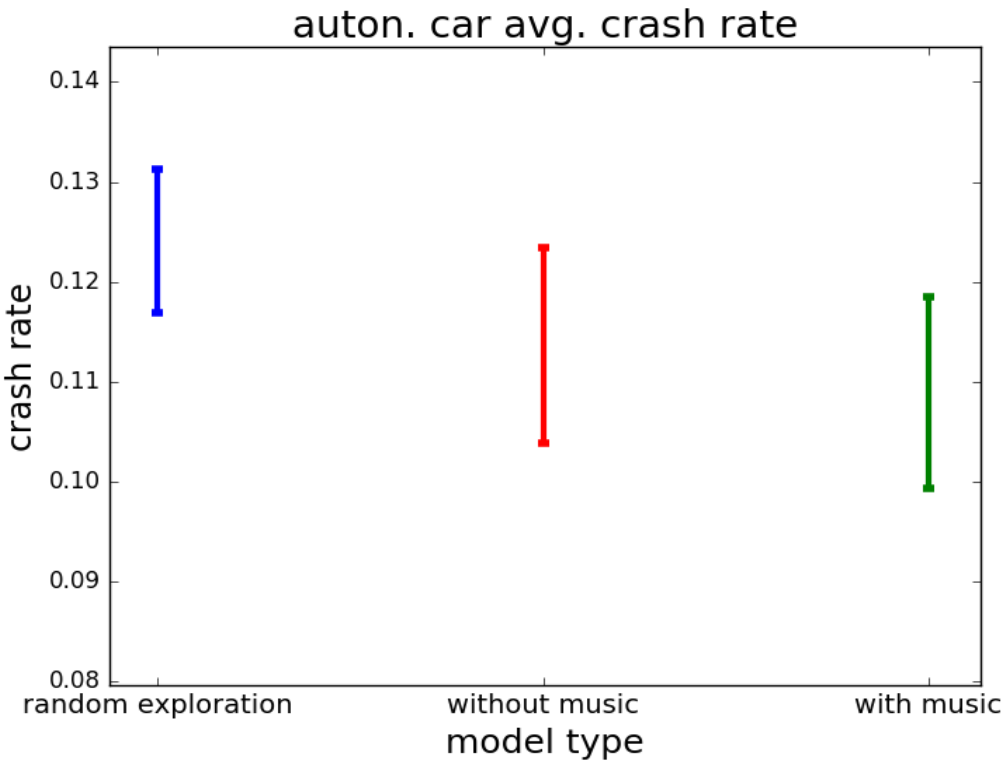

Figure 6.16: Average crash rates for the autonomous vehicle in exploration vs. music-unaware expoitation vs. music-aware exploitation, split by music condition (happy music to the left, sad music to the right). Error bars represent the standard error.

6.7 Summary & Discussion

The purpose of this chapter was to connect the findings presented in Chapter 5 towards satisfying Contribution 3 of this thesis – namely, that music does have an impact on human decision-making, and that this impact can be modeled – with a more advanced level of autonomous agency, towards engendering person-agent and agent-agent cooperation in musical environments, satisfying Contribution 4 of this thesis.

To this end, in this chapter I presented two separate experiments. In the first experiment I analyzed how people's decision-making behavior was affected by music in the context of a social task which required a certain level of cooperation to avoid adverse consequences. Participants were required to drive a simulated car through an intersection while another car, controlled by an autonomous agent, was also crossing from a different direction. Examining the results reveals a complex picture befitting the subtleties of the performed task. While happy music induced some aspects of behavior that could be described as more social, namely that participants kept a safer distance from the other car when crossing, they also manifested less social behavior by driving

faster and being less likely to let the autonomous vehicle go first. All in all the conclusions of this study were that music did affect people's behavior, but in a nontrivial and multifaceted way.

In the second experiment, I studied whether an agent could leverage knowledge of background music to better perform with respect to the intersection-crossing task presented in the first experiment. To this end, a learning agent was introduced in lieu of the simpler preprogrammed agent utilized in the first experiment. This agent spent the first half of the experiment gathering data and learning two separate behavior models - one which included the type of music in the world state representation, and one that did not. In the latter half of the experiment, the agent utilized both models to decide on its actions. Results of this study indicated that indeed utilizing background music as part of the world representation helped the agent improve its performance in terms of completion time, without increasing the risk of crashing. This result establishes that person-agent interaction in the context of music is possible and potentially beneficial, in partial fulfillment of Contribution 4 of this thesis. In Chapter 7 I will expand on this goal from a different direction, that of multiagent music collaboration in the face of different and potentially conflicting initial preferences.

In the next chapter, I will connect the notion of individual preferences (which was at the core of Chapter 3), reasoning about other agents' intentions in musical contexts (which lay at the core of chapter 6), and multiagent collaboration, adding both to Contribution 4 by further fleshing out possibilities of multiagent collaboration in music-related tasks, and to Contribution 6 by showing the applicability of the proposed approach in other domains.

7. Multiagent Collaboration Learning: A Music Generation Test Case

In Chapter 5 I discussed how the impact of music on human decision-making can be modeled. Subsequently in Chapter 6 I discussed how this impact can be leveraged by an agent to engender better interaction with a person. However, that is only one facet of person-agent interaction in musical context. The other includes a scenario in which people and machines actively collaborate in music generation. What would such an interaction be like? An important aspect of person-agent interaction, or of agents interacting with multiple people and/or other agents, is that of reasoning about *preferences*. Particularly in a domain such as music generation, people's subjective tastes play a pivotal role, and reasoning about them when trying to collaborate is critical. This train of thought leads to a deeper question: how can multiple agents reason with each other in a shared task *while also maintaining individual preferences* that may be at odds with the shared task and with the preferences of others? Studying the balance between shared tasks and individual preferences in multiagent interaction is a significant step in fulfillment of Contribution 4 of this thesis, building towards multiagent music interaction as a meaningful step towards person-agent music generation.

This question is of broader interest in the context of *Multiagent Reinforcement Learning*. As reinforcement learning becomes more prominent, the subfield of Multiagent Reinforcement Learning (Multiagent RL) [51] – i.e., studying systems in which multiple learning agents interact with one another in both cooperative and competitive situations – becomes of great interest as well. One specific research direction is the study of Multiagent RL is ad hoc teamwork [18, 371]. In this setting, multiple agents need to cooperate on a task without prior coordination or communication. However, in the ad-hoc teamwork setting it is typically assumed that the participating agents prioritize the shared task completely, and that they have no intrinsic preferences beyond maximizing task performance.

In contrast, in this chapter I consider the scenario in which agents that work together on a shared task also have their own preferences, or individualized reward signals. These agents then have to cooperate or adjust their policies to accomplish the shared task, but they may also still pursue the maximization of their individualized reward signals, to a varying and tunable degree. Indeed, an obvious example of such a real world setting is playing in music ensembles. Each musician learns to play their instrument independently, and typically has his or her own aesthetic preferences regarding how their instrument should be played and what they'd like the shared music to sound like. However, when performing together, musicians also have to harmonize and coordinate so that they produce music that is pleasing to their audience, who serve as a shared and extrinsic reward signal. Even when playing for the first time, good musicians are able to do so despite their

different initial individual preferences, and as this chapter suggests, perhaps even partly as a result of them. However, this relationship between individual preferences and shared task performance is by no means restricted to musical domains. In this chapter, I will present evidence for this fact by demonstrating the applicability of the findings in the music domain to a canonical multiagent RL environment, the pursuit domain (which I refer to as "predator-prey" for simplicity). Establishing the generalizability of observations made in musical contexts in other, non-music domains further goes towards the fulfillment of Contribution 6 of this thesis.

In this chapter I first propose a multiagent RL paradigm that explicitly captures the interplay between intrinsic reward signals (reflecting individual preferences) and a shared task reward. Second, I propose a training regimen specific to this scenario. Lastly, I study how different blending proportions for intrinsic and extrinsic reward impact agent learning towards the shared task signal, as well as their individualized preference, in two separate domains: multiagent chord generation, and predator-prey. The results presented in this chapter indicate that certain blending strategies are in fact beneficial for shared task performance. Indeed, the experimental results presented in this chapter suggest that such a multi-objective reward does not hinder the agent from performing well on the shared task, if the agents balance their internal reward signal and the shared reward with the appropriate, agent-specific balancing factor. In fact, with the right balance, these agents are able to leverage their preferences to do better at the shared task than if they did not have the internal reward signal. Surprisingly, this result holds true even if the agents use disparate preference signals, or if their individual balancing factors are different from one another.

This chapter is based on work done in collaboration with Ishan Durugkar.

7.1 Background

In this section I introduce the learning techniques that are used in this chapter as part of the methodology for training agents to balance their preferences with the shared objective.

7.1.1 Reinforcement Learning

As in Chapters 3, 4 and 6, the learning framework used in this chapter is that of reinforcement learning. To recap in brief, Reinforcement Learning (RL) considers an agent acting in a Markov Decision Process (MDP), a mathematical framework for modeling sequential decision-making [374].

An MDP is defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma >$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $\mathcal{T}$ is the transition probability, $p(s'|s,a)$, where $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. $R(s,a)$ is the reward for taking action a in state s. The discount factor $\gamma \in [0,1)$ specifies how much to discount future rewards.

Reinforcement learning aims to maximize the expected discounted future reward $\mathbb{E}_\pi \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$, called the returns.

π is a stochastic policy that specifies the probability of taking an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. To maximize the expected returns, one possible technique is to to optimize the policy directly.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Policy Gradient approaches use the reparameterization trick to allow gradient descent to optimize the policy [375], and Actor Critic algorithms estimate the value of states using a parameterized critic to reduce the variance of these updates.

In this study we used one such policy gradient technique, PPO [354], to learn the policies of our agents. PPO updates the policy by minimizing the divergence of the new policy with respect to its current policy. It does this by using conservative approximations of the Trust Region [353].

7.1.2 Multiagent Reinforcement Learning

In the multiagent setting, we consider a fully observable environment with K agents acting simultaneously. Most of the MDP formulation for Reinforcement Learning with a single agent carries over to this setting. The action space is modified to $\boldsymbol{\mathcal{A}} \equiv \mathcal{A}^K$. At each time step, all agents take their actions, and the joint action vector $\boldsymbol{a}_t = [a_{1,t}, a_{2,t}, \ldots, a_{K,t}]$ acts on the state to produce the transition to the next state and reward.

We consider a cooperative setting where the environment's reward R_e is shared among all the agents. In this study we focused on the decentralized learning scenario, in which each agent's policy is learned and updated separately by each individual agent.

In order to train the multiple agents to accomplish this task, we provide this shared reward to all the agents and the agents individually use policy gradient (PPO) to improve their policies. The simultaneous change of policies might destabilize the game, but frequent updates with a small learning step using the PPO algorithm allow learning to proceed in a fairly stable manner.

7.2 Methods

In this section I introduce the methodology we used in training agents according to their individualized preferences, and balancing these with the shared objective they have. This methodology can be used in order to provide a richer training signal for the agents, or it can be used as a technique for ad hoc teamwork, in a setting in which agents need to quickly adjust their existing strategies, which may have been learned to solve a different problem altogether, to coordinate on and complete a shared task without training jointly beforehand.

Figure 7.1 illustrates this idea of individualized reward balanced with a shared objective.

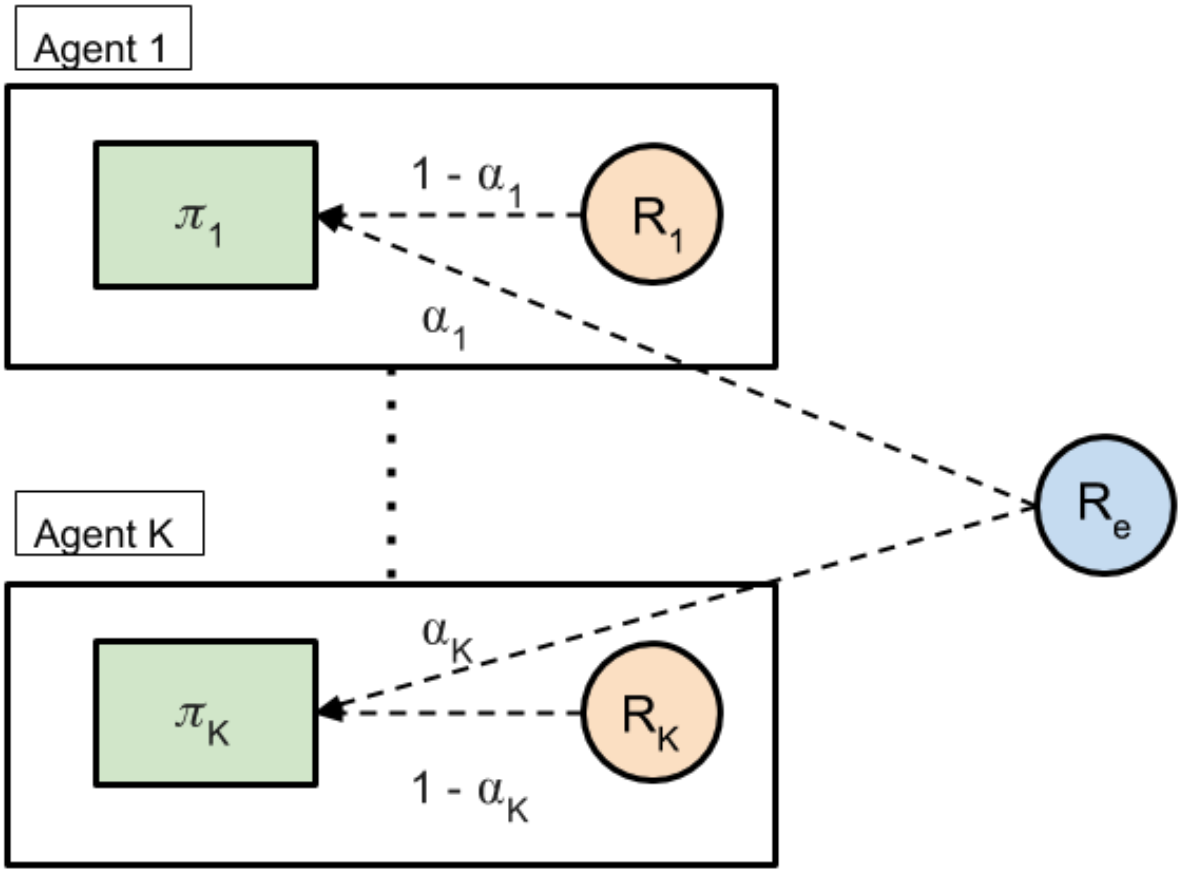

Figure 7.1: Multiagent Preference Balancing. Each Agent k has a personal reward function R_k, and all the agents have a joint environmental reward R_e. The dashed arrows indicate that these rewards are used to learn the policy π_k, and the weights on these arrows denote the weight given to each reward.

Each agent k uses a policy π_k to sample actions a_k. The state s and joint actions $\boldsymbol{a}$ together lead to the reward r_e from the shared reward function R_e. But each agent also has it's personal reward signal R_k, which we call a preference. For each agent, R_e and R_k are balanced using a mixing factor $\alpha_k \in [0, 1]$. The composite reward $R_{k,e}$ for the k^{th} agent can then be written as:

$$R_{k,e} = \alpha_k R_e + (1 - \alpha_k) R_k \tag{7.1}$$

Now consider how we train these agents. The shared task is considered a collaborative multiagent RL task, while each agent can separately view its personal preference as a single agent (while still being impacted by the actions of the other agents). Since we want the policy to train jointly, the state space for the preference task as well as the shared task is the same.

The preference reward function for each agent and the mixing factor α_k are the main components that affect the policy search. The experiments presented in this chapter compare in detail how different mixing values affect the learning behavior of the agents, with respect to both shared task performance and satisfying individual preferences. The overall methodology proposed in this chapter is shown in pseudocode in Algorithm 10.

7.2.1 Learned Preferences

The preference reward for each agent can be defined directly. In this section, however, we consider strategies that can be employed when the demonstrations for a preferred policy can be obtained effectively. Further, we show the advantage of directly learning from a preference policy or preferred demonstrations using supervised learning techniques, which we consider pre-training before executing the agents on the shared task.

Below we discuss techniques to use demonstrations from a preferred policy to pre-train the agent's policy, and to generate a reward function adversarially. If a preferred policy is directly available instead of samples, it is executed in the environment to gather samples for these techniques. Samples from the preferred policy π_k^* for agent k are saved in a dataset $\mathcal{D}_k$, which is used for the following techniques.

7.2.2 Behavioral Cloning

In behavioral cloning we use samples $s_e, a_e \sim \mathcal{D}_k$ and try to learn a function $\mathbf{bc} : S \rightarrow A$ that maximizes the likelihood of taking action a_e in state s_e. Learning a paremeterized imitation policy can be done using supervised learning, namely by finding policy parameters θ_k which minimize the negative log likelihood of the action a_e in state s_e:

$$\min_{\theta_k} -\log(\pi_{\theta_k}(s_e, a_e)) \tag{7.2}$$

With θ_k representing the imitation policy learned from exemplars drawn from demonstration k, $\mathcal{D}_k$.

Algorithm 10 Methodology for Multi-Preference Learning

Input: K, preferences $R_1, R_2, \ldots R_K$, dataset $\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_K$, mixing values $\alpha_1, \alpha_2, \ldots, \alpha_k$, shared reward R_e

Initialize policies $\pi_1, \pi_2, \ldots \pi_K$ **for** $i \leftarrow 0$ **to** K **do**
    pretrain(π_i, $\mathcal{D}_i$) using Equation 7.2
for $t = 0;\ t < T;\ t+ = 1$ **do**
    get joint action $\boldsymbol{a}_t$ act in environment compute mixed reward like in equation 7.1 optimize policies

Such behavioral cloning is used to pre-train the policy to start it off with initial parameters that are better than purely random initialization. However, this initial policy typically does not engender good performance by itself [338]. The reasons why behavioral cloning alone typically doesn't induce a good policy are that behavioral cloning doesn't account for the sequential nature of the problem, and demonstrations are typically too sparse to encapsulate the full complexity of the environment. Furthermore, the slightest errors in mimicking the policy can lead to catastrophic compounding as the episode unrolls. A good imitation policy learns to correct for previous mistakes.

7.2.3 Generative Adversarial Imitation Learning

Inverse RL [1] avoids some of the above pitfalls, by instead generating a reward function from demonstrations. Generating a reward function allows the agent to interact with the environment and correct for previous mistakes. GAIL [149] uses samples $s_e, a_e \sim \mathcal{D}$ from the preference policy and samples $s, a \sim \pi$ from the agent policy, and learns to differentiate them. The agent meanwhile tries to fool the discriminator, by trying to learn a policy that prevents this discrimination. Practically, learning a policy is done by using the discriminator output as a reward signal. This adversarially learnt reward signal can be used as the preference reward by the agent. GAIL uses the following objective function:

$$\min_{\theta_k} \max_{\psi_k} \mathtt{E}_{\pi_e} \left[log D_{\psi_k}(s_e, a_e) \right] + \mathtt{E}_{\pi_{\theta_k}} \left[log(1 - D_{\psi_k}(s_k, a_k)) \right] \tag{7.3}$$

where θ_k are the parameters of the agent policy π_k and ψ_k are the parameters of the Discriminator D associated with that agent. The preference reward then becomes $R_k(s_k, a_k) = -\left(log(1 - D_{\psi_k}(s_k, a_k))\right)$

7.3 Game-Theoretic Analysis

The purpose of this section is to provide a high-level intuition of what the interplay between individual preferences and shared goals is from a game-theoretic perspective, and provide a more formal understanding of under which condition there exists a vector of α values that allow for good shared goal performance while agents still pursue their individual objectives.

Let us assume there are two agents A and B interacting in a stage game in which each agent can take one of two actions (this can be Bach-Stravinsky or Matching Pennies or any other game of this form). Agents A, B's preferences are denoted with the following payoff matrices, respectively:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

However, *unlike* a stage game, in this game there is also *an environment* which gives both agents the following reward according to their matching actions:

$$\textit{payoff}(\textit{environment}) = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix}$$

Let α_a, α_b be the α values for agents A, B respectively, determining their extent of blending environment and individual reward signals. The resulting game payoff matrices for A,B are then:

$$A_e = \begin{bmatrix} (1-\alpha_a)\cdot a_{11} + \alpha_a \cdot e_{11} & (1-\alpha_a)\cdot a_{12} + \alpha_a \cdot e_{12} \\ (1-\alpha_a)\cdot a_{21} + \alpha_a \cdot e_{21} & (1-\alpha_a)\cdot a_{22} + \alpha_a \cdot e_{22} \end{bmatrix}$$

$$B_e = \begin{bmatrix} (1-\alpha_b)\cdot b_{11} + \alpha_b \cdot e_{11} & (1-\alpha_b)\cdot b_{12} + \alpha_b \cdot e_{12} \\ (1-\alpha_b)\cdot b_{21} + \alpha_b \cdot e_{21} & (1-\alpha_b)\cdot b_{22} + \alpha_b \cdot e_{22} \end{bmatrix}$$

Assume wlog that e_{11} yields the highest environment reward. Let us consider which α values could lead to e_{11} being an equilibrium. The combination of actions leading to e_{11} is a Nash equilibrium [**?**] if: $(1-\alpha_a)\cdot a_{11} + \alpha_a \cdot e_{11} > (1-\alpha_a)\cdot a_{21} + \alpha_a \cdot e_{21}$ and $(1-\alpha_b)\cdot b_{11} + \alpha_b \cdot e_{11} > (1-\alpha_b)\cdot b_{12} + \alpha_b \cdot e_{12}$.

Simplifying the math leads to: $\alpha_a > \frac{a_{21}-a_{11}}{e_{11}-e_{21}+a_{21}-a_{11}}$, and analogously,
$\alpha_b > \frac{b_{12}-b_{11}}{e_{11}-e_{12}+b_{12}-b_{11}}$.

Let us denote $\delta(a) = a_{21}-a_{11}$ and $\delta(e) = e_{21}-e_{11}$. Observe that $\delta(e) \geq 0$ since definitionally e_{11} yields the highest environment reward.

Then $\alpha_a > \frac{\delta(a)}{\delta(a)-\delta(e)}$.

In other words, α needs to be greater than the ratio of selfish improvement by defecting relative to the combined impact of defecting on the difference between selfish and environment reward.

If we also consider that $1 \geq alpha_a \geq 0$ and $1 \geq alpha_b \geq 0$, we get a simple set of linear inequalities - if it has a feasible solution, then there exists a vector of α values for the agents that would lead to the environment-wise ideal solution becoming a game-theoretic equilibrium. Observe that this analysis would hold even for arbitrary values of e_{ij}, which would imply we can exhaustively study how multiple α values can lead to multiple equilibria.

We now show this analysis holds for $k+1$ agents and $n+1$ actions.

Let us assume there are k agents $A_i, 1 \leq i \leq k$, all interacting in a stage game in which each agent can take one of n actions. Let's denote the preferences over actions for the agent (conditional on the vector of other agents' actions) in the following way: $\textit{Prefs}(A_i|a^1, a^2, \ldots, a^{i-1}, a^{i+1}, \ldots, a^k) = \langle a^i_0, a^i_1, \ldots, a^i_n \rangle$. At the same time, the environment reward for an agent A_i, conditional on the vector of other agents' actions) is: $\textit{Prefs}(A_i|a^1, a^2, \ldots, a^{i-1}, a^{i+1}, \ldots, a^k) = \langle e^i_1, e^i_2, \ldots, e^i_n \rangle$. We observe that $\langle e^i_{1 \leq l \leq n} \rangle$ is the same for all agents given a joint action $a^0, a^1, \ldots, a^i, \ldots, a^k$ with all actions besides those of agent A_i being fixed. For each agent A_i there is also a correspodning

α coefficient, α_i, which balances selfishness with collaboration for each agent. Then, with other agents' actions being fixed, any agent's observed utility for a given actiona_l is $(1-\alpha_i)\cdot a^i_l + \alpha_i \cdot e_l$. For an action l to be the best response for an agent A_i the following inequality needs to hold for all agents and actions:

$$\forall i \forall j \neq l . (1-\alpha_i)\cdot a^i_l + \alpha_i \cdot e^i_l > (1-\alpha_i)\cdot a^i_{j\neq l} + \alpha_i \cdot e^i_{j\neq l}$$

which again leads to the inequality

$$\forall i \forall j \neq l . \alpha_i > \frac{a^i_{j\neq l} - a^i_l}{e^i_l - e^i_{j\neq l} + a^i_{j\neq l} - a^i_l}$$

Let us denote $\delta(a^i_j) = a^i_{j\neq l} - a^i_l$ and $\delta(e^i_j) = e_{j\neq l} - e^i_l$ (recall a^i_l and e^i_l are assuming the other agents' actions are fixed). Then we obtain a generalized set of inequalities:

$$\forall i \forall j . \alpha_i > \frac{\delta(a^i_j)}{\delta(a^i_j) - \delta(e^i_j)}$$

These inequalities, combined with the constraints that $\forall i 1 \geq \alpha_i \geq 0$, can all be solved efficiently, and if satisfied, it means the vector of α_i values again leads to a Nash equilibrium.

The analysis presented in this section is revealing in several ways. First, it serves as formal support to the idea that having individual preferences can be beneficial to shared task performance. Furthermore, it illustrates that this benefit can be conferred by the individual preferences even if these preferences are not perfectly aligned with the shared task. Lastly, it specifies concrete conditions on the extent of selflessness/selfishness balance (α value) required from each agent for the optimal joint action to become a Nash equilibrium. This grounding motivates us to study the problem empirically more extensively in the far more complicated domains studied in this chapter.

7.4 Experiments: Multiagent Music Generation

To study the preference balancing approach proposed in this chapter, we devised a novel music generation domain that involves learning how to jointly generate chords. In the following section I will describe the studied task and the subsequent experimental setup for studying how well agents can balance individual preferences with the shared task in this domain.

7.4.1 Domain Description

As I will discuss in Chapter 8, the task of autonomous music generation has been of interest for decades. In the context of this chapter, we introduce a novel, simplified multiagent environment that is focused on learning how to generate sequences of chords. In this setting each agent plays a single note at each timestep out of the 12 possible pitch classes, simply denoted with integer values

$\{0, \ldots, 11\}$.[12]

A valid chord is a combination of four notes which plays simultaneously and is a permutation of either a major-major, major-minor, minor-minor or minor-major seventh, leading to a total of at most 4 possible configurations times 12 possible base note values (relative to the total of 12^4 pitch class combinations, meaning only a small fraction of possible chords is valid). For example, without getting into any music theory, a major-major seventh chord with a base note of 0 has pitch values $[0, 4, 7, 11]$. If the base note were 2, this chord would have values $[2, 6, 9, 13]$. However, since we have restricted the action space to one octave, or one series of pitch classes $\{0, \ldots, 11\}$, the value 13%12 gives pitch value 1, leading to a valid chord which is a permutation of the standard major seventh, $[1, 2, 6, 9]$.

The environment gives a reward both on the transition and the chord (see next subsection). An example for a single step for all four agents is provided in Figure 7.2.

7.4.2 Reward Function

The core idea in the chord generation domain is that the four agents must coordinate more than to just generate an acceptable chord (as discussed above, no more than $4 \cdot 12 = 48$ note configurations can possibly be considered valid, out of the $12^4 = 20736$ possible 4-note configurations). They also have to ensure that the transition between chords is valid. The restrictions on these transitions are reflected through the following reward criteria:

- There is a weighted penalty for repeating the exact same notes from timestep t to timestep $t+1$, even if different agents are playing different notes. This penalty is meant to discourage the agents from playing two inversions of the same chord one after the other.
- There is a weighted penalty for repeating intervals from timestep t to timestep $t+1$, even if the base note is changed. This penalty is meant to discourage the agents from playing the same chord configuration twice in a row, even if the base note is changed (for instance moving from [0,4,7,9] ([C,E,G,A]) to [2,6,9,10]([D,F#,A,B]).

These criteria are heuristics designed by us, but with solid grounding in music theory, as surprise and novelty are basic driving forces in Western music [68].

Assume the current note configuration is $state = \{n_0^t, n_1^t, n_2^t, n_3^t\}$, and the actions taken by the agents, i.e. the new notes played, are $action = \{n_0^{t+1}, n_1^{t+1}, n_2^{t+1}, n_3^{t+1}\}$. Then:

[12] Relating the scale to actual notes, 0 denotes C, 1 denotes $C\#$, 2 denotes D and so forth up to 11 = B.

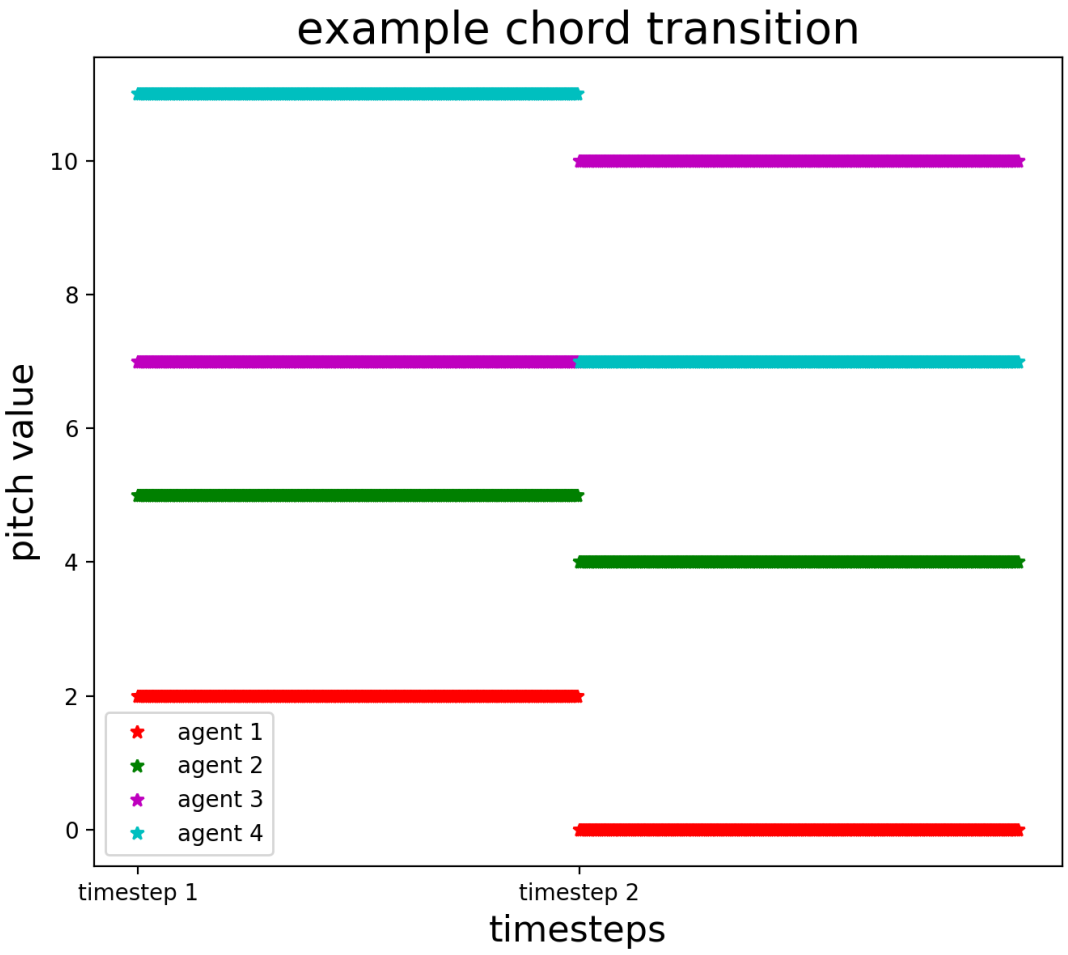

Figure 7.2: An illustration of the chord sequence generation domain, with four agents forming a chord transition. In this example the transition is from an inverted G major 7th chord, [D, F, G, B], represented with pitch class values [2,5,7,11] to a canonical form C major 7th chord, [C, E, G, Bb], represented with pitch class values [0,4,7,10].

$$note\_repeat(s,a) = \begin{cases} 1 \text{ if } |s \triangle a| > 0, \\ 0 \text{ otherwise} \end{cases}$$

$$interval\_repeat(s,a) = \begin{cases} 1 \text{ if } |intervals(s) \triangle intervals(a))| > 0, \\ 0 \text{ otherwise} \end{cases}$$

With $\triangle$ being the symmetric difference operator. $is\_chord(a)$ is defined to be 1 if the actions

taken by the agents are a valid chord configuration and 0 otherwise. Ultimately we get the following reward function:

$$R_e(s,a) = (1-w_2)\cdot((1-w_1)\cdot \textit{interval\_repeat}(s,a) + \\ w_1 \cdot \textit{note\_repeat}(s,a)) + w_2 \cdot \textit{is\_chord}(a)$$

Thus, the weights of w_1, w_2 define a specific reward function in this family of chord environment reward functions.

It is worth noting that the design of the reward function for the chord generation environment does rely to some extent on domain knowledge and familiarity with musical concepts (even though learning a policy which maximizes this reward function does not). In lieu of those, the environment can be perceived intuitively as a sequence generator in which agents should produce number sequences with some specific joint values and transitions to new numbers.

7.4.3 Generating Examples for Behavioral Cloning

Each agent learns its intrinsic preference function by imitation learning based on pre-generated examples. To generate examples, we fix the w_1, w_2 weights defining a reward function described in the previous section, and in addition define a subset of the permissible chords to be a single agent's "preferred chord set". We then synthesize example chord sequences at each time step by generating a random set of $n = 20$ permissible chords, and picking the one that maximizes the reward with respect to the pre-defined individualized reward function. This procedure is described in Algorithm 11.

Algorithm 11 Generating individualized exemplar chord sequences.

Input: sample search size n, number of steps T, preference weights w_1, w_2, preferred chord set C

Initialize current chord played c_t at $[0,0,0,0]$ (all agents playing same base note);
Initialize chord sequence $S = \{c_t\}$;
Define $R_k := (1-w_2)\cdot((1-w_1)\cdot \textit{interval\_repeat} + w_1 \cdot \textit{note\_repeat}) + w_2 \cdot \textit{is\_chord}$
for $i \leftarrow 0$ **to** T **do**
    sample C_t^T possible chords from C with random note offsets; Pick next chord $c_{t+1} = \textit{argmax}_{R_T}(c_t, C_t^k)$ append next chord $S = S + c_{t+1}$

7.4.4 Learning Architecture

Every agent policy is parameterized as a multi-layer perceptron with 3 layers of 512 units each. In addition, each agent also has a Discriminator D_k which is made up of another 2-layer network with 64 units in each layer. We used tanh activations for the policy and ReLU activations for the discriminator.

7.5 Results in the Music Generation Domain

All four agents were initially taught their initial preferences via behavioral cloning as described in Sections 7.2.2 and 7.2.3. Each agent sampled different examples generated via Algorithm 11 to perform behavioral cloning for 2000 steps. With these experimental settings, the log-likelihood plateaus at about $-.5$ after this number of pretraining steps (giving a likelihood of about .6 to the expert rollout action on average).

As a reminder, as defined in Equation 7.1, the mixing ratio parameter α_k for agent k determines the proportions of how the environment and individualized imitation rewards are blended together: $R_{k,e} = \alpha_k R_e + (1 - \alpha_k) R_k$.

Our first experiment was to set all four agents' mixing rates to one of three distinct values: $[0, 0.5, 1]$. With $\alpha = 1$, all the agents use the environment reward exclusively in training (thereby ignoring their pre-trained preference signal in the learning process). With $\alpha = 0$, all the agents use their intrinsic preferences learned via behavioral cloning (thereby ignoring the environment signal in the learning process). With $\alpha = .5$, all agents equally balance both their intrinsic preference signals and the environment reward in learning.

When all four agents use $\alpha = 0$, their performance with respect to the environment reward does not improve significantly compared to the learning rates with higher α values. However, somewhat counter-intuitively, an α mixture ratio of 0.5 does about as well as using a value of 1, and in fact learns substantially faster and outperforms a uniform $\alpha = 1$ in the initial phase of the experiment. When we consider the intrinsic preference as reflected by the GAIL reward, unsurprisingly, these individual preference rewards are consistently higher with $\alpha = 0$, but it is also apparent that $\alpha = 0.5$ does significantly better than $\alpha = 0$ as well in that respect, meaning setting $\alpha = 0.5$ indeed balances individualized preferences with the shared task without sacrificing too much on the performance of the shared task.

These results are presented in Figures 7.3 and 7.4.

Though having the agents share an α value is in itself a plausible setting, it is of interest to study how the agents perform when the value of α changes from agent to agent. We consider 3 additional variable α configurations: $[1, 1, 1, 0]$, which means three agents care exclusively about the environment reward and one only cares selfishly about its own intrinsic preferences; $[1, 1, 1, 0.5]$, which means three agents care exclusively about the environment reward and one balances its intrinsic preferences with the environment reward; and $[.9, .9, .9, .1]$, which can be seen as a softer version of $[1, 1, 1, 0]$.

Interestingly, the results indicate that a $[1, 1, 1, 0.5]$ mix of α values does better than either a uniform $\alpha = 0.5$ or $\alpha = 1$. Furthermore, an α values configuration of $[1, 1, 1, 0.5]$ outperforms both $[.9, .9, .9, .1]$ or setting all α values to 1.

This result suggests that having one agent inject personalized preferences into the learning process is helpful, but once one agent does so additional individualized agents aren't necessarily as helpful.

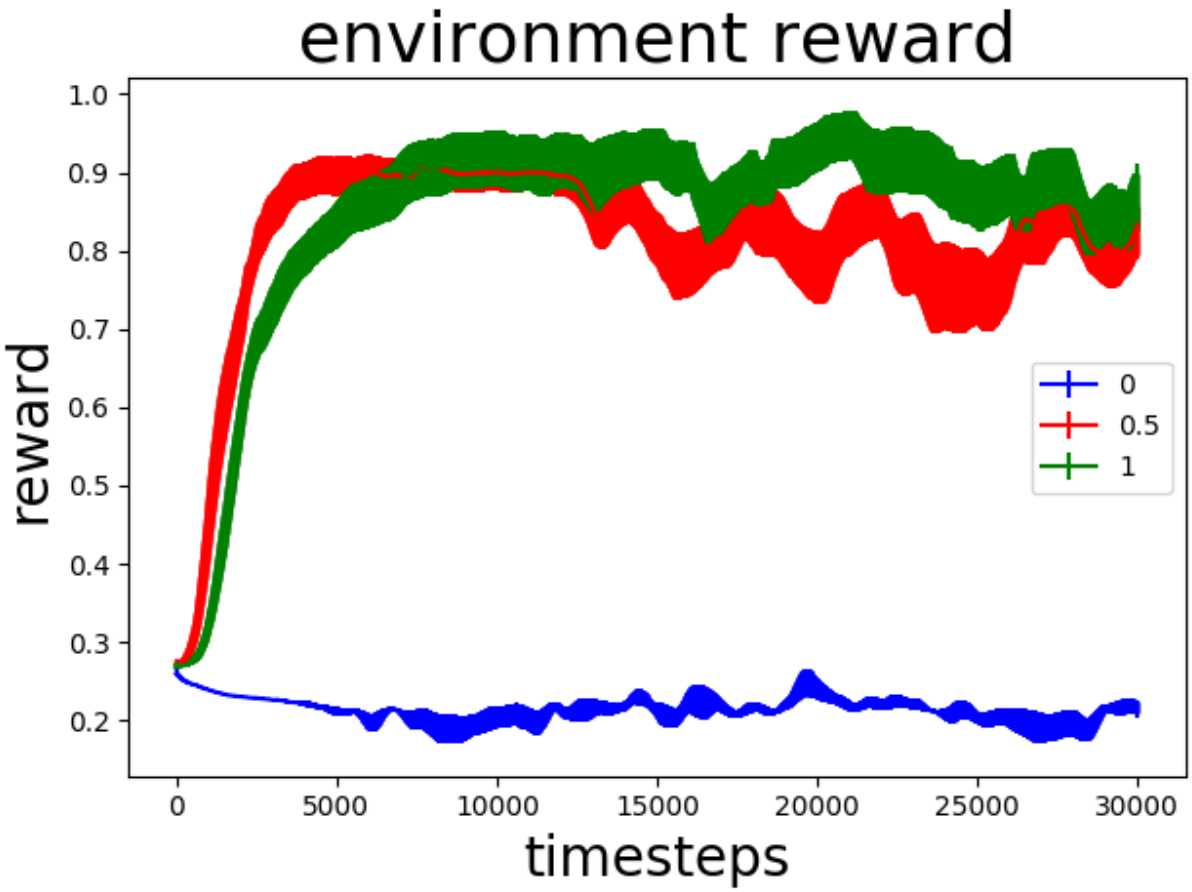

Figure 7.3: Average environment reward over timesteps in the chord generation domain. Results averaged out over $n = 8$ repetitions. Envelopes represent the standard error. The results compare three values of mixing ratio parameter α - $[0, 0.5, 1]$.

It also gives rise to the hypothesis that a possible benefit of optimizing for both individualized preferences and environment reward is that it introduces a regularizing effect, limiting the policy space in a way that makes the agents more predictable. To complement this claim, however, in Section 7.7 we show that this explanation is in itself insufficient, since simply fixing some of the agents (i.e. having their policies be fixed, without learning or adaptation) does not always lead to improved performance.

Since we do know the specific individualized reward functions that were used to generate the examples the agents trained on to initialize their preferences, another measure of intrinsic reward is to evaluate how well the agents do with respect to those reward functions. Indeed, we can see that, unsurprisingly, setting α values for all agents to $\alpha = 0$ does the best with respect to that measure, but a blend of α values such as $[1, 1, 1, 0]$ does consistently better than setting all α values to $\alpha = 1$.

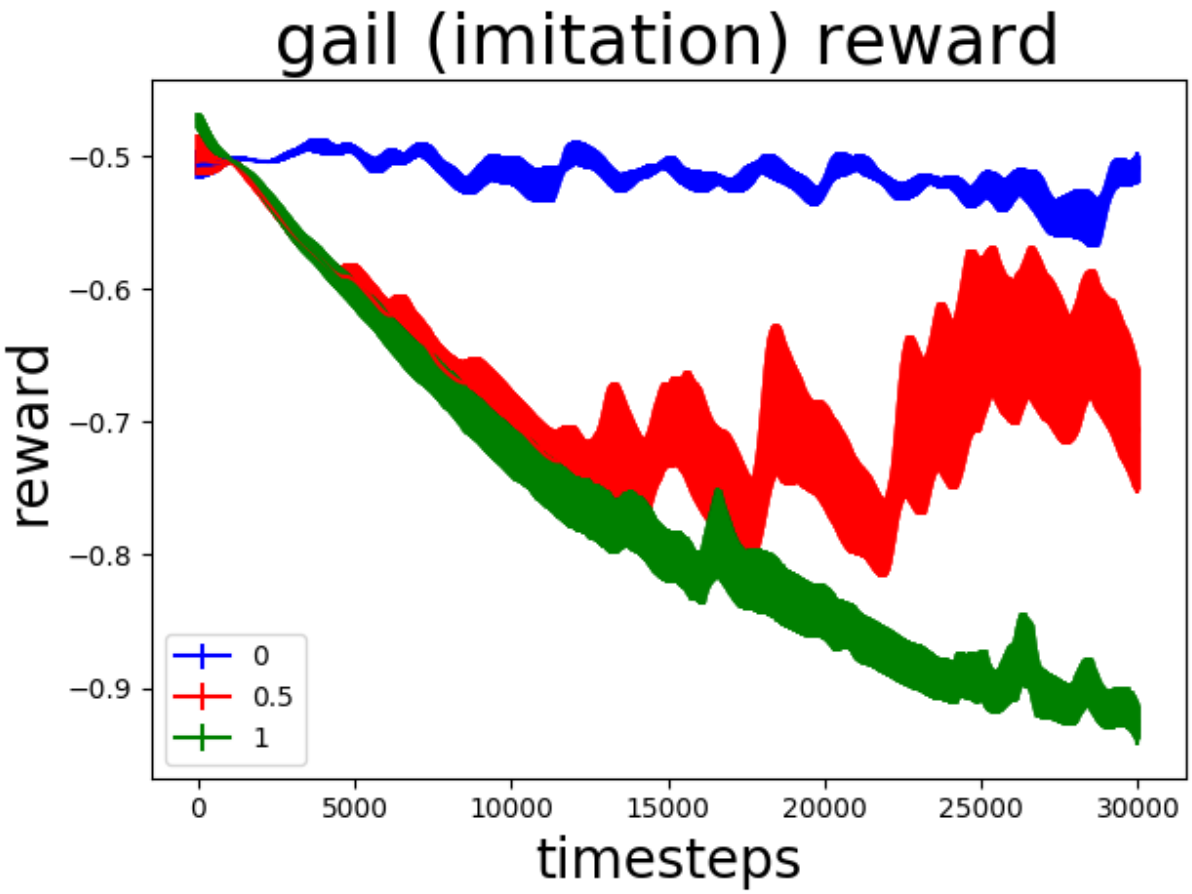

Figure 7.4: Average GAIL (imitation) reward over timesteps in the chord generation domain. Results averaged out over $n = 8$ repetitions. Envelopes represent the standard error. The results compare three values of mixing ratio parameter α - $[0, 0.5, 1]$.

The results with variable α configurations are presented in Figures 7.5, 7.6 and 7.7.

7.6 Extending the Results to a Wider Range of Agent and Preference Mixtures

The results presented in Section 7.5 are both telling and encouraging - they establish that indeed learning a shared task with existing preferences does not hinder collaboration and in fact *can be beneficial* in certain configurations. However, a more extensive analysis of different configurations is necessary in order to make a more substantial claim regarding the relationship between shared task reward and individualized preferences.

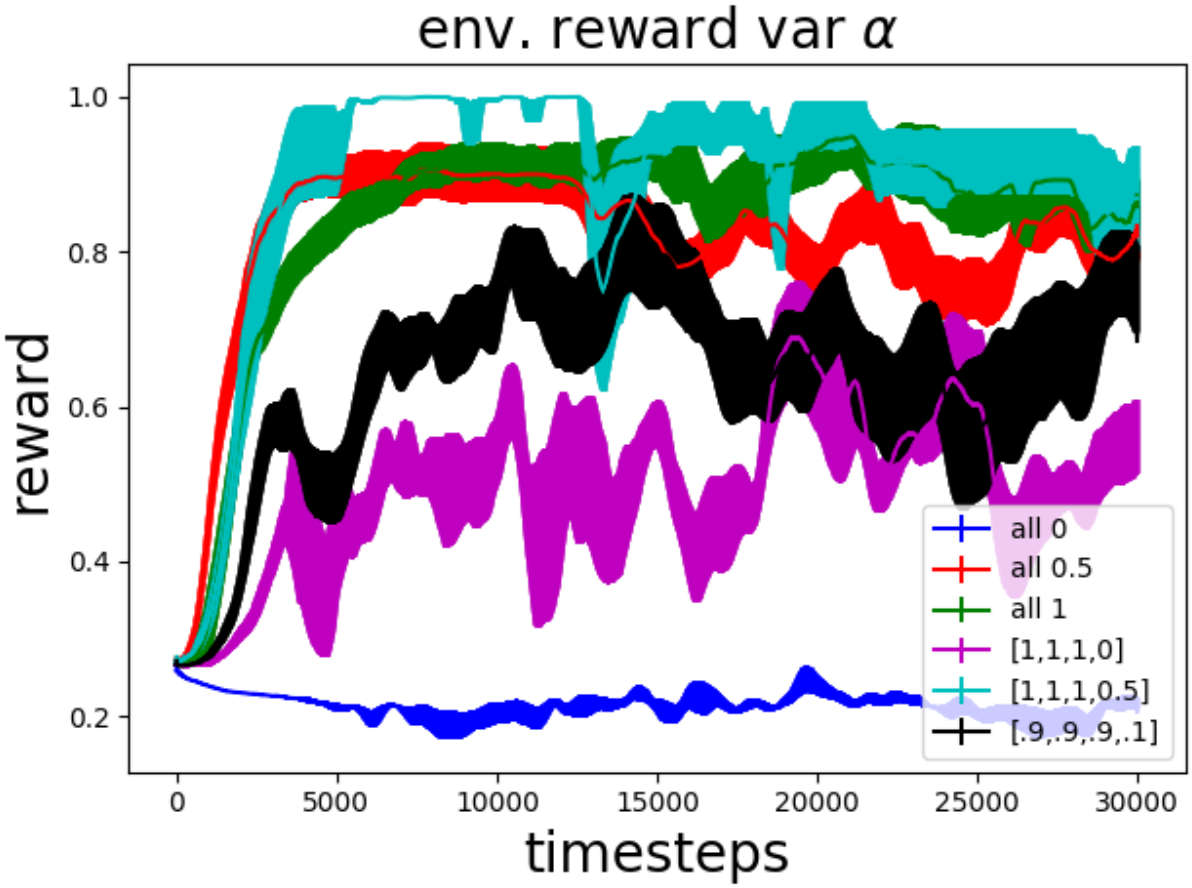

Figure 7.5: Average environment reward over timesteps in the chord generation domain. Results averaged out over $n = 8$ repetitions. The results compare six values of mixing ratio parameter α combinations.

7.6.1 Agent Types

The first step towards this end is to extend the palette of agents. Recall that the coefficients w_1, w_2, described in Section 7.4.2, control the blending between chord and transition constraints and thus define, along with a specific preferred chord subset, a unique preference we can draw examples from and train an agent to emulate. In this set of experiments, we configured four distinct agent policies by generating examples with the following two coefficient configuratins: $\{0.2, 0.8\}, \{0.8, 0.2\}$, and choosing two sets of preferred chords per agent: either major chords or minor chords. The Cartesian product of the weight configuration and the chord configuration yields 4 distinct preference profiles denoting very different preferences with respect to the proportions between chord utility and transition utility, between chords and melodies, and between types of preferred chords. Marking these four agent preference profiles arbitrarily as $\{0, 1, 2, 3\}$, we considered 23 different mixes

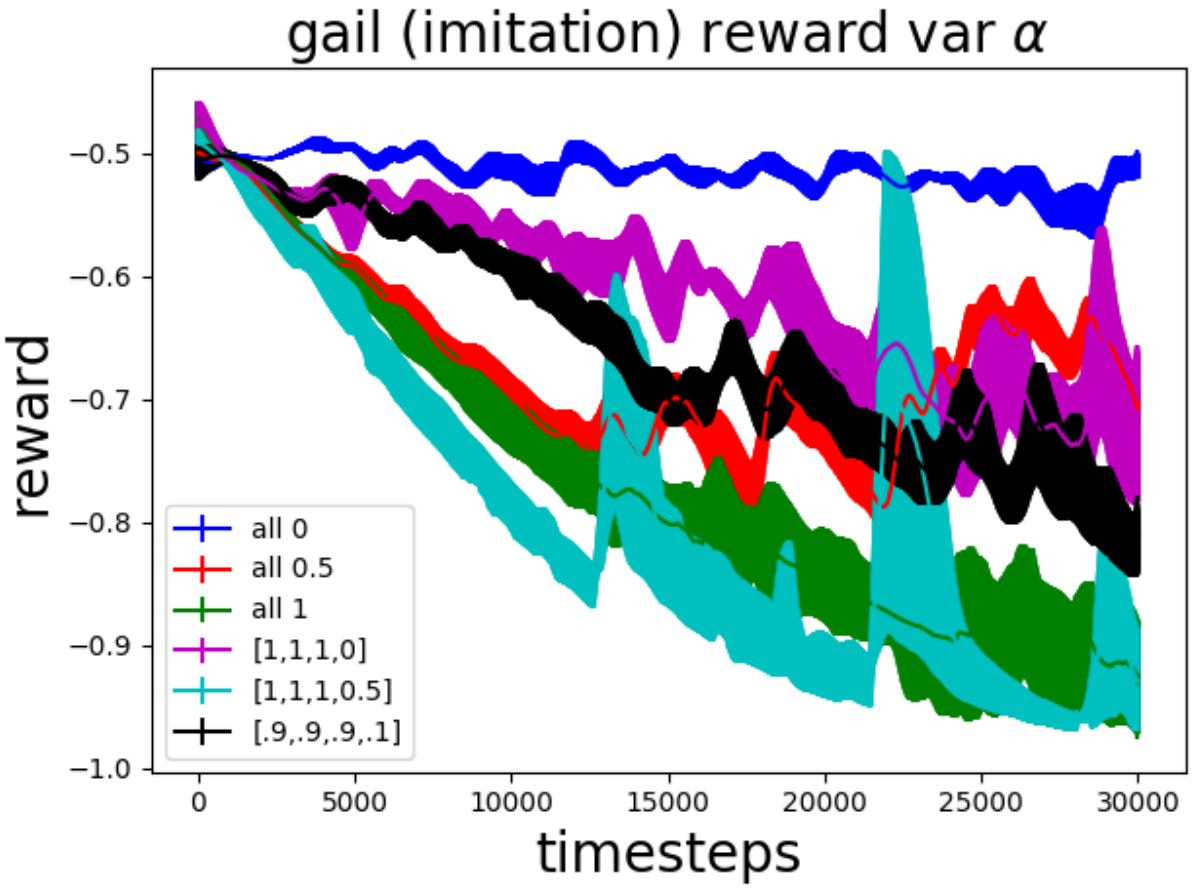

Figure 7.6: Average GAIL (imitation) reward over timesteps in the chord generation domain. Results averaged out over $n = 8$ repetitions. The results compare six values of mixing ratio parameter α combinations.

of different agent types:

$$\begin{aligned} \textit{agent configurations} = \{&(0,0,0,0),(0,0,0,1),(0,0,0,2),\\ &(0,0,0,3),(0,0,1,1),(0,0,2,2),(0,0,3,3),(0,1,1,1),(0,1,2,3),\\ &(0,2,2,2),(0,3,3,3),(1,1,1,1),(1,1,1,2),(1,1,1,3),\\ &(1,1,2,2),(1,1,3,3),(1,2,2,2),(1,3,3,3),(2,2,2,2),\\ &(2,2,2,3),(2,2,3,3),(2,3,3,3),(3,3,3,3)\} \end{aligned} \tag{7.4}$$

7.6.2 Reward Mixing Configurations

Remember that α is the coefficient that demarcates how an individual agent blends its own intrinsic reward, derived from its preference, with the environment signal. To better study the tradeoffs

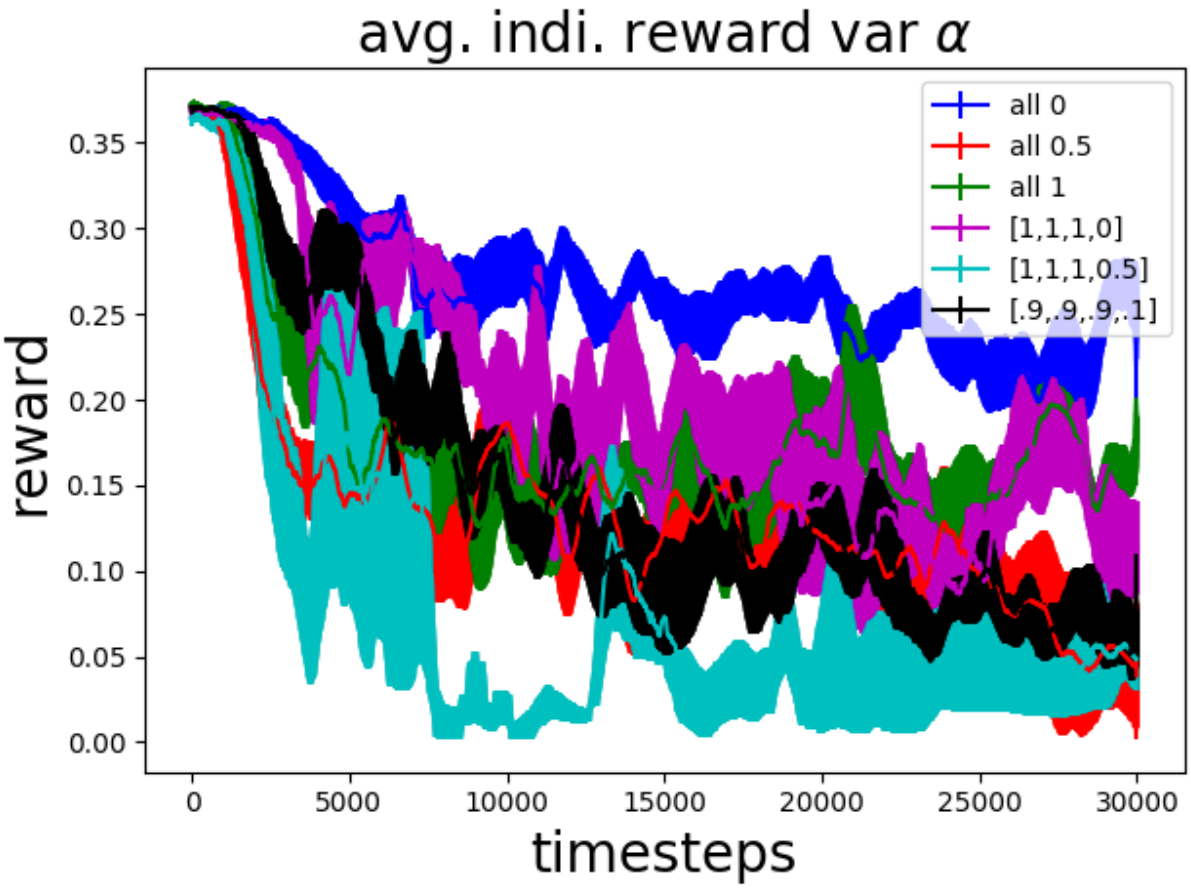

Figure 7.7: Average intrinsic reward (as reflected by latent individualized reward function) over timesteps in the chord generation domain. Results averaged out over $n = 8$ repetitions. The results compare six values of mixing ratio parameter α combinations.

between different agent configurations and different alpha configurations, we canvased a much wider range of α coefficient configurations:

$$\begin{aligned} \alpha \textit{ configurations} = \{&[0,0,0,0],[0,0,0,0.1],[0,0,0,0.5],[0,0,0,0.9],\\ &[0,0,0,1],[0,0,1,1],[0.1,0.1,0.1,0.1],[0.5,0.5,0.5,0.5],\\ &[0.9,0.9,0.1,0.1],[0.9,0.9,0.9,0.1],[0.9,0.9,0.9,0.9],\\ &[1,1,1,0],[1,1,1,0.1],[1,1,1,0.5],[1,1,1,0.9],[1,1,1,1]\}. \end{aligned} \tag{7.5}$$

7.6.3 Analysis

To comparatively analyze the performance of different mixtures of agents with different proportions of environment vs. individualized reward preferences, Figures 7.8 - 7.10 represent a cross-section of average rewards at a given number of steps, smoothed over a window of 500 steps and averaged

over 8 repetitions. Each cell in these heatmaps represents the average reward at a given timestep for a given agent mixture and α coefficient configuration - the brighter the color, the higher the reward. A red star demarcates the maximum over α coefficient configurations for each given agent mixture configuration.

In Figure 7.8 we first observe that in the initial steps of learning, homogeneous agent teams - demarcated as (0,0,0,0), (1,1,1,1), (2,2,2,2) and (3,3,3,3) on the Y axis - do better initially than heterogeneous teams. This observation is easy to understand intuitively, as homogeneous agents have similar preferences and their behavior is therefore implicitly better captured by each other's models, leading to better coordination. A second and significant observation is that in the initial steps, an α coefficient configuration of $[0.9, 0.9, 0.9, 0.9]$ does much better for almost all agent mixtures than a configuration of $[1, 1, 1, 1]$, meaning that in the initial stages of learning injecting individual preferences into the agents' decision-making meaningfully improves on their performance compared to having the agents learn only from the environment reward. There is also a discernible peak at $[0.5, 0.5, 0.5, 0.5]$.

This observation can again be explained by the fact that the intrinsic reward signal is less sparse in structure - the environment reward signal is mostly very low unless a correct note group is selected by the agents, whereas the imitiation signal produces a richer signal. It is also plausible that successful imitation is conducive of better performance in the long run even if it does not pan out at any given step, as the skills learned through imitation are productive for joint task performance. In Figure 7.9 we see that while other agent mixture configurations catch up to the homogeneous agent configurations, implying the advantage of homogeneous teams is only significant in the first phase of joint learning, the overall better performance of $[0.9, 0.9, 0.9, 0.9]$ (and $[0.5, 0.5, 0.5, 0.5]$) is still apparent. Most significantly, even nearing convergence at 29500 steps, a configuration of $[0.9, 0.9, 0.9, 0.9]$ does about as well or better than $[1, 1, 1, 1]$, meaning injecting intrinsic preferences into the agents' learning processes, at the very least does not hinder learning in the longer run, and indeed setting the agents' α values to $[0.9, 0.9, 0.9, 0.9]$ yields better results than any other specific α coefficient configuration.

7.7 Experiments with Fixed Agents

To further study the role of individual preferences in team learning dynamics, we have pursued experiments in which 1, 2 or 3 agents are fixed (i.e. use a fixed policy without learning or adapting) and the rest are learning.

In the first experiment , we considered four *homogeneous* agents of type 1 (weights $\{0.2, 0.8\}$ and a preference for major chords). We examined two sets of fixed policies - policies that were learned with all the agents being selfish ($\alpha = 0$), and all agents being selfless and fully focused on the shared task ($\alpha = 1$). Then we proceeded to fix one, two or three agents and observed how team performance changed. The fixed policies we used were originally learned jointly, which sould

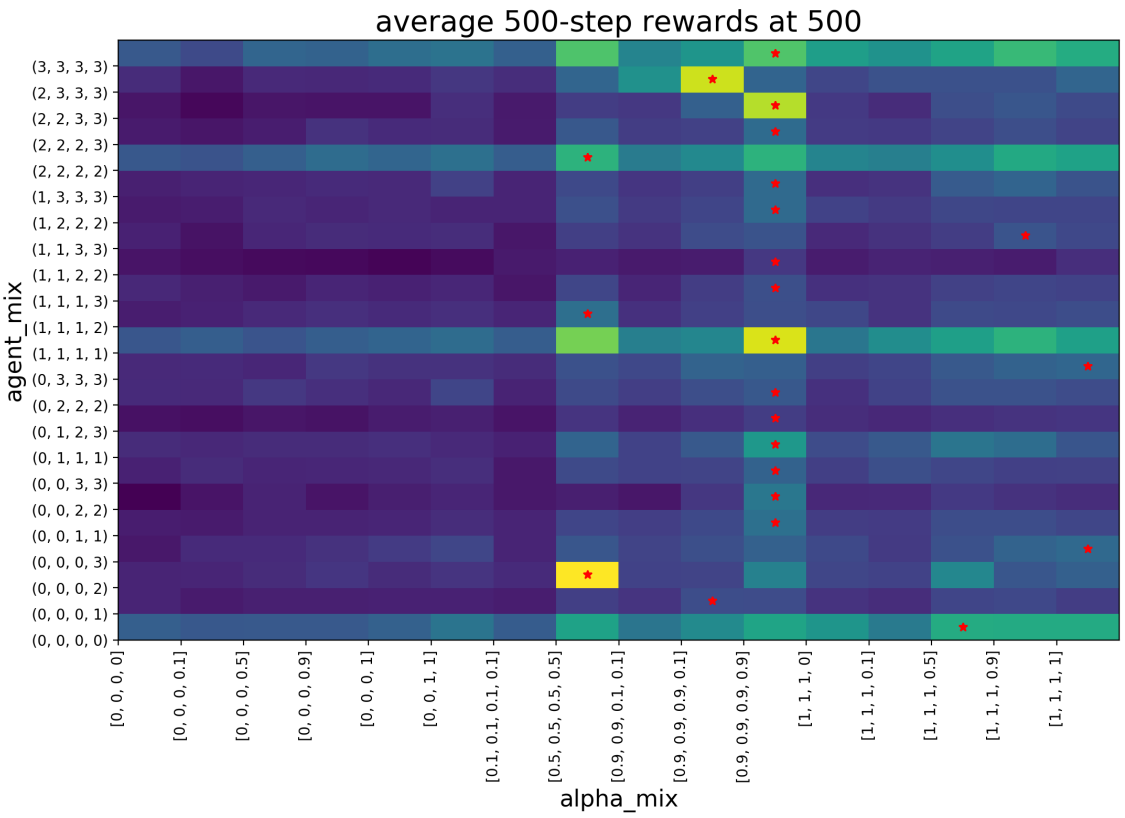

Figure 7.8: Heatmap of average rewards at the 500-th step, smoothed using a 500 step window (meaning the results are averaged over a rolling window of 500 timesteps). The brighter the color, the higher the reward. X axis represents α coefficient configurations, ordered lexically. Y axis represents agent mixture configurations. Results are averaged over $n = 8$ repetitions per agent mixture and α coefficient configuration. A red star denotes the maximum over α coefficient configurations for each given agent mixture configuration.

suggest they should be better coordinated than separately learned policies. The results of this experiments are presented in Figures 7.11 and 7.12.

As one may observe, in the homogeneous case, if the agents were trained to be selfish ($\alpha = 0$), the more fixed agents you add to the environment, the worse team performance (i.e. average reward in the shared task) goes down, implying that fixed selfish agents effectively sabotage the group effort in a way that makes it harder for the learning agents to adapt. This result also serves as a good demonstration for the fact that individual preferences can be significantly deleterious to the entire team even if they're pursued by just one agent.

Conversely, if we consider fixed agents whose policies were learned in a collaborative setting ($\alpha = 1$), fixing just one agent benefits the team the most. Indeed this can be intuitively explained by the fact that since the agents are homogeneous and the fixed policy was geared towards collaboration, with an easy-to-predict fixed agent, the other three agent can easily learn a

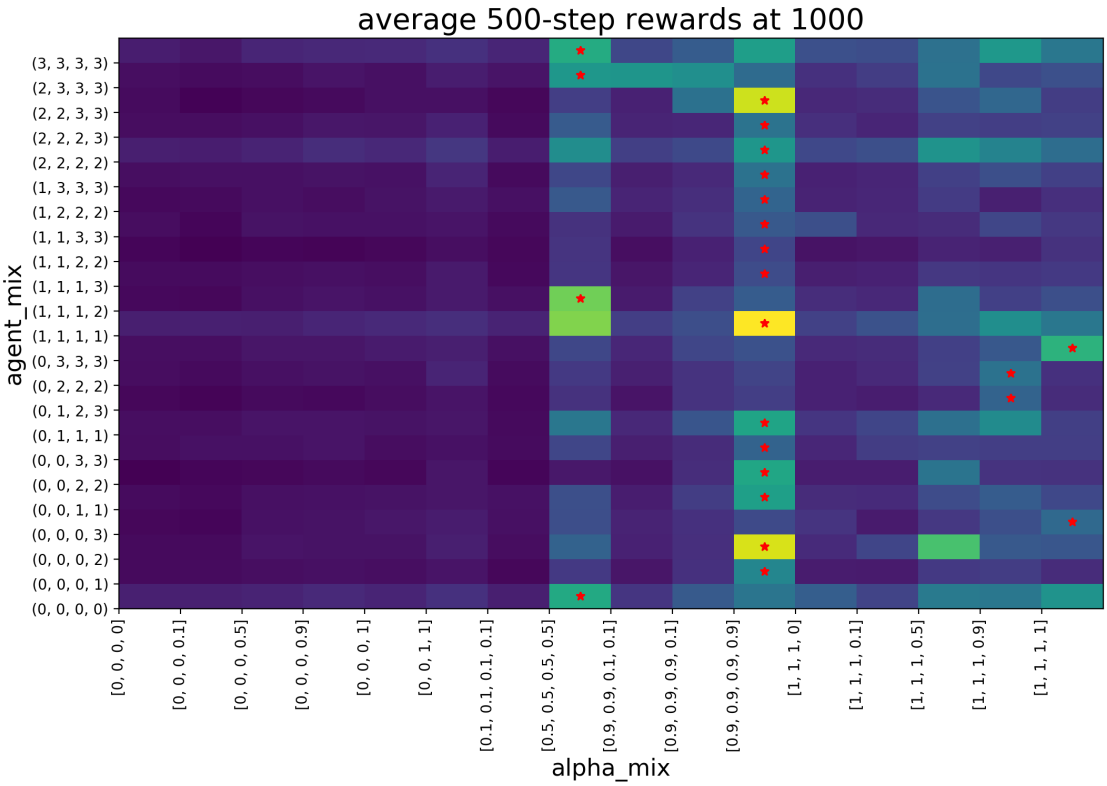

Figure 7.9: Heatmap of average rewards at the 1000-th step, smoothed using a 500 step window. The brighter the color, the higher the reward. X axis represents α coefficient configurations, ordered lexically. Y axis represents agent mixture configurations. Results are averaged over $n = 8$ repetitions per agent mixture and α coefficient configuration. A red star denotes the maximum over α coefficient configurations for each given agent mixture configuration.

policy that always produces acceptable chords and transitions. Fixing two or three agents leads to a speed-up in learning, but does not asymptotically outperform the baseline of four agents learning simultaneously (in a fully collaborative manner, i.e. $\alpha = 1$ for all agents).

In the second experiment, we considered four *heterogeneous* agents of types 1,2,3 and 4 as described in Section 7.6. Again, we examined two sets of fixed policies - policies that were learned with all the agents being selfish ($\alpha = 0$), and all agents being selfless and fully focused on the shared task ($\alpha = 1$). Then we proceeded to fix one, two or three agents and observed how team performance changed. Again, the fixed policies we used were originally learned jointly. The results of this experiments are presented in Figures 7.13 and 7.14.

Let us first consider the setting in which the fixed policies were generated in a collaborative setting (learning with $\alpha = 1$ for all agents). One may first observe that in the heterogeneous case, unlike the homogeneous case, no extent of agent fixing leads to better long-term performance, but

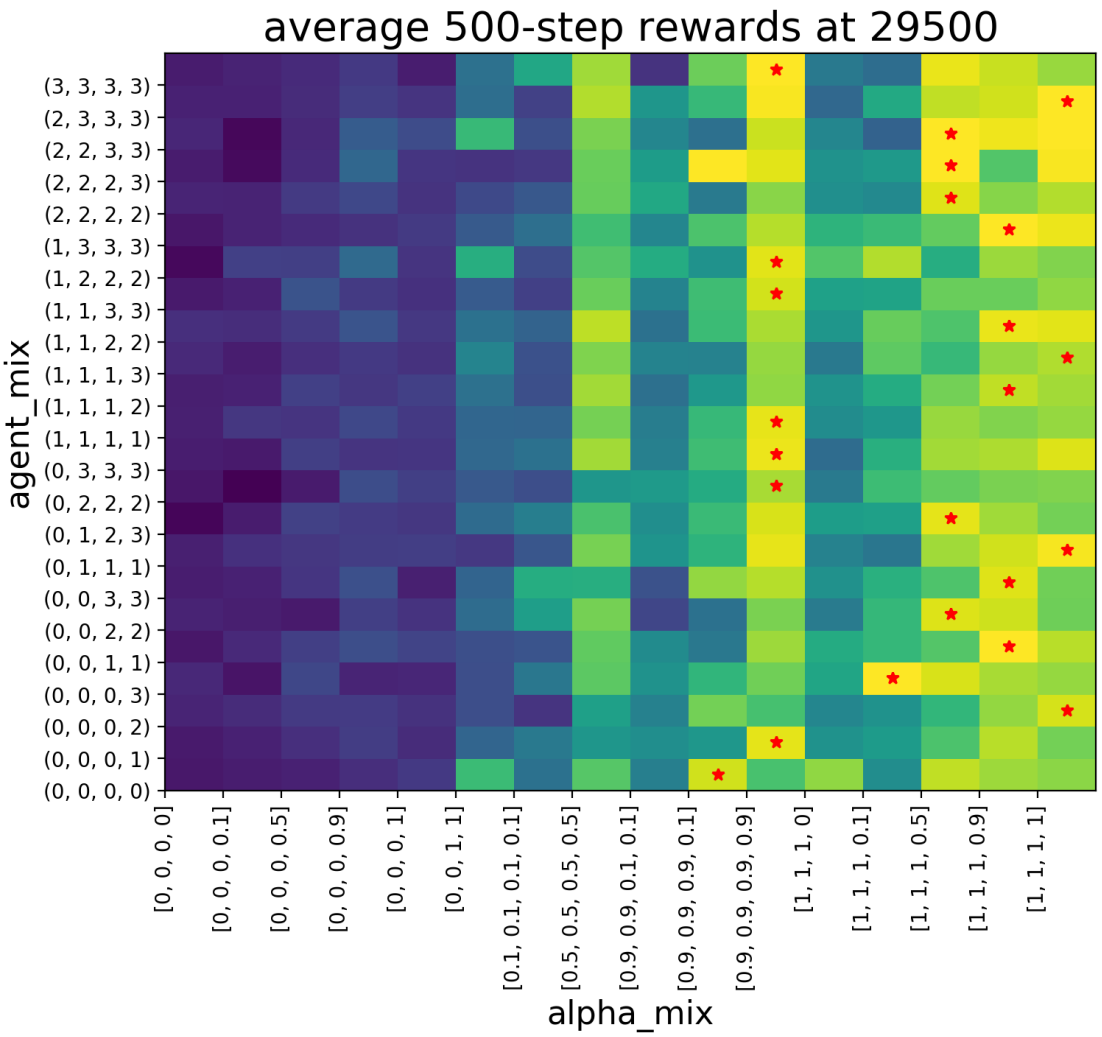

Figure 7.10: Heatmap of average rewards at the 29500-th step, smoothed using a 500 step window. The brighter the color, the higher the reward. X axis represents α coefficient configurations, ordered lexically. Y axis represents agent mixture configurations. Results are averaged over $n = 8$ repetitions per agent mixture and α coefficient configuration. A red star denotes the maximum over α coefficient configurations for each given agent mixture configuration.

fixing three agents leads to a significant speedup in learning (fixing two or three agents leads to a certain initial speedup but seems to lead to a somewhat worse final result). As for fixed policies learned in a selfish ($\alpha = 0$) setting, results are largely the same as in the homogeneous case (except for the case of fixing a single agent, interestingly).

Overall these results seem to suggest that in the context of this task, the ability of agents to adapt to one another is crucial, and fixing one or more agents may not always be helpful. This is particularly true if the extent of "selfishness" of the agents is unknown. Furthermore, it appears

injecting agents with preferences has a complicated impact on the learning dynamics.

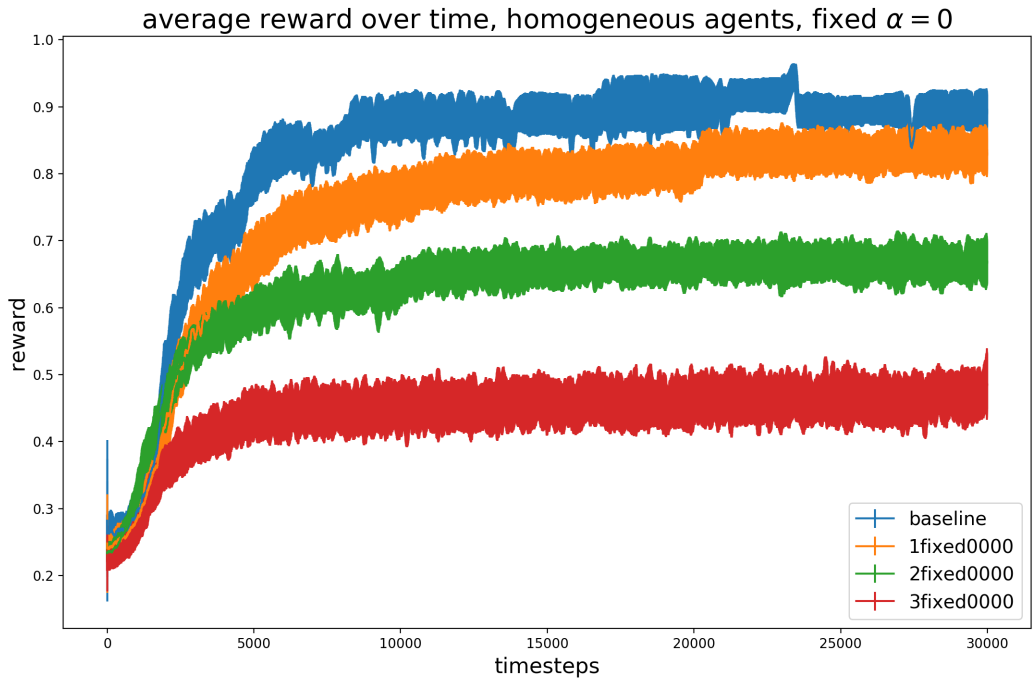

Figure 7.11: Average environment reward over timesteps in the chord generation domain. Agents are homogeneous, meaning their initial individual preferences are the same. Results averaged out over $n = 8$ repetitions. The results compare 4 configurations: i) baseline - four learning agents, fully collaborative ($\alpha = 1$); ii) 1fixed0000 - one agent is fixed with a policy trained in a fully selfish setting ($\alpha = 0$); iii) 2fixed0000 - two agents are fixed with policies trained in a fully selfish setting; iv) 3fixed0000 - three agents are fixed with policies trained in a fully selfish setting.

7.8 Extending the Results to a Second Domain: Predator-Prey

To establish that the approach presented in this chapter is both applicable and useful in other domains and in more general multiagent RL research, we study the same principles in a well-established multiagent RL task: predator-prey. In this section I will describe the predator-prey domain and the experimental setup used to examine our ideas in its context.

The predator prey domain [18] is a popular testbed of multiagent learning and cooperation, as it requires multiple autonomous agents to cooperate in order to achieve a shared goal.

In our configuration of the problem, there are four predator agents and a single prey agent inhabiting a 10×10 grid world. At each time step, the total of five agents can take one out of five possible actions: LEFT, RIGHT, DOWN, UP, corresponding to the four cardinal directions

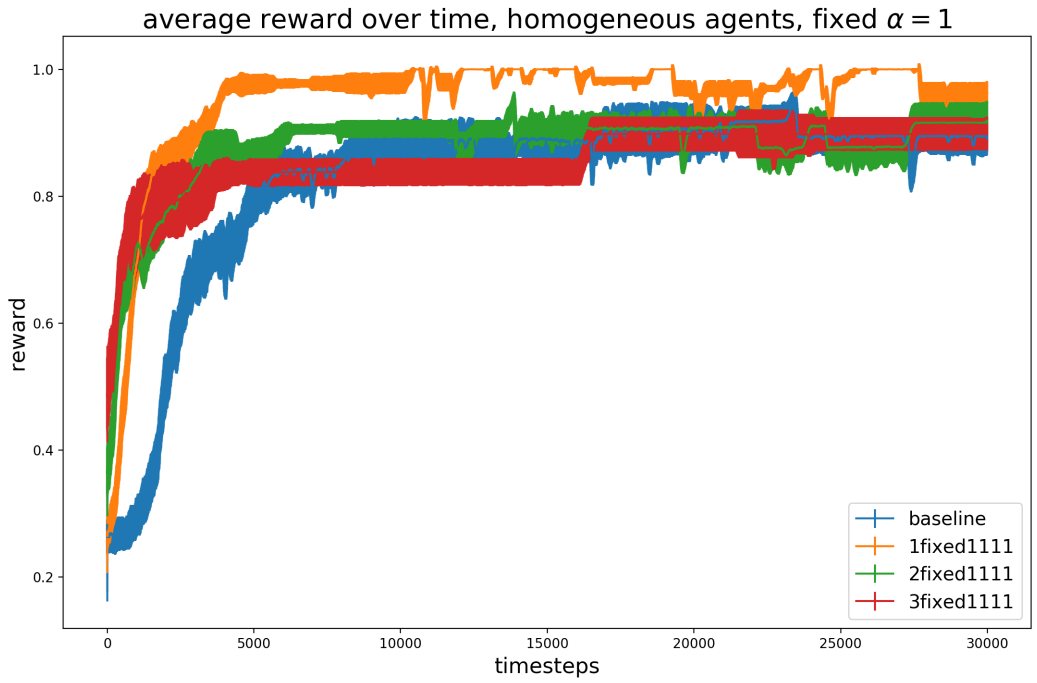

Figure 7.12: Average environment reward over timesteps in the chord generation domain. Agents are homogeneous, meaning their initial individual preferences are the same. Results averaged out over $n = 8$ repetitions. The results compare 4 configurations: i) baseline - four learning agents, fully collaborative ($\alpha = 1$); ii) 1fixed1111 - one agent is fixed with a policy trained in a fully collaborative setting; iii) 2fixed1111 - two agents are fixed with policies trained in a fully collaborative setting; iv) 3fixed1111 - two agents are fixed with policies trained in a fully collaborative setting.

on the grid, and NO-OP, which means the agent does not move that turn. The world is toroidal (i.e., wrapped around from all directions), meaning it has no boundaries, eliminating the resulting state-action transition edge cases.

An example environment can be seen in Figure 7.15.

The transitions are deterministic and take each agent 1 square in the direction it wants to move, unless it is already occupied, in which case there is no movement. Each episode starts with the 4 predators and 1 prey randomly located in the grid world. The prey's strategy is set to pick uniformly at random between the five possible actions.

The episode terminates after 100 steps, or if the predators manage to surround the prey completely. If the prey is surrounded by the predators they each get a positive reward of 1. If the episode ends without them catching the prey, they get a negative reward of -1 and all other steps have 0 reward. The environment has a discount value of $\gamma = 0.99$.

Our evaluation measure for comparing predator policies is the length of the episode. If the

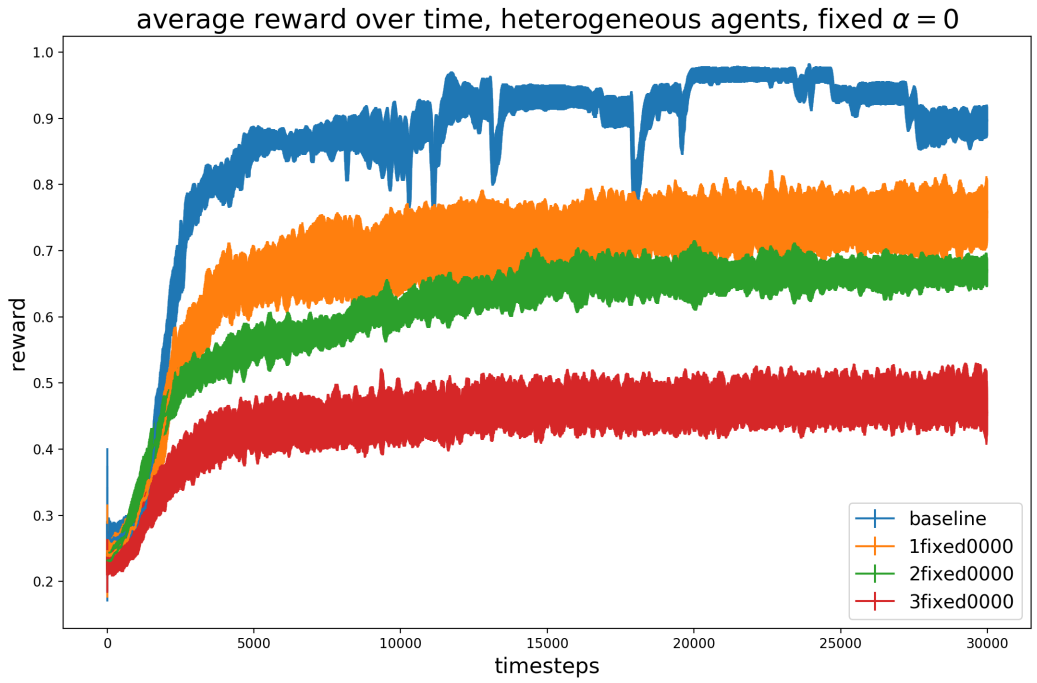

Figure 7.13: Average environment reward over timesteps in the chord generation domain. Agents are heterogeneous, meaning their initial individual preferences are different for each agent. Results averaged out over $n = 8$ repetitions. The results compare 4 configurations: i) baseline - four learning agents, fully collaborative ($\alpha = 1$); ii) 1fixed0000 - one agent is fixed with a policy trained in a fully selfish setting ($\alpha = 0$); iii) 2fixed0000 - two agents are fixed with policies trained in a fully selfish setting; iv) 3fixed0000 - three agents are fixed with policies trained in a fully selfish setting.

prey always gets away, this number will be 100. The more proficient the predators are at catching the prey, the lower this number will be.

In this chapter we consider each predator as having its own preference, or initial policy, that provides it with an individual reward signal that is separate from the environment reward signal which corresponds with success or failure in the shared task of capturing the prey. We model this personal reward to only take into account the single predator's positions and the prey's position, and not the positions of any of the other predators. We do this so that even if all 4 agents follow the same preference, they still need to learn how to coordinate based on the environment reward.

As discussed in Section 7.2, this mixing of reward signals is tunable, and in Section 7.9 we show how different mixing proportions alter the learning dynamic. In the next subsection we describe the predator preferences models in detail.

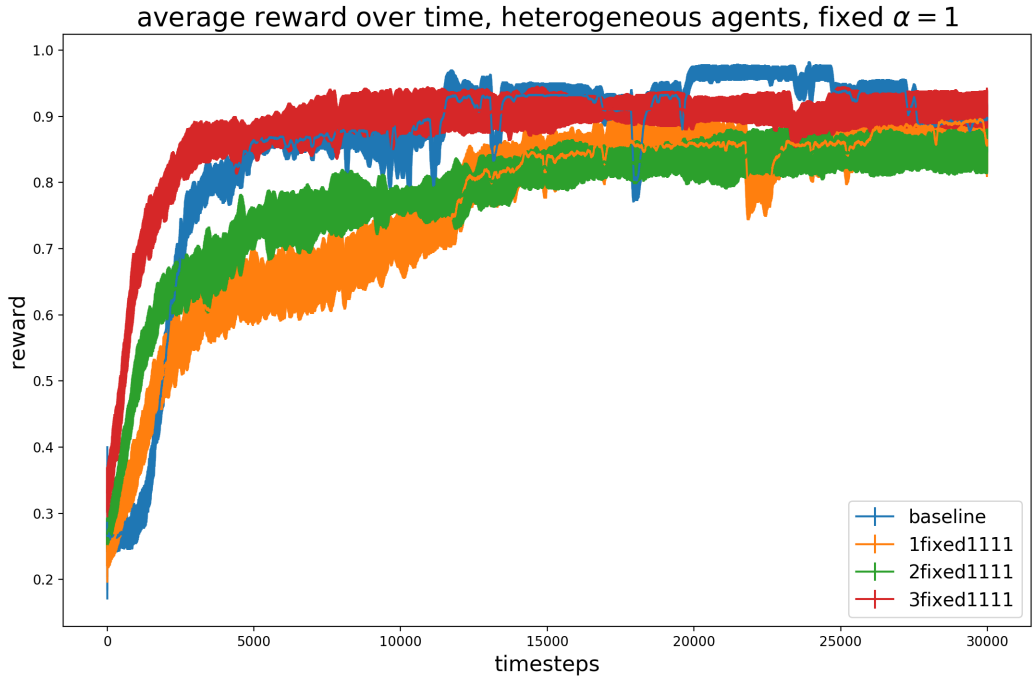

Figure 7.14: Average environment reward over timesteps in the chord generation domain. Agents are heterogeneous, meaning their initial individual preferences are different for each agent. Results averaged out over $n = 8$ repetitions. The results compare 4 configurations: i) baseline - four learning agents, fully collaborative ($\alpha = 1$); ii) 1fixed1111 - one agent is fixed with a policy trained in a fully collaborative setting; iii) 2fixed1111 - two agents are fixed with policies trained in a fully collaborative setting; iv) 3fixed1111 - two agents are fixed with policies trained in a fully collaborative setting.

7.8.1 Preference Models

The type of preference models are derived from the following basic predator strategies:

- A *random* policy that takes one of the 5 actions randomly. This strategy does not catch the prey, leading to an average length of 100.
- A *greedy* policy that takes the step that takes the agent closest to the current location of the prey. If more than one such step exists the agents picks randomly. This strategy takes 91.62 steps to capture a prey on average.
- A *surround* policy which takes the action which would lead it to a neighboring square to where the prey is. Each predator employs the Hungarian algorithm[194] to find an optimal matching between the four current predator positions and the positions required to surround

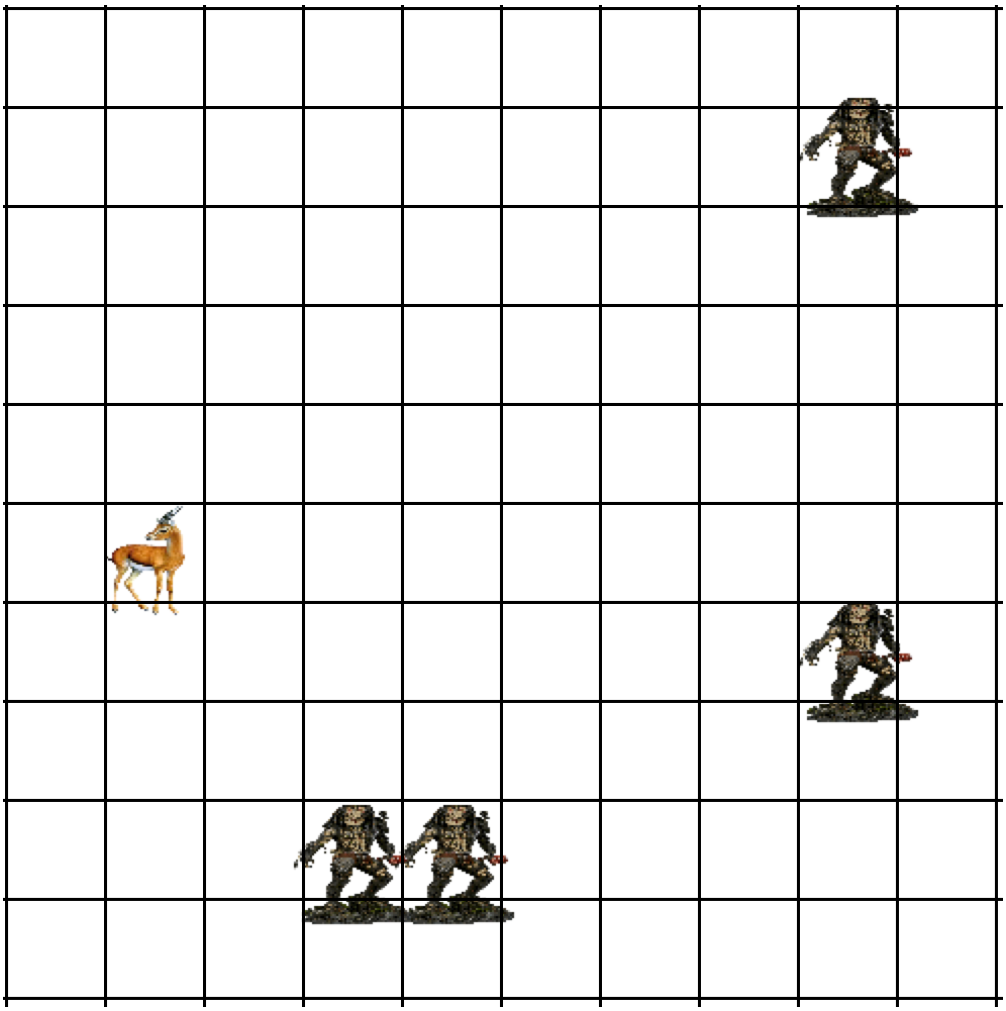

Figure 7.15: An illustration of the Predator Prey domain.

the prey. This policy is aware of other agents and hence does the best among the strategies we consider here. It takes 38.97 steps on average to catch the prey.

- A *probabilistic destinations* policy which attempts to tighten a circle around the prey. It favors destinations that are both nearer to the prey and to itself, but may choose farther destinations to prevent getting stuck on other predators and dealing with a moving prey [**?**]. This strategy does better than greedy, taking 84.38 steps on average to catch the prey.

To determine the efficiency of these predator strategies, we use the average time-steps before the end of the episode as a scoring measure. However, we compare policies by the average number of steps before the agents capture the prey. If the prey escapes the predators every time, this metric leads to a score of 100. Better policies would catch it more often and end up with a lower average number of steps to capture the prey, and resulting score.

The baseline predator policies we compare to are the following:

- A greedy policy that takes the step that takes the agent closest to the current location of the prey (denoted "greedy"). If more than one such step exists the agents picks randomly.
- A policy that takes a random step with ϵ probability and takes the greedy action otherwise ($\epsilon-$greedy).
- A policy that tries to surround the prey by taking the action which would lead it to a neighboring square to where the prey is (surround). In effect, this policy employs the Hungarian algorithm respect to the 4 bins neighborhood of the prey, taking the step that will take it closest.

We run each of these strategies for 100,000 steps, and consider the above score. If all the predators employ the greedy strategy, they get an average score of 86.28, $\epsilon-$greedy with $\epsilon = 0.3$ gets 45.94, and surround gets 98.13, meaning it almost never catches the prey. This is in contrast with a completely random policy, which has a score of 82.22. It should be noted that the reason that the $\epsilon-$greedy policy does so well, or why the random policy is better than a greedy policy is that we have formulated the prey policy in such a way that it does well against predator approaches that make a beeline for the prey. Random policies tend to place the agents in different parts of the state space, leading to the prey being more susceptible to capture.

7.8.2 Learning Architecture

Every agent policy is parameterized as a multi-layer perceptron with 2 layers of 500 units each. In addition, each agent also has a Discriminator D_k which is made up of another 2-layer network with 64 units in each layer. We use Tanh activations for the policy and ReLU activations for the Discriminator.

7.9 Results in the Predator-Prey Domain

In the context of these experiments, as described above, each predator agent learns its initial preference policy via behavioral cloning. The policy that generates these examples is the greedy policy, which always takes each predator closest to the prey. However, the set of demonstrations used to learn the preference is different for each agent, implying they learn somewhat different, albeit overlapping, individualized preference policies.

We use the greedy policy as the preference, as it does not depend on other predators and is a fairly intuitive preference. We generate 10000 (state, action) sequences by executing the greedy policy. 2000 iterations of supervised imitation are performed to bias the agents to their preference policy. With these experimental settings, the log-likelihood plateaus at about $-.2$ after this number of pretraining steps (giving a likelihood of about .82 to the expert rollout action on average).

Figures 7.16 - 7.18 present the effects of different preferences and different mixing weights at three different stages in training. We see that preferences for policies that are related to the global

objective help to learn initially (Figure 7.16), seen by the brighter bands across all α configurations. This is especially true when we consider the 'greedy' (indexed by 1) or 'surround' (indexed by 2) preferences and their combinations. On the other hand preferences that are not helpful to the overall task do not assist the agents in learning initially. The 'random' policy, indexed as 0, is not helpful for the agents in their learning (Figure 7.18).

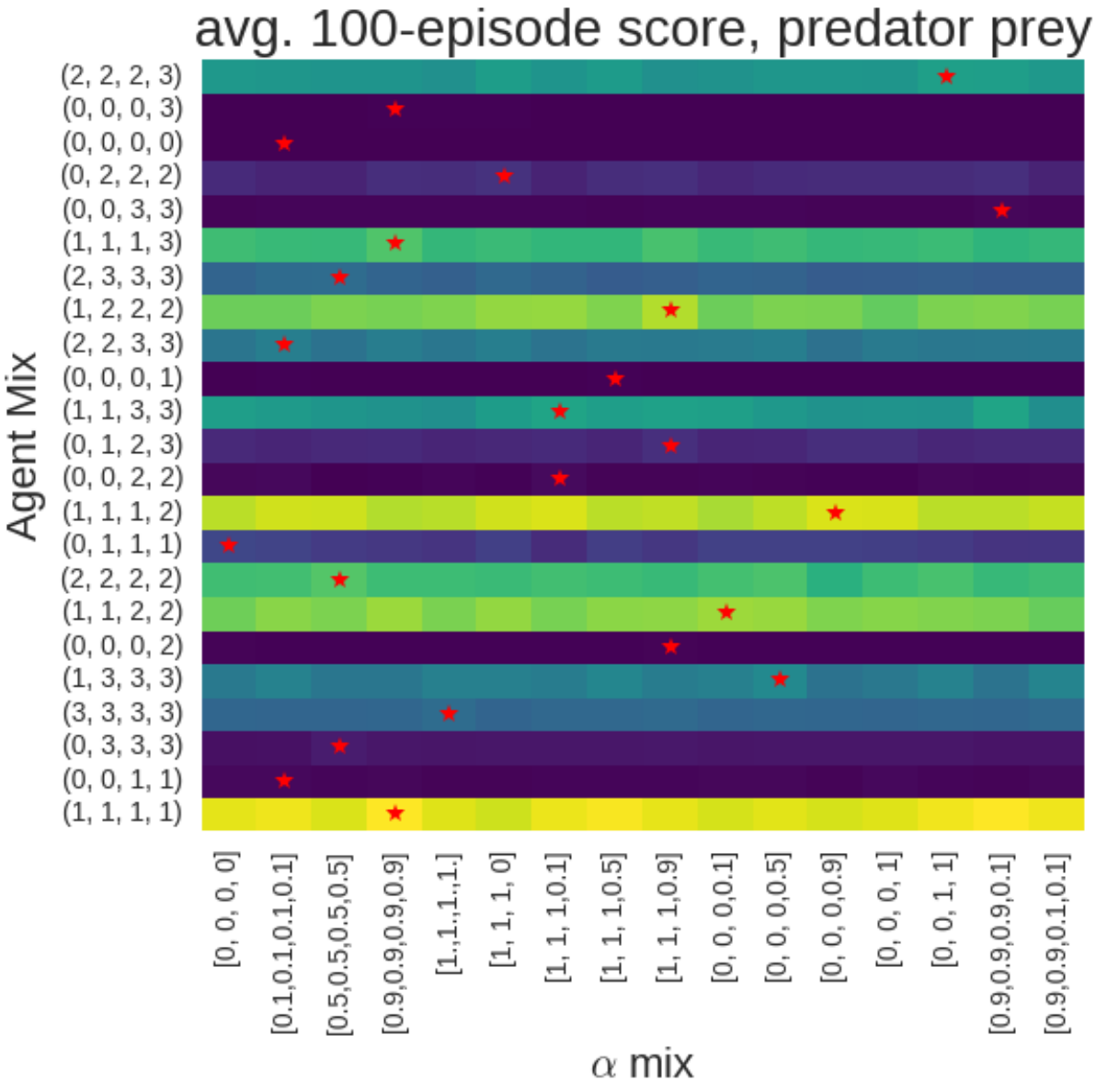

Figure 7.16: Comparison of different preferences and mixtures at 500 learning steps

The red asterix in each column marks the α configuration that works best for preference

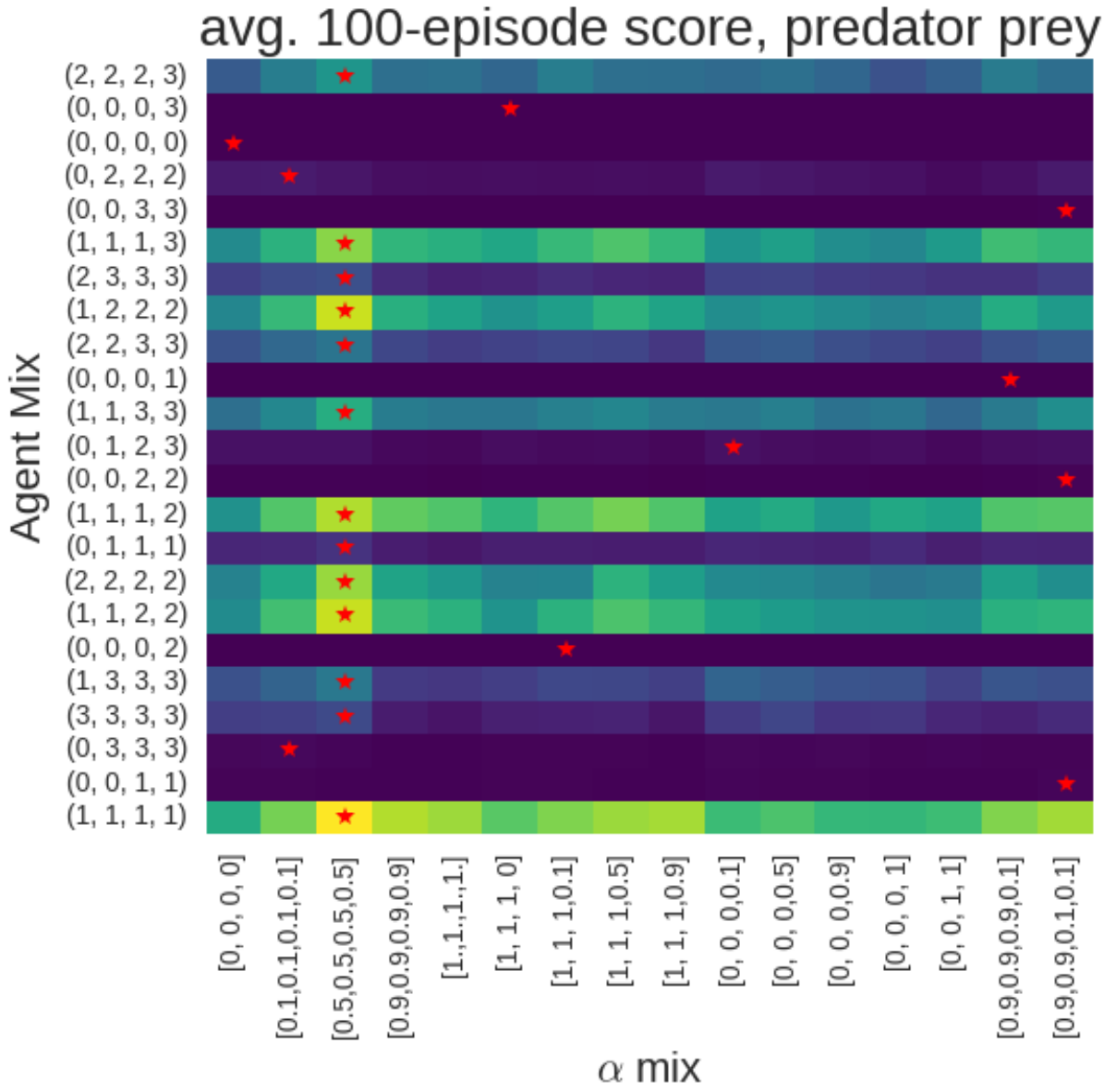

Figure 7.17: Comparison of different preferences and mixtures at 16500 learning steps

combinations at that time step. As we go through time, we see that the α weights that balance the individual preferences with the environment reward (with $\alpha = 0.1$ or $\alpha = 0.5$) lead to the best performance on the shared task (Figure 7.18).

We see that these α values do well across a variety of preference combinations. These results suggest that it is useful to use individual preferences to guide agents towards their own preferred policies and use the environment reward for coordination and fine-tuning. More importantly, as

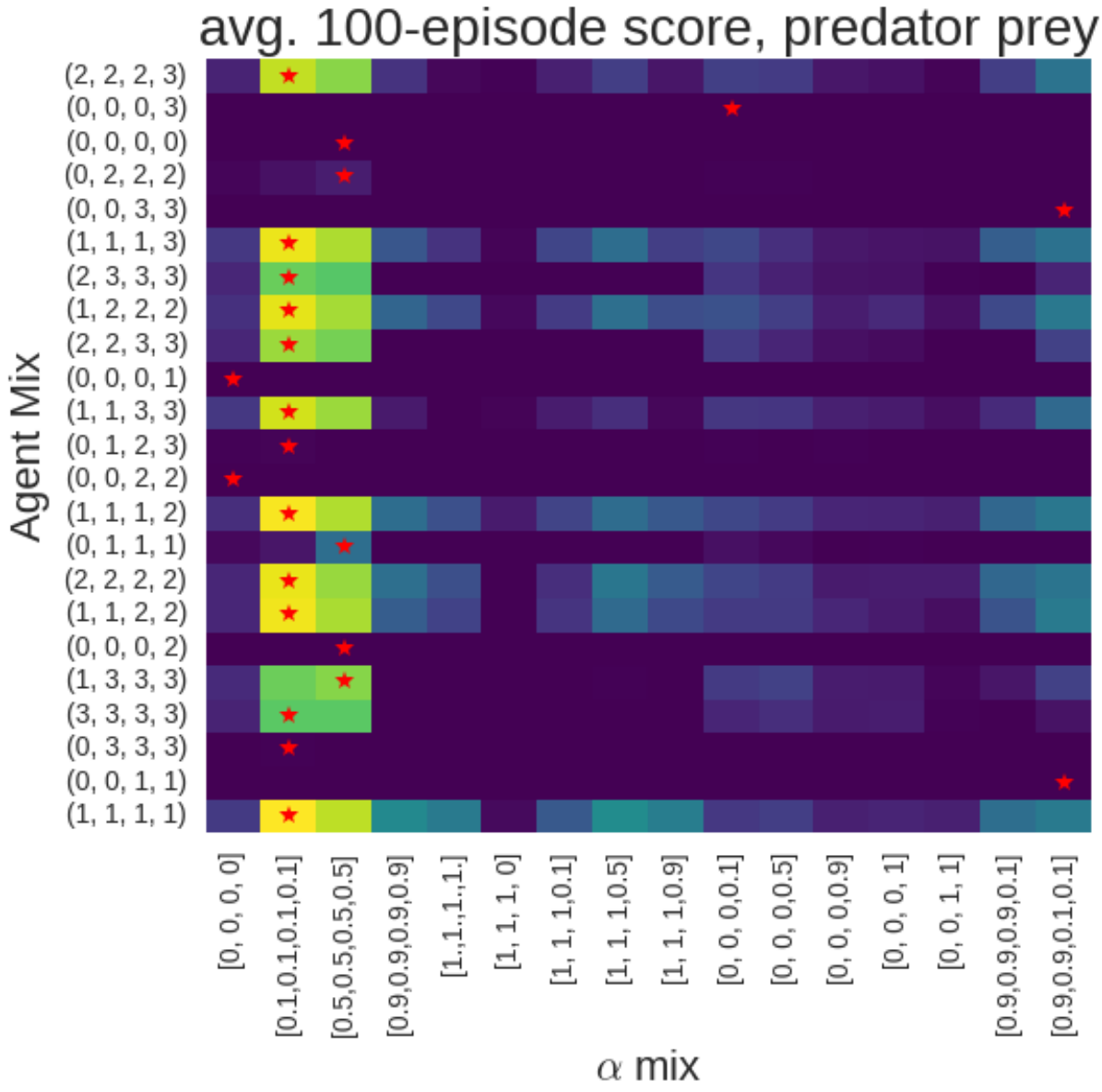

Figure 7.18: Comparison of different preferences and mixtures at 99000 learning steps

observed in the chord generation domain (Section 7.6), the fully selfless α mix of $\{1, 1, 1, 1\}$ is not consistently optimal. This observation serves as further evidence that interjecting various forms of preferences can prove beneficial to team performance.

7.10 Summary and Discussion

In this chapter, I presented a novel multiagent RL framework that considers the case in which multiple agents need to learn to collaborate on a shared task while actively pursuing (to varying degrees) a separate, individualized reward signal. I subsequently proposed a paradigm for initializing individualized preferences and combining these individualized reward signal with the shared task reward via a tunable mixing parameter α.

The empirical results presented in this chapter indicate, somewhat surprisingly, that actively pursuing an individualized reward signal in addition to the shared task reward can in fact be advantageous for shared task performance. This outcome is nontrivial since there was no a priori reason to assume that combining two reward signals, that are only partially correlated, would yield improved performance with respect to the shared task signal. Our hypothesis is that introducing an individualized component to the reward signal helps induce directed exploration by pushing all the agents towards a smaller region of the policy space. Another factor that can largely explain the improvement is that the intrinsic reward signal is provided as an adversarial imitation signal, that is, by design, considerably less sparse than the environment reward signal, thus providing a richer learning signal, even if the signal is only partially correlated with shared task performance.

In the context of multiagent music generation, the ability to effectively balance and even leverage existing preferences is essential for any setting in which we envision agents and people creating music jointly. Therefore the research presented in this chapter takes a meaningful step towards this goal, in further fulfillment of Contribution 4 of this thesis. Furthermore, by extending this approach to the general multiagent RL domain of predator-prey, I highlight the applicability and utility of insights and approaches derived from musical tasks and environments in general RL research, an addition to Contribution 6 of this thesis.

In the next Chapter I will contextualize the research done in this thesis in the broader context of the intersection of AI research and music, and delineate a taxonomy of music AI tasks.

8. Related Work and a Taxonomy of Musical Intelligence Tasks

In order to better understand the contributions of this thesis it is important to put them in context. Given the breadth and complexity of research work at the intersection of AI research and music informatics, a need arises to find useful ways of breaking down and organizing previous research. Indeed, in this section I will not only introduce a broad overview of the related literature, but also propose a unified framework for parsing this varied and complex body of work, delineating a taxonomy for music AI tasks, and mapping out the overall state of the art. In doing so, this chapter constitutes Contribution 1 of this thesis.

Since its emergence in the 1950s, artificial intelligence has become an ever more prevalent field of scientific research. Technology to which we may assign varying degrees of intelligence is virtually all around us – from from sophisticated navigation systems[96] and anti-collision sensors placed on cars [418] to recommender systems meant to help us pick a book or movie[5]. However, while great emphasis has been placed on improving the performance of such systems, other meaningful facets have not been as thoroughly explored. Such additional facets cover a wide array of complex mental tasks which humans carry out easily, yet are hard for computers to mimic. These include the human ability to understand social and cultural cues, to interpret and infer hidden meanings, to perceive the mental state of their counterparts, and to tailor their responses accordingly. A prime example for a domain where human intelligence thrives, but machine understanding is limited, is music.

In recent years, the application of algorithmic tools in cultural domains has become increasingly frequent. An interesting example is Sisley the Abstract Painter, a project aimed to algorithmically emulate modern paintings of varying abstraction levels, given an input photograph[433]. Another example uses visual recognition tools to study what makes the architectural styles in different cities distinctive[88]. A more mainstream example for the application of machine learning in a cultural domain can be seen in a recent paper in which 16 episodes from the famous TV series Lost are automatically tagged for character presence using weakly supervised data [71]. In the domain of natural language processing, many works have used literary texts as input data, and some works have cultural goals such as document style classification [14], authorship attribution [369], and literature analysis [181]. A theme that surfaces from examining this type of research is that tools from the AI and machine learning communities often reveal new insights and shed new light on fundamental cultural questions – what characterizes an author (or an architect); which geometric properties best separate Kandinsky from Pollock (Or Steven Spielberg from Stanley Kubrick); is it possible to chart the evolution of Latin dance styles, etc. Another important observation is that such cultural domains may often prove excellent testbeds for new algorithmic tools and approaches.

There are many ways in which artificial intelligence and music intersect, ranging from analysis of large bodies of existing music to the creation of music itself. Computers have accompanied both the analysis and creation of music almost since they first came into existence. In 1957, Ljaren Hiller and Leonard Isaacson developed software that applied Markov chains and rule-based logic to compose a string quartet [146]. Iannis Xenakis used computers in the early 1960s to generate numerical patterns, which he later transcribed into sheet music [420], and later led the development of the first music programming language, the Stochastic Music Programme (SMP) [419]. A decade later, Pierre Boulez founded IRCAM (Institut de Recherche et Coordination Acoustic/Musique), where composers, computer scientists, and engineers study music and sound and invent new tools for creating and analyzing music [43]. Only a few years after its establishment, IRCAM already served as the home of the Spectralist movement, where composers such as Gerard Grisey and Tristan Murail used computers and spectral analysis to compose new music [10]. Since then, the notion of applying artificial intelligence to create music has remained of interest to many, and there are many other examples for this type of composition, ranging from stochastic generation tools and elaborate mathematical models to grammar-based generation and evolutionary approaches [166, 276, 317].

Another recent body of work lying at the intersection between artificial intelligence and music analysis is that of the music information retrieval (or MIR) community. Over the last decade, many researchers have applied computational tools to carry out tasks such as genre identification [90], music summarization [246], music database querying [107], melodic segmentation [307], harmonic analysis [55], and so on. Additional research questions with implications for preference learning and computational musicology include (but are not limited to) performance analysis and comparison [217], music information modeling[66], music cognition[191], and surprise[2].

Indeed, the study of music perception within the cognitive science community has also served as a bridgehead between computational learning research and music analysis. Considerable effort has been put into using algorithmic tools to model patterns of psycho-physical responses to music stimuli [174], and the interaction between musical concepts and their interpretations in the brain[191]. Another related field of study is that of human-computer interaction and human-robot interaction. Previous work has been carried out in order to provide AI with the ability to interact with humans in one form of social setting or another[78]. These works, however, usually do not capture the complexity of human interaction, and more importantly, rarely take into account the complex array of pre-existing cultural knowledge that people "bring to the table" when they interact socially, or the cultural information they accrue through interaction.

The separate fields and perspectives on music informatics, spanning music information retrieval, cognitive science, machine learning and musicology, have largely proceeded independently. However, they are all concerned with overlapping facets of what we define in this chapter as "musical intelligence", specifically in the context of artificial intelligence. To define something as complex and as abstract as "musical intelligence" is at least as difficult as defining what intelligence is in general - a notoriously slippery and tenuous endeavor. However, for the purposes of this chapter,

we adopt the following working definition:

> "Musical Intelligence", or "music understanding", describes a system capable of reasoning end-to-end about music. For this purpose, it needs to be able to reason at different levels of abstraction with respect to music; from perceiving low-level musical properties, to intermediate levels of abstraction involving the organizational structure of the music, to high level abstractions involving theme, intent and emotional content.

The breakdown of musical abstractions as "low-level", "intermediate" and "high-level" is rather murky. Nonetheless, we can consider basic auditory properties regarding the overall spectrum, tempo, instrumentation etc. as the lowest levels of music understanding. Intermediate levels of abstraction include concepts such as identifying melody vs. accompaniment, identifying the functional harmonic structure of musical segments, identifying recurring motifs, or placing the music in broad genre terms. High-level abstractions include more principled motific and thematic analysis, understanding the intended emotional valence of various pieces of music, the interplay between different structural and motific choices, drawing connections between different pieces of music, recognizing the style of individual musicians, being able to successfully characterize the musical tastes of others, and ultimately, being able to generate musical sequences that people would consider meaningful.

While somewhat analogous to the notion of scene understanding in machine vision [209], musical intelligence is a much more elusive term, given that the "objects" music deals with are considerably less concrete or functionally defined than those usually considered in computer vision. Nonetheless, the definition above is useful in providing a high-level motivation and goal for connecting disparate aspects of music informatics research.

The purpose of this survey chapter is threefold. First, it is meant to serve as an updated primer for the extremely large and interdisciplinary body of work relating to artificial musical intelligence. Second, it introduces a detailed taxonomy of music related AI tasks that is meant to provide a better perspective on the different achievements made in the intersection of both worlds in the past two decades. Third, this survey chapter analyses different evaluation methods for various music-related AI tasks.

Due to the enormous literature that is relevant to this survey, we limit its scope in several ways. We focus on works that involve a significant machine learning or artificial intelligence component. We focus on several potential representations of music, either symbolic or at the audio level, and consider tasks primarily involving this input. While we acknowledge the large body of work which focuses on the signal-processing and audio-extractive aspects of automated music analysis, we do not consider it a core part of this survey, and only reference it to the extent that such work lies at the heart of many of the feature extraction procedures used in machine learning frameworks for music related tasks. Another large body of work, which focuses on natural language processing of song lyrics, music reviews, user-associated tags etc. is also considered outside the scope of

this chapter. We also consider automated sheet music recognition (traditionally through image processing techniques) as outside the scope of this chapter.

The structure of this chapter is as follows: in Section 8.1 we discuss the target audience of this chapter, and provide initial background for reading this chapter. We proceed to discuss the motivation behind music-related AI research and its potential uses. In Section 8.2 we focus on the proposed taxonomy and break down the extensive body of literature into different categories, utilizing different perspectives. Subsequently, in Section 8.3 we review the literature from the perspective of the tasks that have been studied. In Section 8.4 we discuss the different types of representations that have been used in the literature. In Section 8.5 we break down the extensive list of AI techqniques that have been applied in music AI research. Section 8.6 discusses the different evaluation methods used in the literature to assess the effectiveness of proposed approaches. Lastly, in Section 8.7 we summarize the contributions of this survey, consider the idea of artificial musical intelligence from a broader perspective, and discuss potential gaps existing in the literature.

8.1 Background and Motivation

Beyond its purpose as an overview of related work for this thesis, the survey presented in this chapter is aimed at computer scientists working in AI who are interested in music as a potential research domain. Since both the study of music and the artificial intelligence and machine learning literature are too extensive to be covered by any single survey paper, I assume the reader has at least some knowledge about the basic machine learning paradigm (e.g. supervised vs. unsupervised learning, partition to training and testing data, evaluative metrics for learning algorithms such as area under the ROC curve etc). I also assume some familiarity with various learning architectures and algorithms, such as regression, support vector machines, decision trees, artificial neural networks, probabilistic and generative models, etc. From a more classical AI viewpoint, some understanding of knowledge representation, search and planning approaches is assumed, but is not directly relevant to a large component of this paper. Good introductory sources for machine learning and AI concepts can be found in various textbooks (such as [340]).

Regarding music terminology, I assume familiarity with a few basic terms. These terms include pitch, note, scale, key, tempo, beat, chord, harmony, cadenzas and dynamics. I provide brief explanations for these terms and more in Appendix 9.5. Further details can be found in sources such as The Oxford dictionary of musical terms [198], among many others. Throughout the chapter we will assume the general meaning of these terms is known.

I believe that work at the intersection of artificial intelligence and music understanding is beneficial to both communities on multiple levels. As a rich, complex research domain, we expect that the study of artificial musical intelligence will potentially produce fundamental scientific discoveries, as well as engineering insights and advances which could be applicable in other domains. These expectations are supported by the following lines of reasoning:

- **Music is a quintessential form of intelligence:** Music, being intrinsically complex and multifaceted, involves some of the most sophisticated mental faculties humans have. Musical skills such as playing, analyzing or composing music involve advanced data analysis, knowledge representation and problem solving skills. The challenge of developing such skills in artificial agents gives rise to interesting research problem, many of which are transferable to other application domains (such as analyzing video or interactive gameplay). Furthermore, some abstract issues such as modeling a "utility function" that captures a person or a group's enjoyment of different types of musical information are in fact inherent to any attempt to quantify aesthetic value, mass appeal or creative content. Advances in the modeling of such a function would have immediate applications in any case where understanding "what people want" is key to good performance but no easily quantifiable objective functions exist.

- **Music is inherent to the human experience, and therefore to social interaction:** If we envision a future where intelligent artificial agents interact with humans, we would like to make this interaction as natural as possible. We would therefore like to give AI the ability to understand and communicate within cultural settings. This issue has actual benefits, as software capable of tailoring its behavior to the tastes and the preference of specific people would do better both in understanding the behavior of its human counterpart and influence it, leading to a much more successful interaction.

- **Deeper music AI will lead to better performance of real world systems:** Let us consider a recommendation system for music. Such a system would greatly benefit from the ability to model the intrinsic properties of the music it deals with, rather than solely rely on statistical correlations or simplistic measures. This capacity would also enable recommendation models to learn with less input data, thus ameliorating the infamous cold start problem in recommender systems. The work presented in Chapter 3 is an example for this approach. The architecture presented in that work is able to learn some basic signal of what a person likes based on very little experience by directly mapping musical properties of songs and transitions to predicted human preferences.

- **AI can lead to new cultural insights:** The intersection of artificial intelligence and music often leads to insights regarding music, how it is perceived by humans, and what makes it unique. These observations have significant cultural value, and are of interest to many researchers in a wide range of disciplines.

While admittedly the study of musical intelligence can be seen as somewhat more esoteric than other core academic disciplines and application areas, and the assessment of musical quality is inherently subjective, to those concerned about such issues we offer the following observations:

- **Widespread commercial interest:** The market for music recommendation, for instance, is

large[13], and growing. Video games such as Rocksmith[14] , which automatically analyzes actual instrument playing to provide feedback and accompaniment, are also growing in popularity. The commercial success of such applications reflects a strong industrial interest in research that enables better autonomous music understanding.

- **Widespread academic interest:** In the past two decades, there have been hundreds of papers at the intersection of AI and music published in top tier conferences and journals (including those which we discuss in this chapter), with thousands of citations, cumulatively. This fact in itself serves as evidence for the existing interest for such work across varied research communities.

- **Realizable goals exist:** While the subjectivity inherent to music may pose difficulties in evaluating the performance of various music AI systems, many inter-subjective goals (such as increasing user satisfaction and engagement, or better matching people's perceptions and expectations) can be effectively evaluated using lab experiments and crowd-sourcing.

8.2 A Taxonomy of Music AI Problems

Consider a song by the Beatles, or a piano trio by Beethoven. What kinds of computational research questions can we ask about these cultural artifacts? What kinds of tasks might we expect intelligent software to perform with respect to them?

Due to the complexity and richness of the music domain, countless different perspectives can be assumed when studying the intersection of music and artificial intelligence. Different perspectives give rise to different research questions and different approaches. In order to compare and contrast the literature using a consistent and unified language, I introduce the following dimensions along which each contribution can be placed:

- The target task
- The input type
- The algorithmic technique(s)

In this section I will broadly outline these three perspectives, which together span the taxonomy introduced in this chapter. A visual representation of the proposed taxonomy is shown in Figure 8.1

8.2.1 Partition by the Nature of the Task

There is a wide variety of potential research tasks we might concretely try to accomplish in the music domain. We use the term "task" to describe a small, concrete and well-defined problem. For

[13] http://techcrunch.com/2015/01/21/apple-musicmetric
[14] http://en.wikipedia.org/wiki/Rocksmith

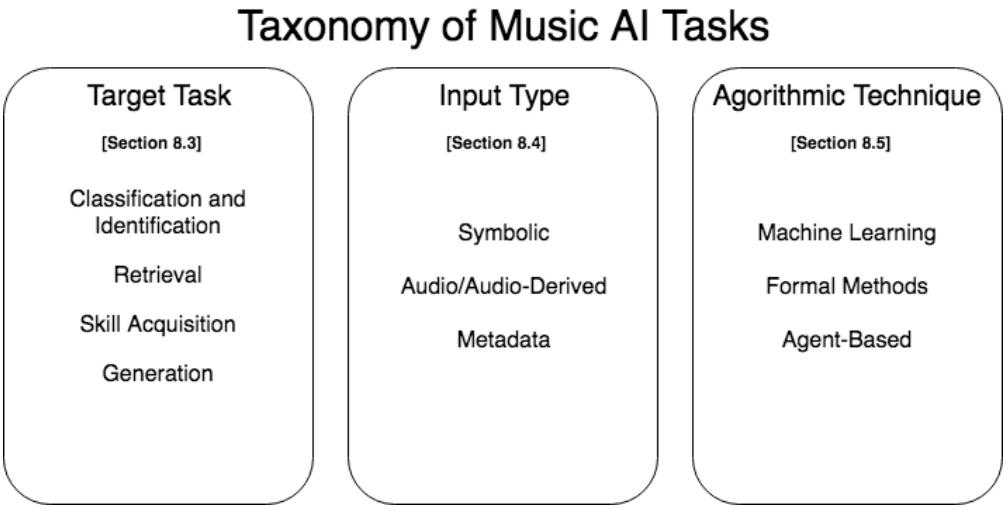

Figure 8.1: Visual high-level illustration of the proposed taxonomy.

instance, in the Beatles song example above, we may wish to discern the chorus from the refrain, or identify the beat and the key of the song, or identify whether it is an early vs. a late Beatles song. While these are all small and concrete tasks, they are not atomic or independent; knowing the key and the beat of a song is relevant both to determining its structure (chorus vs. refrain), identifying which sub-genre it belongs to, etc.

To better understand shared themes across tasks and facilitate a more helpful overview of the current literature, we break tasks down to several categories:

1. Classification and Identification - any tasks which associates musical segments with one or more out of a closed set of labels. For example, classifying pieces by composer and/or genre.

2. Retrieval - as in the broader case of information retrieval, these tasks involve obtaining relevant items, often ranked by relevance, from a music dataset. A typical example is a recommender system that suggests specific songs to a specific user given his listening history.

3. Musical Skill Acquisition - this category encompasses the varied set of basic analysis skills required for music processing, from pitch and tempo extraction to chord recognition.

4. Generation - these tasks involve some facet of creating new musical expression, ranging from generating expressive performance from audio, generating meaningful playlists by sequencing existing songs, and, probably the most elusive of all, generating new music.

These categories aren't mutually exclusive, as many actual tasks might share more than one aspect, or contain components that belong in other categories. Still, we believe it is a natural way to group tasks in a way that sheds light on recurring themes and ideas.

8.2.2 Partition by Input Type

It is almost always the case that the type of input dramatically affects the range and complexity of tasks which can be performed on that input. Generally, there are three input categories –

1. Symbolic Music Representations - these are the simplest and easiest to analyze, as they capture pitched event information over time. Variants of symbolic representation range from the ubiquitous MIDI protocol [231] to complex digital representation of common practice music notation.

2. Audio (and audio-derived) Representations - this category of representations ranges from raw unprocessed audio to compressed audio to concise spectral features, depending on the level of reduction and abstraction.

3. Related Musical Metadata - all non-audio information regarding a musical piece, ranging from artist and song names to associated mood tags, social media information, lyrics, occurrence history etc.

In this survey we will focus on the first two representations, but due to its ubiquity, we will occasionally refer to the third type.

8.2.3 Partition by Algorithmic Technique

A wide variety of machine learning and artificial intelligence paradigms and techniques have been applied in the context of music domains. From a machine learning and artificial intelligence research perspective, it is of interest then to examine this range of techniques and the specific musical domains where they were applied. Due to the extensive nature of the related literature and the wide range of musical tasks where the following methods have been used, this list is not intended to be entirely comprehensive. To the best of our knowledge, however, it is representative of the full array of methods employed. Broadly speaking, we consider the following general technical approaches:

1. Machine Learning Approaches - a wide range of machine learning paradigms has been employed for various music informatics tasks. The list of techniques used is as varied as the machine learning literature itself, but some examples include methods such as support vector machines (SVM) [140], generative statistical models such as Hidden Markov Models (HMM) [318], Markov Logic Networks (MLN) [333], Conditional Random Fields (CRF) [195], and Latent Dirichlet Allocation (LDA) [39]. In recent years, deep neural network architectures such as Convolutional Neural Networks (CNN) [202], Recurrent Neural Networks (RNN) [132], and Long Short Term Memory networks (LSTMs) [150] have become increasingly popular and ubiquitous in the music informatics literature.

2. Formal methods - multiple attempts have been made to employ formal techniques, similar to the formal methods subfield in computer science, to handle music informatics tasks via formal specification methods to describe and generate musical sequences. Under this umbrella one may find approaches inspired by generative grammars [166], formal specification of tonal and chordal spaces with production rules [80], probabilistic logic [405], and fuzzy logic [432].

3. Agent-based techniques - multiple papers in the music AI literature have studied complex approaches that go beyond the scope of a learning algorithm or a formal specification, but rather fall in the subfield of intelligent agent research. That is to say, this category deals with AI systems that combine perception and decision-making in a nontrivial manner to handle musical tasks. In this category one may find examples such as person-agent accompaniment and improvisation systems [382], robotic systems for music performance [151], multiagent music generation architectures [38], and reinforcement learning agents for music generation [67].

Having outlined the general structure of the taxonomy proposed in this chapter, we can now delve more deeply into each category and provide examples for the varied types of questions and approaches studied in the past 15 years, following the rise of online music platforms and medium-to-large-scale music datasets. In the next sections we consider each dimension of the taxonomy separately and overview key examples in each partition category.

8.3 Overview of Musical Tasks

The first aspect through which we examine the literature is the functional one - which musical tasks have been addressed via machine learning and artificial intelligence approaches? Following our taxonomy from Section 8.2, we partition tasks into four main groups - classification and identification, retrieval, skill acquisition, and generation. A visual summary of the content surveyed in this section is provided in Figure 8.2.

8.3.1 Classification and Identification Tasks

Suppose we are presented with a newly unearthed score for a classical piece. This score, it is claimed, is a lost cantata by Johann Sebastian Bach, one of the many assumed to have been lost to posterity. Is this really lost music by the great Baroque master? Or perhaps the work of a talented imposter? Was it written in Bach's time? Or is it a recent work of forgery? These questions may seem hypothetical, but they are actually quite real, for instance in the case of several organ fugues by J.S. Bach [403]. An even more famous example involving J.S. Bach, one that most music students should be familiar with, is that of Bach's famous liturgical chorals. Of these 327 chorals, which have been widely used to teach traditional harmony and counterpoint for more than two centuries, only a third have definite sources in known Bach cantatas. The others are without

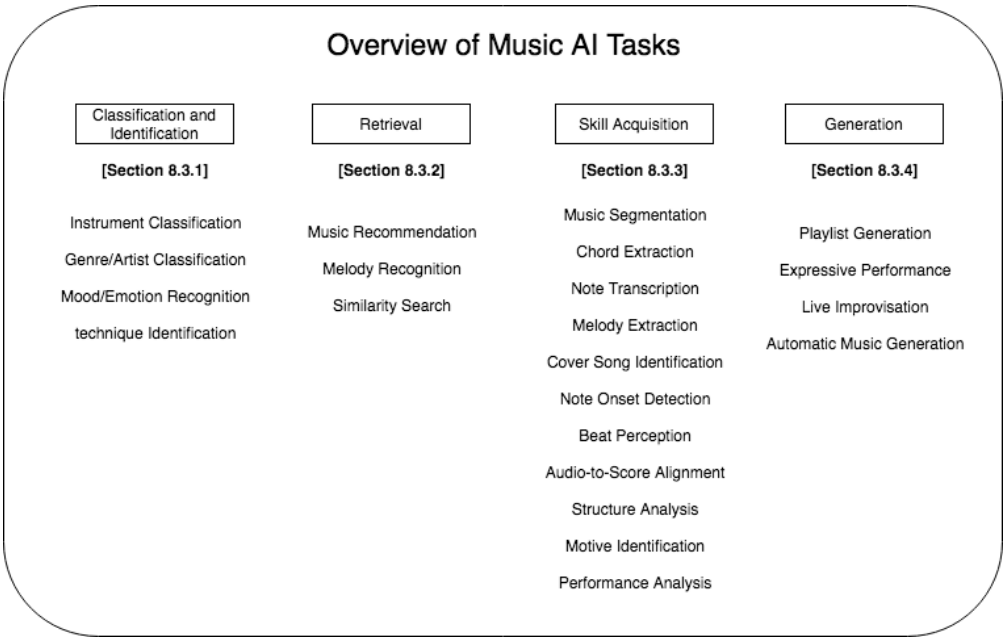

Figure 8.2: Visual high-level illustration of music AI tasks.

a known source, and many have speculated that at least some of them were written by Bach's students (indeed, many have disputed the authorship of entire Bach cantatas, for instance [292]). If we had a reliable way to computationally predict the likelihood that a previously unknown piece was actually written by Bach (vs., say, any other of Bach's many contemporaries), it would help greatly not only in shedding light on such musicological mysteries, but also in revealing what it is that makes Bach unique.

Music domains offer a wide array of potential *classification tasks*. Therefore, partly due to their ease of evaluation (as we discuss further in Section 8.6), they have been a mainstay of the music informatics field for several decades. Indeed, surveying the literature from the past 15 years, a varied list of classification tasks emerges.

Early examples for modern approaches include Scheirer and Slaney, who compared various machine learning techniques, including maximum-aposteriori (MAP) estimators, Gaussian Mixture Models, feature space partitioning and nearest-neighbor search, in order to discriminate speech from music based on acoustic features [346]. Another such early example is the work of Marques and Moreno, who tackled the issue of instrument classification using Gaussian mixture models and SVMs [251].

Instrument classification has been a common thread in the music information retrieval literature. In a survey from 2000, Herrera et al [144] point out several machine learning techniques already employed to identify which instrument is playing in solo recordings. Examples of such works include K-nearest neighbors (KNN), employed for example by Martin and Kim [253], Naive Bayes classifiers (see Martin [252]), and support vector machines (SVMs) (see Marques [250]). Eichner

et al. have introduced the usage of Hidden Markov Models for this purpose in a more realistic and diversified setting with multiple instruments of the same kind [104]. In their experiments, they inferred HMMs in which the states are Gaussian probability density functions for each individual instrument and for each individual note, comparatively, in a data-driven manner, and were able to show that for their particular dataset of real-world recordings, this approach outperformed the baselines. Benetos et al. [26] applied Nonnegative matrix factorization and subset selection, resulting in improved classification accuracy compared to results obtained without these modifications. Joder et al. [171] introduced the notion of *temporal integration* to instrument recognition. Simply put, temporal integration involves the combination of features across multiple time frames to construct more context-aware, higher-level features (the notion was first introduced in a music domain by Meng et al. [263]). By combining temporally aggregated features they were able to beat the state of the art for that time. Considering the harder problem of multi-instrument classification, Garcia et al. were able to classify multiple instruments as long as they were in separate recording channels (with some tolerance to leaking) by training statistical models for individual partials per instrument class [118]. In more recent work, Fourer et al. [112] took a hierarchical approach to classifying timbre in ethnomusicological audio recordings. Their method introduces a hierarchical taxonomy from instruments to sound production categories, which bifurcate further (aerophones $\rightarrow$ blowing; cordophones $\rightarrow$ bowing; plucking or striking; etc), and embeds each timbre class in a projection space that captures weights over these descriptors (the training is done via Latent Discriminant Analysis [264]).

The issue of instrument classification ties in organically to another prevalent line of research, that of *genre classification*. Tzanetakis et al. introduced a hierarchy of audio classification to speech vs. music, genres, and subgenres [119]. Using timbral audio features, they were able to reach accuracy of 60% using Gaussian mixture models. Dubnov et al. [94] trained statistical models to describe musical styles in a way that could also be harnessed towards music generation (a topic we expand on in subsection 8.3.4). Their approach employs dictionary-based prediction methods to estimate the likelihood of future sequences based on the current context (in the paper they compare incremental parsing to the more sophisticated predictive suffix trees). In a comparative study from the same year as the Dubnov et al. work, Li et al. compared multiple audio feature sets and classifier variations (based on SVMs, KNN, GMM and discriminant analysis), and across several different datasets [210]. In 2007, Meng et al. studied the application of temporal integration (a method we mentioned in the paragraph above) to genre classification [263], leading to improvements in performance and robustness.

Different researchers have taken different perspectives on the issue of *finding useful representations* for genre classification (an issue we also discuss in Section 8.4). For instance, Panagakis et al. applied nonnegative multilinear PCA to construct better features for genre classification [296], while Salamon et al. used melody extraction for genre classification in polyphonic settings, reaching accuracy of above 90% on a small sample of distinct genres (instrumental Jazz, vocal jazz, opera,

pop, and flamenco) [342]. Anan et al. used a theoretically grounded approach for learning similarity functions for the purpose of genre recognition [9]. In order to train these similarity functions, they converted their MIDI input to binary chroma sequences (a single binary chroma vector is a sequence of length 12 for each tone, in which present tone indices are assigned the value of 1 and the rest are 0). Marques et al. applied optimum path forests (OPF), a graph-partitioning ensemble approach, for genre classification in potentially large and dynamic music datasets [249]. Rump et al. separated harmonic and percussive features in recordings with autoregressive spectral features and SVMs to improve performance over a non-separated baseline [339], while Panagakis et al. used locality-preserving nonnegative tensor factorization as another means of constructing better features for genre classification [298]. In contrast, West and Cox studied the issue of optimizing frame segmentation for genre classification [411] (we discuss the issue of segmentation more in-depth in Section 8.3.3). Arjannikov et al. tackled the issue of genre classification from a different perspective by training an associative classifier [15] (conversely, association analysis in this context can be perceived as KNN in multiple learned similarity spaces). Hillewaere et al. applied string methods for genre classification in multiple dance recordings, transforming the melodic input into a symbolic contour sequence and applying string methods such as sequence alignment and compression-based distance for classification [148]. Somewhat apart from these works, Mayer and Rauber combine ensembles of not only audio but also lyric (i.e. textual) features for genre classification [256]. In more recent work, Herlands et al. tackled the tricky issue of homogenous genre classification (separating works by relatively similar composers such as Haydn and Mozart), reaching accuracy of 80% using specifically tailored melodic and polyphonic features generated from a MIDI representation [143]. Interestingly, Hamel et al. also studied the issue of transfer learning in genre classification, demonstrating how classifiers learned from one dataset can be leveraged to train a genre classifier for a very different dataset [134].

Another common classification task in the music domain is that of *mood and emotion recognition* in music, a task which is interesting both practically for the purpose of content recommendation, and from a musicological perspective. Yang and Lee used decision trees to mine emotional categorizations (using the Tellegen-Watson-Clark mood model [380]) for music, based on lyrics and tags, and then applied support vector machines to predict the correspondence of audio features to these categories [425]. Han et al. applied support vector regression to categorize songs based on Thayer's model of mood [381], placing songs on the Thayer arousal-valence scale [136]. Trohidis et al. also used both the Tellegen-Watson-Clark model and Thayer's model, and reframed the emotion classification problem as that of multilabel prediction, treating emotional tags as labels [393]. Lu et al. applied boosting for multi-modal music emotion recognition [233]. In their work, they combined both MIDI, audio and lyric features to obtain a multi-modal representation, and used SVMs as the weak learners in the boosting process. Mann et al. classified television theme songs on a 6-dimensional emotion space (dramatic-calm, masculine-feminine, playful-serious, happy-sad, light-heavy, relaxing-exciting) using crowd-sourced complementary information for labels, reaching

accuracy of 80-94% depending on the emotional dimension [244]. More recently, Song et al. studied how different auditory features contribute to emotion prediction from tags extracted from last.fm [366]. It is worth noting that obtaining ground truth for a concept as elusive as mood and emotion recognition is tricky, but labels are often obtained through mining social media or through crowd-sourcing, under the assumption that people are the ultimate arbiters of what mood and emotion music evokes. We discuss this matter in greater detail in section 8.6.

The works described above are a representative, but not comprehensive, sample of the type of work on music classification that has taken place in the last 15 years. Various other classification tasks have been studied. To name a few, Su et al. recently applied sparse cepstral and phase codes to identify guitar playing technique in electric guitar recordings [372]; Toiviainen and Eerola used autocorrelation and discriminant functions for a classification based approach to meter extraction [386]; several works including that of Lagrange et al. tackled the issue of singer identification [196], while Abdoli applied a fuzzy logic approach to classify modes in traditional Iranian music recordings [4].

8.3.2 Retrieval Tasks

Consider now that you are in charge of picking music for a specific person. The only guidance you have is that previously, that person listed some of the songs and the artists he likes. Provided with this knowledge, your task is to find additional music that he will enjoy. You can accomplish this goal by finding music that is *similar* to the music he listed, but different. For this purpose, you must also define what makes different pieces of music similar to one another. Alternatively, you may be faced with a recognition task not that far removed from the classification tasks we listed in the previous subsection: given a piece of music, find a subset of other musical pieces from a given corpus which are most likely to have originated from the same artist. These are just a couple of examples for music retrieval tasks, which combine music databases, queries, and lower-level understanding of how music is structured.

In this subsection we consider different types of retrieval tasks in musical context. These tasks usually require a system to provide examples from a large set given a certain query. Selecting examples that best suit the query is the main challenge in this type of task.

The most straightforward context for such retrieval tasks is that of *music recommendation*. Given some context, the system is expected to suggest songs from a set best suited for the listener. This type of task has been a widely studied problem at the intersection of music and AI. Yoshii et al. combined collaborative and content-based probabilistic models to predict latent listener preferences [429, 430]. Their key insights were that collaborative filtering recommendation could be improved, first by combining user ratings with structural information about the music (based on acoustic data), revealing latent preferences models; and secondly, by introducing an incremental training scheme, thus improving scalability. Similarly, Tiemann et al. also combined social and content-based recommenders to obtain more robust hybrid system [385]. Their approach is ensemble-based, with

separate classifiers trained for social data and for music similarity later combined via a learned decision rule.

A different thread in the music recommendation literature explores the aspect of associating tags with songs. Roughly speaking, tags are a broad set of user-defined labels describing properties of the music, ranging from genre description ("indie", "pop", "classic rock" and so forth), to mood description ("uplifting", "sad" etc), to auditory properties ("female vocalist", "guitar solo" etc), and so forth. Along these lines, Eck et al. trained boosting classifiers to automatically associate unobserved tags to songs for the purpose of improving music recommendation [100]. Similarly, Horsburgh et al. learned artificial "pseudo-tags" in latent spaces to augment recommendation in sparsely annotated datasets [156].

From a temporal perspective, Hu and Ogihara tracked listener behavior over time to generate better models of listener song preferences [160]. Specifically, they use time-series analysis to see how different aspects of listener preference (genre, production year, novelty, playing frequency etc) are trending in order to shape the recommendation weighting. In a related paper, Hu et al. also comparatively evaluated how different features contribute to favorite song selection over time [159]. From the somewhat related perspective of balancing novelty with listener familiarity and preferences, Xing et al. enhanced a standard collaborative filtering approach by introducing notions from the multi-armed bandits literature, in order to balance exploration and exploitation in the process of song recommendation, utilizing a Bayesian approach and Gibbs Sampling for arm utility inference [422].

A full discussion of the components and intricacies of music recommender systems is beyond the scope of this paper, but can be found in Schedl et al. [184] and Song et al. [367].

Another example for a common retrieval task is that of *melody recognition*, either from examples or via a query-by-humming system. Betser et al. introduced a sinusoidal-modeling-based fingerprinting system and used it to identify jingles in radio recordings [35]. Skalak et al. applied vantage point trees to speed up search of sung queries against a large music database [362]. A vantage point tree partitions a metric space hierarchically into intersection spheres. By embedding songs in a metric space and using vantage point trees querying can be significantly reduced. Miotto and Orio applied a chroma indexing scheme and statistical modeling to identify music snippets against a database [266]. Similar to the representation discussed in Anan et al. [9], a chroma index is a length 12 vector which assigns weights for each pitch class based on the Fourier transform of a music fragment. A statistical model representing their chroma frequencies over time is then used with an HMM model for song identification. Another paper that considers identification in a time-series context, but from a different perspective, is that of Wang et al., who iteratively segmented live concert recordings to sections and identify each song separately to recover complete set lists [408]. Also in the context of considering structural properties of music over time, Grosche et al. recovered structure fingerprints, which capture longer structural properties of the music compared to standard fingerprints, to improve the retrieval of matching songs from a database given a query

[130]. These similarity fingerprints are constructed via self-similarity matrices [110] on CENS features [275]. Recently, Bellet et al. introduced a theoretically grounded learned discriminative tree edit similarity model to identify songs based on samples using information about the music semantics [23].

The previously mentioned tasks of music recommendation and melody recognition are strongly connected to the key notion of *similarity in music information retrieval*. Given a query, instead of being asked to retrieve the exact same songs, the system may be expected to retrieve songs which are similar to the query. This sort of problem leads to an extensive branch of research on similarity search in music. Platt considered sparse multidimensional scaling of large music similarity graphs to recover latent similarity spaces [311]. Similarly inspired, Slaney et al. studied various metric learning approaches for music similarity learning [363], while McFee and Lanckriet proposed a heterogeneous embedding model for social, acoustic and semantic features to recover latent similarities [259]. McFee et al. also employed collaborative filtering for this purpose [258]. In a later paper, McFee and Lanckriet expanded the scale of their similarity search approach using spatial trees [260]. Similarly to Mcfee et al., Stenzel and Kamps were able to show that employing collaborative filtering can generate more robust content based similarity measures [370]. From an entirely different perspective, Hofmann-Engl proposed a cognitive model of music similarity to tackle the complicated and multi-dimensional issue of how we define two pieces of music to be similar, applying general melotonic (pitch distribution) transformations [154]. Flexer et al. studied the modeling of spectral similarity in order to improve novelty detection in music recommendation [109]. Mueller and Clausen studied transposition invariant self similarity matrices (which we mentioned in the context of Grosche et al. [130]) for music similarity in general [272]. Hoffman et al. studied the application of hierarchical Dirichlet processes to recover latent similarity measures [153]. In that work, each song is represented as a mixture model of multivariate Gaussians, similar to a Gaussian Mixture Models (GMM). However, unlike GMMs, in the Hierarchical Dirichlet Process, the number of mixture components is not predefined but determined as part of the posterior inference process. The hierarchical aspect is derived from the fact that each song is defined by a group of MFCC features. Similarity between songs can be defined according to the similarity between their corresponding distributions over components. In a somewhat conceptually related paper, Schnitzer et al. employed ensembles of multivariate Gaussians and self organizing maps to learn a similarity metric for music based on audio features [350]. Wang et al. used bag of frame representations to compare whole pieces to one another [407]. Other approaches include that of Ahonen et al. who used a compression based metric for music similarity in symbolic polyphonic music [6], and that of Garcia-Diez et al., who learned a harmonic structure graph kernel model for similarity search [118]. In that specific work, binary chroma vectors (dubbed "chromagrams" in this paper) are transformed to tonal centroid vectors to reduce the chromagram space from 2^{12} to 2^6. Subsequently, the similarity between query and dataset inputs is measured via the Normalized Compression Distance (NCD) [53]. For a specific review of audio-based methods for music classifi-

cation (specifically, genre and mood classification, artist identification and instrument recognition) and annotation (or auto-tagging, to be more exact), see [115].

8.3.3 Musical Skill Acquisition Tasks

The tasks we described above tend to rely on the ability to effectively represent music in a meaningful way which reflects its property and structure. Such a representation is often obtained through manually designed features (see[28] for example). However, a large and varied body of work focuses on the ability to automate the construction of such representations. We consider the spectrum of tasks that lead to useful representations of musical features and structure as *musical skill acquisition*. In the music recommendation example we discussed in the previous subsection, we raised the question of what makes two pieces of music similar to one another, and what makes them distinct. Similarity can lie in basic things like tempo and amplitude, and the overall spectral signature of the piece (what frequencies are heard most of the time). It can lie in subtler things, like how the spectrum changes over time. It can also lie in more abstract musicological properties, such as the rhythmic, harmonic and melodic patterns the music exhibits. Capturing such higher level musical properties is the common thread tying the different tasks we consider as musical skill acquisition tasks.

While the separation between classification or retrieval tasks and "musical skill acquisition" is somewhat nuanced, the key distinction is the following. Classification and retrieval tasks reduce music-related problems to a "simple" computational question that can be studied directly with its musical aspect abstracted away, as the problem has been reframed as a pure classification or information retrieval problem. On the other hand, in the case of musical skill acquisition, we are interested in training a system to learn some fundamental nontrivial property that involves music. Such a task can be in service of a classification or retrieval task further down the line (for instance, identifying harmonic structure for similarity search) or rely on a lower level classification or retrieval task (for instance, harmonic progression analysis by first classifying individual pitches in each frame), but learning the musical property is in itself the goal and therefore the nature of these tasks is different.

Ever since the 18th century, Western scholars have studied the different structural and auditory patterns and properties that characterize music, in what eventually became the modern field of musicology [390]. Musicologists study the structure of melody (how sequences of pitches are combined over time), harmony (how multiple pitches are combined simultaneously over time), rhythm and dynamics. Since the 1960s, musicologists have been using computers to aid in their analyses, when studying large corpora of music or previously unfamiliar music [22]. and when focusing on aspects of music that were previously harder to study quantitatively, such as nuances in vibrato or articulation for violin performers [217]. The automation of these tasks is often more closely related to signal processing than to artificial intelligence, but nonetheless it often involves a large component of machine intelligence, such as analyzing the internal structure of music [306],

recovering shared influences among performers [217], or identifying performers by nuances in their performance [196].

A good example for a musical skill task, or music understanding task, is *music segmentation*. Music segmentation strives to understand the structure of music by partitioning it into functionally separate and semantically meaningful segments. This partitioning can happen on multiple levels - a song could be partitions into an intro, verse, chorus, bridge, and outro for instance, and musical segments can be further broken down into independent musical phrases. The notion of recovering the rules of musical temporal structure is as old as musicology itself, and computational approaches to it date back to the work of Jackendoff and Lerdahl, who proposed a generative theory of tonal music in the early 1980s [208]. In the modern computational research literature, early examples include the work of Batlle and Cano, who used Hidden Markov Models to identify boundaries in music sequences [21], and Harford, who used self organizing maps for the same purpose [138]. Similarly to Batlle and Cano, Sheh et al. also applied HMMs to segment chord sequences [357]. Unlike Batlle and Cano, their approach is unsupervised - the most likely segmentation is extracted using the expectation-maximization (EM) method. Parry and Essa studied feature weighting for automatic segmentation, combining both local and global contour patterns to recover boundaries between musical phrases [304]. Liang et al. used Gaussian models to hierarchically segment musical sequences as a preprocessing step for classification [214]. Pearce et al. compared statistical and rule-based models for melodic segmentation, achieving accuracy of nearly 87% with a hybrid approach [307]. This work was interesting because it revealed (at the time) that data driven approaches alone underperformed compared to a method that combined both statistical boundary prediction and rule-based heuristics that incoporated preexisting knowledge of music theory. Considering the harder problem of segmenting non-professional (and therefore messier and harder to process) recordings, Mueller et al. employed heuristic rules to segment raw recordings of folk tunes to individual notes in order to align them with MIDI versions [274]. To achieve this alignment, the audio was segmented in reference to the much neater MIDI input using a distance function that measures the distance between the chroma expected from the MIDI and those observed in the recording, thus accounting for potential deviations in the non-professional performance.

In strongly related work, Praetzlich and Mueller applied dynamic time warping to segment real opera recordings based on aligning them with a symbolic representation [313]. In a different work, the same authors used techniques from the string matching literature to identify segments in recordings on a frame-level similarity basis [314]. From a probabilistic perspective, Marlot studied a similar type of recordings made by amateur folk musicians, and trained a probabilistic model to segment them into phrases [247]. In Marlot's approach, the signal is first partitioned into fragments that are classified into one of the following categories: speech, solo singing, choir singing, and instrumental music. Then, candidate segment boundaries are obtained by observing how the energy of the signal and its content change. Lastly, Maximum aposteriori inference is applied to find the most likely set of boundaries (the training and evaluation were supervised and were done against

a set of 30 hand-annotated folk music recordings). In more recent work, Rodriguez-Lopez et al. combined cue models with probabilistic approaches for melodic segmentation [336]. Interestingly, in a paper from recent years, Lukashevich compared multiple possible metrics for song segmentation accuracy (a work also related to structure analysis, which we discuss in greater detail later in this subsection) [234]. In this work she exposed the fact that performance of different approaches can vary significantly when altering the accuracy metric. The somewhat subjective character of this task is also evident in the work of Pearce et al.

Along the same lines, much work has been invested in the tasks of *chord extraction and harmonic modeling*, the practice of extracting the harmonic properties of a musical sequence, and reducing it to a more abstract representation of typical patterns. This task is of interest both from a general music understanding perspective and for practical applications such as music recommendation and preference modeling. The literature in this subfield has evolved in an interesting manner. Initial modern approaches, such as that of Paiement et al., were based on graphical models. Paiement et al. trained a graphical probabilistic model of chord progressions and showed it was able to capture meaningful harmonic information based on a small sample of recordings [294]. Burgoyne et al. compared multiple approaches of sequence modeling for automatic chord recognition, mainly comparing Dirichlet-based HMMs and conditional random fields (CRFs) over pitch class profiles [48]. In something of a departure from the earlier problem setting, Mauch and Dixon used structural information about the music to better inform chord extraction, and utilized a discrete probabilistic mixture model for chord recognition, reaching average accuracy of 65% [255]. Cho and Bello introduced recurrence plots (essentially a derivative of the previously discussed self-similarity matrices) as a noise reduction method in order to smooth features and facilitate more accurate chord recognition, improving performance over a non-smoothed baseline.

Unlike the probabilistic graphical models approach, Ogihara and Li trained N-gram chord models for the ultimate purpose of composer style classification (basically treating chords as words) [288]. Combining the N-gram and probabilistic perspectives, Yoshii and Goto introduced a vocabulary free, infinity-gram model composite generative model for nonparametric chord progression analysis, which was able to recover complex chord progressions with high probability [428]. Chen et al. expanded the standard HMM approach to chord recognition using duration-explicit HMM models [56]. Among their innovations is the utilization of a transformation matrix for chroma (learned via regression) that yields a richer spectral representation than that of the traditional chroma vector. On top of this learned representation a generalized, duration-aware HMM is used to predict the most likely chord sequence (using the Viterbi algorithm [318]). Papadopoulos and Tzanetakis chose to combine graphical models with a rule-based approach directly by utilizing a Markov logic networks to simultaneous model chord and key structure in musical pieces. More recently, deep neural networks have become increasingly prevalent for the purpose of chord recognition. Boulanger-Lewandowski et al. studied the application of recurrent neural networks (RNN), and specifically Restricted Boltzmann Machines (RBMs), for audio chord recognition [45], and

Humphrey and Bello applied convolutional neural networks (CNN) for the same purpose [162]. In a strongly related paper, Zhou and Lerch trained a Deconvolutional neural networks (DNN) for feature construction, and combined SVM and HMM classifiers on a bottleneck layer of the DNN for final chord classification [435].

The problem of chord extraction and harmonic modeling is closely linked to that of *note transcription and melody extraction*. Note transcription involves the translation of audio information into a sequential symbolic representation. Melody extraction is the related task of identifying a melodic sequence in a larger musical context and isolating it. Abdallah and Plumbley applied non-negative sparse coding [157] on audio power spectra for polyphonic music transcription [3]. Similarly, Ben Yakar et al. applied unsupervised bilevel sparse models for the same purpose [424]. Madsen and Widmer introduced a formal computational model for melody recognition using a sliding window approach [240]. In their work, they compared entropy measures with a compression-based approach to predict melody notes. Polliner and Ellis framed the melody transcription task as a classification problem, identifying notes in each frame based on the audio spectral properties [312]. From a more statistical perspective, Duan and Temperley apply maximum likelihood sampling to reach note-level music transcription in polyphonic music [93]. Alternatively, taking a Bayesian filtering approach, Jo and Yoo employed particle filters to track melodic lines in polyphonic audio recordings [170]. Kapanci and Pfeffer treated the melody extraction problem from an audio-to-score matching perspective, and trained a graphical model to align an audio recording to a score, recovering melodic lines in the process [177]. A different graphical approach to the problem was introduced by Raczynski et al., who trained a dynamic Bayes network (DBN) for multiple pitch transcription [319]. In their study they were able to show this choice significantly improved performance compared to a reference model that assumed uniform and independently distributed notes. Grindlay and Ellis propose a general probabilistic model suitable for transcribing single-channel audio recordings containing multiple polyphonic sources [128]. As in other related problems, in the last few years multiple researchers have applied deep neural network architectures for this task. Boulanger-Lewandowski et al. applied RNNs to recover multiple temporal dependencies in polyphonic music for the purpose of transcription [44]. Connecting the graphical model literature with the deep architectures thread, Nam et al. applied deep belief networks for unsupervised learning of features later used in piano transcription, showing an improvement over hand designed features [281]. In another recent work on piano transcription, Bock and Schedl applied bidirectional Long Short Term Memory RNNs (LSTMs), reporting improved performance compared to their respective baselines [42]. Berg-Kirkpatrick et al. achieved the same goal of piano note transcription in a fully unsupervised manner, using a graphical model that reflects the process by which musical events trigger perceived acoustic signals [29]. In another recent example, Sigtia et al. presented an RNN-based music sequence model [358]. In the transcription process, prior information from the music sequence model is incorporated as a Dirichlet prior, leading to a hybrid architecture that yields improved transcription accuracy.

Chord analysis, melody extraction and music similarity are all strongly connected to *cover song identification* - another field of music analysis where AI has been applied. Cover song identification is the challenging task of identifying an alternative version of a previous musical piece, even though it may differ substantially in timbre, tempo, structure, and even fundamental aspects relating to the harmony and melody of the song. The term "cover" is so wide that it ranges from acoustic renditions of a previous song, to Jimi Hendrix' famous (and radical) reinterpretation of Bob Dylan's "All Along the Watchtower", to Rage Against the Machine essentially rewriting Bob Dylan's "Maggie's Farm". Beyond its value for computational musicology and for enhancing music recommendation, cover song identification is of interest because of its potential for benchmarking other music similarity and retrieval algorithms. Ellis proposed an approach based on cross-correlation of chroma vector sequences, while accounting for various transpositions [105]. As a critical preprocessing step, chroma vectors were beat-aligned via beat tracking, a separate music information retrieval problem that we discuss further in this section. Serra et al. studied the application of Harmonic Pitch Class Profiles (HPCP) [121, 206] and local alignment via the Smith-Waterman algorithm, commonly used for local sequence alignment in computational biology [364], for this purpose [355]. HPCP is an enhancement of chroma vectors which utilizes the typical overtone properties of most instruments and the human voice to obtain a less noisy representation of the pitch class profile of a musical segment. Serra at el. later proposed extracting recurrence measures from the cross recurrence plot, a cross-similarity matrix of beat-aligned HPCP sequences, for more accurate cover song identification. Since complicated pairwise comparisons for the purpose of en masse cover song identification in large scale datasets is prohibitively computationally expensive, Bertin-Mahieux and Ellis proposed a significant speed-up to previous approaches by extracting the magnitude of the two-dimensional Fourier transform of beat-aligned chroma patches (chroma patches are windowed subsequences of chroma vectors) and then computing the pairwise euclidean distance of these representations (PCA was also applied for dimensionality reduction) [32]. Humphrey et al. further improved on this result by introducing various data-driven modifications to the original framework. These modifications included the application of non-linear scaling and normalization on the raw input, learning a sparse representation, or a dictionary (essentially a set of approximate basis functions that can be used to describe spectral patterns efficiently) in order to further reduce the complexity of the input data [164]. More recently, Tralie and Bendiche cast the cover song identification problem as matching similar yet potentially offset, scaled and rotated patterns in high-dimensional spaces, treating MFCC representations as point-cloud embeddings representing songs [392].

Another important aspect of computational music analysis where machine intelligence has been applied is that of *onset detection*. Onset detection refers to the issue of identifying the beginning of notes in audio representations, and it has been widely studied given its fundamental application to music information analysis. You and Dannenberg proposed a semi-supervised scheme for onset detection in massively polyphonic music, in which more straightforward signal processing

techniques such as thresholding, are likely to fail due to the difficulty in disambiguating multiple adjacent notes with overlapping spectral profiles [431]. To avoid the necessity of hand labeling the countless onsets, audio-to-score alignment is used to estimate note onsets automatically. Because score alignment is done via chroma vectors, which only provide crude temporal estimates (on the order of 50 to 250ms), a trained support vector machine classifier is used to refine these results. Later, Benetos et al. showed that using the auditory spectrum representation can significantly improve onset detection [25]. Inspired by both computational and psycho-acoustical studied of the human auditory cortex, the auditory spectrum model consists of two stages, a spectral estimation model (designed to mimic the cochlea in the auditory system), and a spectral analysis model. Extracting the group delay (the derivative of phase over frequency) [155] and spectral flux (the detection of sudden positive energy changes in the signal) [24], the authors were able to reach dramatic improvements in performance compared to more straightforward Fourier-based onset detection [25]. More recently, Schluter and Bock were able to significantly improve on previous results by training a convolutional neural network for the purpose of beat onset detection [347].

The notion of onset detection naturally leads to another core property of music that has been studied computationally - *beat perception*. The beat of a musical piece is its basic unit of time. More concretely, by "beat perception" we refer to the detection of sequences of temporal emphases that induce the perceived rhythm of a musical piece. We have touched on the issue of beat detection explicitly when we discussed cover song identification (when discussing the works of Ellis et al. [105] and Serra et al. [355]), but in truth the issue of beat tracking is present in almost any task that involves the comparative analysis of audio sequences (in symbolic representations the issue of beat tracking is significantly less challenging for obvious reasons). Raphael introduced a generative model that captures the simultaneous nature of rhythm, tempo and observable beat processes and utilized it for automatic beat transcription. Given a sequence of onset times, a sequence of measure positions, and a Gaussian tempo process, a graphical model is used to describe the process with which these sequences are connected. Using maximum aposteriori inference, the sequence of beats is produced [324]. Alonso et al. defined the notion of spectral energy flux (which we mentioned previously in the context of onset detection) to approximate the derivative of the energy per frequency over time, and use it for efficient beat detection [8]. Paulus and Klapuri combine temporal and spectral features in an HMM-based system for drum transcription [305]. Temporal patterns are modeled as a Gaussian Mixture Model, and are combined with a hidden Markov Model that considers the different drum combinations, and the drum sequence is inferred via maximum likelihood. Gillet and Richard also tackled drum transcription specifically, but took a different approach, training a supervised N-gram model for interval sequences [120]. In their method, after extracting initial predictions based on the N-gram model, a pruning stage takes place in an unsupervised fashion, by reducing the approximate Kolmogorov complexity of the drum sequence. Le Coz et al. proposed a different approach altogether to beat extraction, which does not rely on onset detection, but rather on segmentation [201]. In their paper, they segment each note

into quasi-stationary segments reflecting (approximately) the attack, decay, sustain and release of the notes via forward-backward divergence [11], and reconstruct the beat sequence directly from the resulting impulse train via Fourier analysis.

Beat extraction is closely related to *audio-to-score alignment* and score following - the task of matching audio to a score in an online fashion (we have already touched on this subject in the context of melody extraction and onset detection). Dixon proposed an application of the Dynamic Time Warping algorithm for this purpose [87]. Dynamic Time Warping is a well known dynamic programming algorithm for finding patterns in time series data by aligning two time-dependent sequences [31], and its application in the context of aligning scores to audio data is self-evident (it was also used context such as cover song identification, which we have discussed previously). Pardo and Birmingham tackled the score following from a probabilistic perspective [302]. In their paper, they treating the score as a hidden Markov model, with the audio as the observation sequence, reducing the score following to the problem of finding the most likely state at a given point, which can be done via Viterbi-style dynamic programming. In a recent paper, Coca and Zhao employed network analysis tools to recover rhythmic motifs (represented as highly connected graph sub-components) from MIDI representations of popular songs [62].

Melody, harmony and rhythm modeling, and score alignment, all naturally lead to the task of overall *musical structure analysis*. This problem has been studied as well, from multiple directions. Kameoka et al. employed expectation-maximization to recover the harmonic-temporal overall structure of a given piece. Abdallah et al. propose a Bayesian approach to clustering segments based on harmony, rhythm, pitch and timbre. Peeters applies spectral analysis to the signal envelope to recover the beat properties of recorded music [308]. Peeters' approach was to utilize MFCC and pitch class profile features, construct higher order similarity matrices, and infer the structure via maximum likelihood inference. Mueller and Ewert jointly analyze the structure of multiple aligned versions of the same piece to improve both efficiency and accuracy [273]. This type of analysis is done by finding paths in the pairwise similarity matrix of chroma vector sequences and using them to partially synchronize subsequences in both pieces. Bergeron and Conklin designed a framework for encoding and recovering polyphonic patterns in order to analyze the temporal relations in polyphonic music [30]. To achieve this sort of encoding, they proposed a polyphonic pattern language inspired by algebraic representations of music, which can be seen as a formal logic derivation system for harmonic progressions. From a more utilitarian perspective, as an example for structure analysis as a preprocessing step for other purposes, Mauch et al. used patterns recovered from music structure to enhance chord transcription. Harmonic progressions in Western music tend to obey contextual and structural properties (consider, for instance, the cadenza, a typical harmonic progression signifying the end of a musical phrase). Specifically, in their work, Mauch et al. leverage repetitions in sequences to improve chord extraction by segmenting the raw sequence and identifying those repetitions. From a different perspective, Kaiser and Sikora used nonnegative matrix factorization to recover structure in audio signals [176]. The nonnegative matrix factor-

ization is applied on the timbre self-similarity matrix, and regions of acoustically similar frames in the sequence are segmented. Another unsupervised approach for overall structure analysis is described in more recent work by McFee and Ellis, who employed spectral clustering to analyze song structure. They construct a binary version of the self-similarity matrix which is subsequently interpreted as a unweighted, undirected graph, whose vertices correspond to samples. Then, spectral clustering (through Laplacian decomposition) is applied, with the eigenvalues corresponding to a hierarchy of self-similar segments. In a somewhat related recent paper, Masden et al learned a pairwise distance metric between segments to predict temporally-dependent emotional content in music [238].

A research topic that is related to structure analysis, beat perception, melody, and chord extraction is that of *motive identification* - the extraction of key thematic subject matter from a musical piece. To mention a few papers from the past 15 years, Juhasz studied the application of self-organizing maps and dynamic time warping for the purpose of identifying motives in a corpus of 22 folk songs [173]. Dynamic time warping is used to search for repeated subsequences in melodies (in a way conceptually related to how self-similarity matrices work), and then these sequences are fed to a self organizing map, extracting the most prominent abstracted representations of the core motifs and their correspondence relationships. Lartillot framed the motive extraction problem as combinatorially identifying repeated subsequences in a computationally efficient manner [197]. The subsequences is multidimensional, as it comprises both melodic and rhythmic properties. Lartillot later revisited and refined this approach, and tested in on the Johannes Kepler University Patterns Development Database [64], and was able to show it recovers meaningful motivic patterns.

Lastly, it is worth mentioning another example for the application of AI towards musicological problems - *performance analysis*. The rise in corpora of recorded music has both facilitated and necessitated the application of algorithmic approaches to comparatively analyze multiple recordings of the same pieces. Several good examples for such computational method include the work of Madsen and Widmer, who applied string matching techniques to compare pianist styles [239]. In a related work, Sapp used rank similarity matrices for the purpose of grouping different performances by similarity [344]. Molina-Solana et al. introduced a computational experssiveness model in order to improve individual violinist identification [268]. In past work, the author (Liebman) et al. applied an approach inspired by computational bioinformatics to analyze the evolution and interrelations between different performance schools by constructing an evolutionary tree of influence between performances [217]. Other related works include that of Okomura et al., who employed stochastic modeling of performances to produce an "expressive representation" [289]. More recently, van Herwaarden et al. trained multiple Restricted Boltzmann Machines (RBMs) to predict expressive dynamics in piano recordings [402].

8.3.4 Generation Tasks

Thus far we have considered tasks where intelligent software was required to perform tasks with existing pieces of music as input. However, there is also a wide array of work on employing artificial agents for the purpose of creating music. The autonomous aspect of algorithmic composition has been routinely explored in various artistic contexts [284]. However, while considered by some as the "holy grail" in computer music and the application of AI to music, less scientific attention has been placed on AI for musical content generation compared to other music AI problems.[15] This gap owes at least in part to the fact that evaluating the quality of computer generated content is very difficult, for reasons discussed in Section 8.6

In many ways, the task of *playlist generation*, or recommending music in a sequential and context dependent manner, can be perceived as lying at the intersection of recommendation and generation. In the past 15 years, multiple works have studied machine learning approaches to created meaningful song sequences. Maillet et al. [242] treated the playlist prediction problem as a supervised binary classification task, with pairs of songs in sequence as positive examples and random pairs as negative ones. Mcfee and Lanckriet [261] examined playlists as a natural language model induced over songs, and trained a bigram model for transitions. Chen et al. [57] took a similar Markov approach, treating playlists as Markov chains in some latent space, and learned a metric representation for each song without reliance on audio data. Zheleva et al. [434] adapted a Latent Dirichlet Allocation model to capture music taste from listening activities across users and songs. The author (Liebman) et al. [218] borrow from the reinforcement learning literature and learn a model both for song and transition preferences, then employing a monte-carlo search approach to generate song sequences (this work is discussed at length in Chapter 3 of this dissertation). Wang et al. [409] consider the problem of song recommendations as a bandit problem, attempting to efficiently balance exploration and exploitation to identify novel songs in the playlist generation process, and very similar work has been done by Xing et al. [422] towards this purpose as well. Novelty and diversity in themselves have also been a studied objective of playlists. Logan and Salomon [229, 227] considered novelty in song trajectories via a measure which captures how similar songs are from one another in a spectral sense. Lehtiniemi [207] used context-aware cues to better tailor a mobile music streaming service to user needs, and showed that using such cues increases the novelty experienced by users. More recently, Taramigkou et al. [378] used a combination of Latent Dirichlet Allocation with graph search to produce more diversified playlists that are not pigeonholed to overly specific tastes, leading to user fatigue and disinterest.

Another task of a generative nature is that of *expressive performance*. It is naturally closely related to music performance analysis, but rather than perceiving how humans perform music expressively, the emphasis in this task is on imparting computational entities with the ability to generate music that would seem expressive to a human ear. Early modern approaches to this

[15] By "scientific" we primarily mean principled, measurable and reproducible research in appropriate publication venues.

problem include the work of de Mantaras et al., who applied case-based reasoning for the purpose of algorithmic music performance [83], and that of Ramirez and Hazan, who used a combination of k-means clustering and classification trees to generate expressive performances of Jazz standards [320]. Ramirez et al. later proposed a sequential covering evolutionary algorithm to train a model of performance expressiveness based on Jazz recordings [321]. Diakopoulos et al. proposed an approach for classifying and modeling expressiveness in electronic music, which could also be harnessed for generating automatic performances [84].

The challenge of expressive performance has been of particular interest in robotic platforms. Murata et al. studied the creation of a robotic singer which was able to follow real-time accompaniment [277]. In a somewhat related paper, Xia et al. presented a robotic dancer which tracked music in real time and was trained to match the expressiveness of the music with matching dance movement [421]. Another example is the work of Hoffman and Weinberg, who presented Shimon, a robotic marimba player, and borrowed ideas from the world of animation to make Shimon expressive not just musically, but also visually [151].

Shimon was geared towards *live improvisation*, and indeed improvisation is yet another music generation goal for artificial systems. Eck and Schmidhuber used long short-term memory recurrent neural networks to train a generative model of Jazz improvisation [101]. In a different contemporary work, Thom employed a learned probabilistic model for interactive solo improvisation with an artificial agent [384, 383]. Assayag and Dubnov trained Markov models for music sequences, then employ a type of string matching structures called factor oracles to facilitate algorithmic improvisation [16].

Lastly, there has been some attention from an AI perspective on automatic music generation, though the study of this problem has been relatively limited, particularly due to the difficulty of evaluation (see Section 8.6). In a technical report, Quick borrowed ideas from Shenkerian analysis and chord spaces to create an algorithmic composition framework [317]. Kosta et al. proposed an unsupervised multi-stage framework for chord sequence generation based on observed examples [187]. From a very different perspective, Blackwell has applied multi-swarms to create an improvisational musical system [38]. Very recently, Colombo et al. proposed deep RNN architectures for the purpose of melody composition [65]. Most recently, Dieleman et al. compared different deep architectures for generating music in raw audio format at scale [85], and Huang et al. were able to apply deep sequential generative models with self-attention to generate structured compositions that achieve state of the art performance in synthesizing keyboard compositions [161]. For an interesting overview of AI methods particularly in the use of algorithmic composition, see [108].

8.4 Overview of Common Representations

Thus far, we have focused on breaking down the wide range of musical tasks from a purpose-oriented perspective. However, an equally important perspective involves the types of input used for these

tasks. As noted by Dannenberg [76], representation of the music itself can be viewed as a continuum "ranging from the highly symbolic and abstract level denoted by printed music to the non-symbolic and concrete level of an audio signal". Additionally, one may consider all the additional related information, such as lyrics, tags, artist's identity, etc. as part of the representation. As briefly mentioned in Section 8.2, we consider three main types of information categories for music:

- Symbolic representations - logical data structures representing musical events in time, which may vary in level of abstraction. Examples for different levels of abstraction include but are not limited to the extent of encoded detail regarding pitch, registration, timbre, and performance instructions (accents, slurs, etc).
- Audio representations - this sort of representation captures the other end of the continuum mentioned above, capturing the audio signal itself. Despite its seeming simplicity, here too there is a level of nuance, encompassing the fidelity of the recording (levels of compression, amplitude discretization and so forth), or the level of finesse in representations which perform signal processing on the original audio (such as the ubiquitous chroma and MFCC audio representations we have already mentioned in Section 8.3 and discuss in further detail later in this section).
- Meta-musical information - all the complementary information that can still be legitimately considered part of the musical piece (genre classification, composer identity, structural annotations, social media tags, lyrics etc).

Of these three broad categories, only the first two are within the scope of this survey, since we explicitly focus on aspects of music analysis relating to the music itself, rather than applying machine learning directly and/or exclusively on the complementary information such as lyrics, social media context, or general artist profiles. A visual summary of the contents of this section is presented in Figure 8.3.

We now expand on the first two types of input.

8.4.1 Symbolic Representations for Music

One of the earliest and most common approaches to representing music inputs is via symbolic formats. In essence, a symbolic representation of music is the conceptual abstraction, or the blueprint, of that music. Musical scores using Western notation, for instance, serve exactly as such blueprints. In its most basic form, it includes information on pitches, their length, and when they are played. Additional relevant information includes when each note is released, the amplitude of each note, and the attack (simply put, how rapidly the initial rise in amplitude is and how amplitude decays over time). Classical scores also include a wide range of additional data regarding performance, such as performance instructions, sound effects, slurs, accents, and so forth,

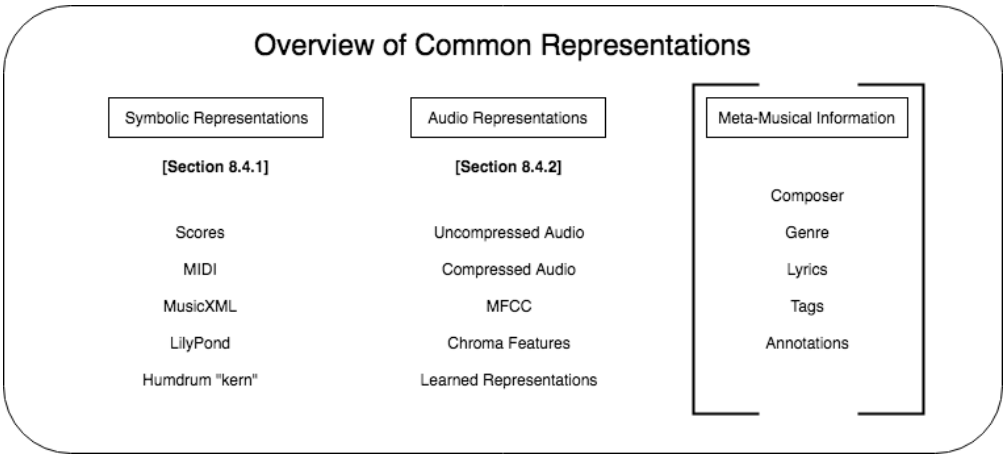

Figure 8.3: Visual high-level overview of music representations used in music AI research. For reasons described in the text, we only consider the first two categories in this chapter.

all of which can often be represented in symbolic formats as well. Additional information such as timbre can be represented, typically by using a preexisting bank of instrument representations.

While this representation isn't as rich as an audio recording, for certain genres, such as classical music or musical theater, which already rely on scores, it is an incredibly informative and useful resource, that eliminates multiple levels of difficulty in dealing with complex auditory data, enabling an artificial agent to know at each moment the core information about pitch, dynamics, rhythm and instrumentation.

One of the most common "blueprint" formats is the MIDI protocol. Since its initial introduction in the early '80s, the MIDI (Musical Instrument Digital Interface) format has served as part of the control protocol and interface between computers and musical instruments [231]. The MIDI format specifies individual notes as "events" represented as tuples of numbers describing varied properties of the note including pitch, velocity (amplitude), vibrato and panning. These note events are sequenced to construct a complete piece, containing up to 16 separate channels of information. These channels typically represent instruments, since each channel can be associated with a separate sound profile, but sometimes the same instrument can be partitioned into multiple channels. Due to its long history and ubiquity, much of the literature utilized this file format as input source (See [335, 427, 426, 129, 330, 240, 147, 394, 9, 246] for a very partial list of examples).

A different approach to symbolic representation aims to digitally represent musical scores, similarly to how traditional music engraving generates scores for mass printing. In the past two decades, several such formats have emerged, including LilyPond [283], Humdrum "kern" [165, 343] and MusicXML [123], among others. While this list is not comprehensive, in terms of symbolic music analysis these formats are largely equivalent and can be converted from one to another with some loss of nuance, but preserving most key features. Examples of research utilizing data in these formats is plentiful and varied (see [343, 360, 74, 13], for, once again, a very partial list of examples).

The advantage of using such music engraving representations, particularly from a musicology perspective, is that they are designed to capture the subtleties of Western notation, including concepts such as notes, rests, key and time signatures, articulation, ornaments, codas and repetitions, etc. This richness of representation is in contrast to the MIDI format, which is conceptually closer to raw audio in terms of abstractions and is designed to describe specific pitched events in time, and is thus less suited to capture the full complexity of more sophisticated music scoring. On the flipside, that is also the relative strength of MIDI compared to these other formats - it is much simpler to parse and process. Furthermore, from a practical standpoint, MIDI largely predates these other formats and is designed as an interface protocol rather than a music engraving language, and is thus far more commonly supported by electronic musical instruments, devices, and software.

8.4.2 Audio Representations and Derived Features

A more intuitive way to represent music is through digital sampling of the raw audio, as is done on audio CDs and using the canonical wave and aiff file formats. In its crudest form, digitizing music audio simply captures amplitude over time in either a single (mono) or dual (stereo) output channel. The quality of recording is dependent on two main aspects:

- The number of bits used to represent amplitudes, which determines quantization noise.
- The sampling frequency, which determines the range of frequencies captured in the digitization process. The standard sampling frequency of 44100Hz ensures that no human audible frequencies are lost.

To these considerations one may also add the possibility of using compression, typically at some cost to frequency resolution [316]. Historically, working directly on raw audio has proven impractical. First, it has traditionally been prohibitively expensive in terms of data storage and processing cost. Second, and more importantly, it has been impractical in terms of the ability of AI software to extract meaningful information from such a low level representation. For reference, this pattern is somewhat analogous to the historical difficulty in using raw pixel data in visual processing.

For this reason, similar to how visual processing resorted to more expressive, condensed representations such as SIFT [230] and HOG [75] features, different features constructed from raw audio have been commonly used. The common are the Mel-frequency cepstral coefficients (MFCC) [228], a derivative of the Fourier transform which captures the short-term power spectrum of a sound. The MFCC is typically constructed using successive temporal windows, thus representing auditory information over time. These coefficients were first used in speech recognition [139], and over the past two decades were shown to be extremely useful in music analysis, serving as a condensed but expressive representation of spectrum over time (see [315, 352, 389, 137, 248] for a few examples).

To reiterate, the symbolic and the auditory aspects of music representation aren't separate categories but rather the two ends of a continuum. A good example for a commonly used representation that lies somewhere in between these two ends is that of chroma features [106]. As we've briefly mentioned in Section 8.3, chroma features record the intensity associated with each of the 12 semitones in an octave, thus, when windowed, capture both melodic and harmonic information over time. Since this representation is typically extracted via analyzing the spectrum of the music, and since it strives to achieve a succinct representation of the notes physically heard throughout a recording, it has something of the auditory representation. At the same time, it also reduces raw audio to a series of pitch information over time, thus also retaining something of the symbolic.

There is an inherent trade-off in choosing a music representation. Audio information is ubiquitous and more immediately useful for large-scale common applications. At the same time, raw recordings are harder to analyze, store and query. Symbolic representations are elegantly concise ways of storing and relaying a great deal of the audio information Western music traditionally cares about (which is in part why reading sheet music is still considered a fundamental skill for musicians), and such representations can be used efficiently for many analysis and retrieval tasks, but they are generally less common, less valuable for mass use and inherently partial in the sense that ultimately crucial auditory information is nonetheless lost. In practice, the choice of representation in the literature is more often than not dictated by availability, ease of use and the nature of the studied task.

In the past few years, as part of the rising popularity and success of deep learning [203], multiple papers have explored the prospects of using deep artificial neural networks to autonomously learn representations - i.e., learn meaningful features - from raw audio. Lee at al. [204] have shown that generic audio classification features learned using convolutional deep belief networks were also useful in 5-way genre classification. Hamel and Eck also explored deep belief nets for both genre classification and automatic tagging, and have shown their learned features to outperform the standard MFCC features [135]. Henaff et al. used sparse coding to learn audio features and showed this approach to be competitive with the state of the art in genre classification on a commonly used dataset [142]. Humphrey et al. surveyed various aspects of deep feature learning, and analyzed how the proposed architectures can be seen as powerful extensions for previously existing approaches [163]. While these new approaches are certainly promising, such architectures have not fully supplanted the previously designed representations discussed in this section, and are not a replacement for existing music interface protocols such as MIDI and music-engraving languages such as LilyPond.

8.5 Overview of Technique

A wide variety of machine learning and artificial intelligence paradigms and techniques have been applied in the context of music domains. From a machine learning and artificial intelligence research

perspective, it is of interest then to examine this range of techniques and the specific musical domains where they were applied. Due to the extensive nature of the related literature and the wide range of musical tasks where the following methods have been used, this list cannot be entirely comprehensive. To the best of our knowledge, however, it is representative of the full array of methods employed. A visual summary of the contents of this section is presented in Figure 8.4.

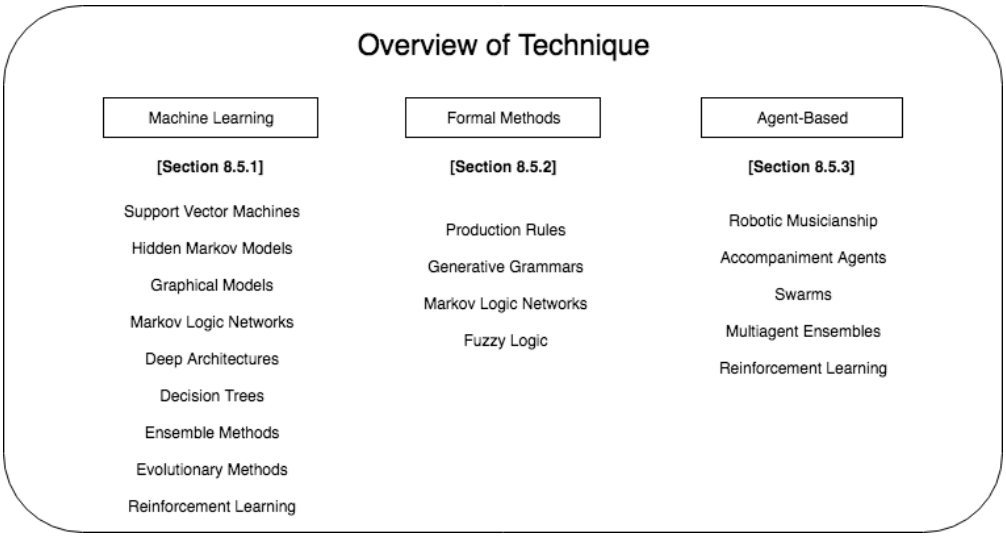

Figure 8.4: Visual high-level overview of algorithmic techniques used in music AI research.

8.5.1 Machine Learning Approaches

Considering the long list of music informatics tasks described in section 8.3, it is clear that many of them can be viewed as machine learning problems. Indeed, a broad spectrum of machine learning techniques have been used to tackle them.

Perhaps one of the oldest documented machine learning approaches for musical tasks is *support vector machines* (SVM) and kernel methods. As mentioned in Section 8.3.1, in an early example of computational approaches to music in general, Marques and Moreno utilized SVM for instrument classification [251]. Xu et al. used a multi-layer SVM approach for genre classification [423]. Their approach was to use the different features representing the spectrum of the audio and hierarchically partition the input first to Pop/Classic or Rock/Jazz, and then within each category (all in all training three SVM models). A similar task was also pursued by Mandel and Ellis, who studied the application of SVM on song-level features for music classification [243]. Meng and Shawe-Taylor studied other types of feature models, namely multivariate Gaussian models and multivariate autoregressive models, for short time window feature representation, with the ultimate goal of improved classification results over 11 genre categories [356]. Han et al. used the strongly related technique of support vector regression for emotion classification in music [136]. Their proposed SMERS system extracts features from the raw audio, maps given audio from its

feature representation to Thayer's two-dimensional emotion model (this emotion representation is based on, and trains a support vector regressor for future prediction. Helen and Vitranen used support vector machines to classify audio components as drums vs. pitched instruments [141]. Ness et al. applied a stacked SVM approach for automatic music tagging, using the key insight that the probabilistic output of one SVM can be used as input for a second layer SVM in order to exploit possible correlations between tags [282]. Maddage et al. trained an SVM classifier to distinguish purely instrumental music sections from ones mixing instruments and vocals, for the purpose of song structure analysis [237]. Gruhne et al. used SVM classifiers for phoneme identification in sung lyrics in order to synchronize audio with text [131]. While useful, the overall popularity of SVM approaches for music informatics seems to have somewhat faded in the past few years, perhaps reflecting its diminishing popularity in the machine learning community in general.

Another well-established and frequently used machine learning approach for musical tasks is that of *probabilistic methods*. Standard examples include Hidden Markov Models (HMM), which are of obvious use given the sequential and partially observable nature of music. In early examples, Battle and Cano used competitive HMMs (or Co-HMMs), a variation on the standard HMM paradigm, for automatic music segmentation [21]. In their study, Co-HMMs were better suited for music partitioning since they required far less apriori domain-knowledge to perform well. Durey et al. used HMMs for the purpose of spotting melodies in music [99], extracting notes from raw audio and treating them as observations in a graphical music language model. Eichner et al. were able to use HMMs for instrument classification. In their paper, they manually collected fragments of solo recordings of four instruments: classical guitar, violin, trumpet and clarinet, and trained separate HMMs for each instrument, leveraging the fact that different instruments induce different note transition mechanics [104]. Sheh and Ellis used HMMs for the more complicated task of chord recognition and segmentation [357], while Noland and Sandler trained an HMM for key estimation [285]. Extending these directions, Burgoyne and Saul applied a hidden Markov model to train Dirichlet distributions for major and minor keys on normalized pitch class profile vectors, for the eventual purpose of tracking chords and keys over time [48].

Chen et al. used a duration-explicit HMM (or DHMM) for better chord recognition [56]. DHMMs work in different time resolutions to estimate the chord sequence by simultaneously estimating chord labels and positions. In their paper, Chen et al. were able to show that explicitly modeling the duration of chords improved recognition accuracy. Considering a different approach, Papadopoulos and Tzanetakis applied Markov Logic Networks (MLNs) for modeling chord and key structure, connecting the probabilistic approach with logic-based reasoning [299]. In practice, their approach is to take Markov networks that encode the transitional chord dynamics of particular scales and combine them with a first-order knowledge base that encodes rules such as "A major chord implies a happy mood". Leveraging the generative capabilities of HMMs, Morris et al. proposed a system that uses a Hidden Markov Model to generate chords to accompany a vocal melody [270]. More recently, Nakamura et al. studied the application of autoregressive Hidden

Semi-Markov Models for score following [279], as well as for recovering piano fingering [280]. In the context of ethnomusicology, Jancovic et al. applied HMMs for automatic transcription of traditional Irish flute music [167].

Graphical models in general have been used in various ways in music domains. Raphael designed a graphical model for recognizing sung melodies [326] and for aligning polyphonic audio with musical scores [325]. Kapanci and Pfeffer explored the related notion of graphical models for signal-to-score music transcription, modeling different aspects of the music such as rhythm and pitch as first-order Gaussian processes [177]. Pickens and Iliopoulos proposed a Markov Random Fields (MRFs) for general music information retrieval tasks [310], citing the power of MRFs in handling non-independent features as their key strength and inherently suitable for music tasks, in which various aspects of features - pitch, timbre, tempo etc) are deeply interdependent. Hoffman et al. used a hierarchical Dirichlet process to estimate music similarity [153]. Hu and Saul proposed an approach a key profiling modeling technique that utilizes a latent Dirchilet allocation (LDA) topic model [158]. The core insight in their paper was that by looking for commonly cooccurring notes in songs, it is possible to learn distributions over pitches for each musical key individually. Yoshii and Goto proposed a novel model for spectral representation called infinite latent harmonic allocation models (iLHA) [428]. Their model represents a Bayesian Nonparametric approach in which each spectral basis is parameterized by means of a Gaussian mixture model (GMM), with both the number of bases and the number of partials being potentially infinite (in practice the least informative elements are zeroed out quickly and a finite approximation remains). In their paper they show this model is useful for multipitch analysis. More recently, Berk-Kirkpatrick et al. proposed a graphical model for unsupervised transcription of piano music, designing a complicated probabilistic activation model for individual keystrokes and inferring the most plausible sequence of key activations to produce a given spectogram [29]. Schmidt and Kim proposed a conditional random field (CRF) approach for tracking the emotional content of musical pieces over time [349]. Later, the same authors would study the application of deep belief networks to learn better music representations, to be used later on in supervised learning tasks [348]. Another very current example of the application of deep generative models for musical task is the work of Manzelli et al., who applied a Long Short Term Memory network (commonly referred to as LSTMs) to learn the melodic structure of different styles of music, and then use the unique symbolic generations from this model as a conditioning input for an audio generation model [245]. The concept of deep belief networks and deep generative models in general is a natural bridge between graphical models and artificial neural network architectures, which indeed constitute the next learning paradigm I will discuss.

Artificial Neural Networks (ANN) are among the oldest paradigms of machine learning. As such, they are also among the oldest to have been used by computational researchers studying musical tasks. To mention a several early modern examples, as early as 1997, Dannenberg et al. used ANNs, among other techniques, for musical style recognition [77]. Kiernan proposed ANNs for score-based style recognition [180], and Rauber et al. applied a self-organizing map (SOM) on

psycho-acoustic features to learn a visualization of music datasets [330]. For some additional details on the prehistory of this approach, it is worth reviewing Griffith and Todd's 1999 short survey on using ANNs for music tasks [126].

In recent years, after an extended lapse in popularity, there has been a resurgence for ANNs via *deep architectures* (commonly dubbed "deep learning"). Naturally, these learning architectures have also been firmly embraced by researchers at the intersection of AI and music. Boulanger-Lewandowski et al. studied audio chord recognition using Recurrent Neural Networks (RNNs) [45]. Herwaarden et al. applied Restricted Boltzmann Machines (RBMs) for predicting expressive dynamics in piano performances [402]. Bock and Schedl applied RNNs for automatic piano transcription [41] and for joint beat and downbeat tracking [41]. In the latter work, an RNN operating directly on magnitude spectrograms is used to model the metrical structure of the audio signals at multiple levels and provides an output feature for a Dynamic Bayes Network which models the bars, thus making this work another example for the fusion of deep architectures and graphical models. Krebs et al. also utilized RNNs for the purpose of downbeat tracking [188], using a very similar RNN + Dynamic Bayes Network learning framework, but in that work they used beat-synchronous audio features rather than the spectogram information. Humphrey et al. applied Convolutional Neural Networks (CNNs) for automatic chord recognition [162]. Humphrey has also been able to show the utility of deep architecture to learn better music representations [163]. CNNs were also recently used by Choi et al. for automatic tagging [59]. In that paper, they use the raw mel-spectorgram as two-dimensional input, and compare the performance of different network architectures, and study their prediction accuracy over the MagnaTagATune dataset. Vogl et al. applied RNNs for automatic drum transcription, training their model to identify the onsets of percussive instruments based on general properties of their sound [404]. Liu and Randall applied bidirectional Long Short Term Memory networks (LSTMs), a form of RNNs, for predicting missing parts in music [226]. Pre-trained neural networks have also been shown useful for music recommendation and auto-tagging, for instance by Liang et al. [213] and Van den Oord et al. [401]. Recently, Durand and Essid proposed a conditional random fields approach for downbeat detection, with features learned via deep architectures, in yet another example for combining graphical models with deep learning models [98]. Another deep generative approach that has been rising in prominence in recent years is that of Generative Adversarial Networks, or GANs, and indeed those too have been used in music AI tasks. As a recent example, Dong et al. proposed MuseGan, a symbolic-domain multi-track music synthesis framework trained on the Lakh dataset [89].

Though somewhat beyond the scope of this paper, one of the most commonplace approaches for decomposing spectral data to individual components is that of *matrix factorization methods*, which can be viewed as an unsupervised learning technique, and were mentioned when discussing music AI tasks, for instance the works of Panagakis et al., who presented a sparse multi-label linear embedding approach based on nonnegative tensor factorization and demonstrate its application to automatic tagging [297], or Kaiser et al., who used these factorization techniques to recover

musical structure [176]. To name a few more examples, Masuda et al. applied semi-supervised nonnegative matrix factorization for query phrase identification in polyphonic music [254], while Sakaue et al. proposed a Bayesian nonnegative factorization approach for multipitch analysis citesakaue2012bayesian. Liang et al. proposed a Beta process nonnegative factorization and show its potential usefulness in several tasks including blind source separation [211], and subsequently Poisson matrix factorization for codebook-based music tagging [212].

Another large family of machine learning models that have seen frequent use in musical domains are *decision trees*. To mention a few examples, Basili et al. applied decision trees for genre classification [20]. Lavner and Ruinskiy proposed a decision-tree based approach for fast segmentation of audio to music vs. speech [199]. Herrera-Boyer and Peeters utilized a decision tree approach for instrument recognition [145]. West and Cox proposed a tree-based approach for learning optimal segmentations for genre classification [412].

As in other domains, the benefits of applying *ensembles of classifiers* has not escaped the music informatics community. To mention a few examples, Tiemann et al. proposed an esnemble learning approach for music recommendation, generating many weak recommendations and combining them via learned decision templates [385]. Dupont and Ravet proposed a novel approach for instrument family classification using ensembles of t-SNE embeddings [97]. Two particularly common ensemble approaches - boosting and random forests - have both been applied in music-related domains. Casagrande et al. used AdaBoost for frame-level audio feature extraction [52]. Turnbull et al. applied boosting for automatic boundary detection [396]. Parker applied AdaBoost to improve a query-by-humming system [303]. Foucard et al. applied boosting for multiscale temporal fusion, later utilized for audio classification [111]. In that paper, data from different timescales is merged through decision trees (serving as another example for the usage of this type of model in music tasks), which are then used as weak learners in an AdaBoost framework. The performance of their proposed system was tested on both instrument classification and song tag prediction, showing that their model was able to improve on prediction using features from only one timescale. Anglade et al. applied random forests to learn harmony rules, which were subsequently applied to improve genre classification [12].

Lastly, it's worth mentioning that though it has not been applied as extensively as other techniques, evolutionary computation has also been used for various music tasks. For instance, Tokui and Iba proposed a system for interactive composition via evolutionary optimization (with human feedback serving as a fitness function) [387]. Biles adapted genetic algorithms for music improvization [37], and as in Section 8.3.4, Ramirez and Hazan employed genetic computation for expressive music performance [321].

While machine learning approaches may indeed be prevalent and ubiquitous in music (as in artificial intelligence research in general), other techniques have been applied as well. In the next subsection I will present two families of such methods: formal (or logic-based) approaches, and agent-based architectures.

8.5.2 Formal Methods

While the learning-based approaches listed above are primarily data driven, many approaches have been employed for music tasks that are inherently rule-based and rely on formal reasoning. We consider this set of techniques as formal methods.

Historically, one of the earliest approaches to the computational understanding of music involved linguistic analysis of music structure. Lehrdal and Jackendoff's seminal work on the generative theory of tonal music [208] is one of the earliest examples for such an approach. Since then, many musicians and researchers have attempted to both analyze and generate music using the derivational structure of *generative grammars* for music and other linguistic constructs [337, 81, 257]. In a somewhat related work, Quick introduced the notion of chord spaces and applied concepts from Schenkerian analysis to define "production rules" for music generation [317].

As previously mentioned, Papadopoulos and Tzanetakis applied *Markov Logic Networks* for modeling chord and key structure [299]. Bergeron and Conklin proposed a structured pattern representation for polyphonic music that defined construction rules for hierarchical patterns, and utilize pattern matching techniques to extract such descriptions from symbolic data [30]. In another relevant example, Abdoli applied fuzzy logic to classify traditional Iranian music [4].

Lastly, though it has declined in fashion over the past 15 years, it is worth mentioning a sizable body of work on music generation through constraint satisfaction techniques. This approach is typified by formulating music rules as constraints and using constraint solving techniques for music generation. For further details and examples, see Pachet and Roy's survey on harmonization with constraints [293].

8.5.3 Agent-Based Techniques

The definition of what exactly makes an "agent" is complicated and open for discussion, and it is outside the scope of this chapter [113]. For our purposes, we define an agent as an artificial system (either physical or, more commonly, implemented in software) that operates in an environment with which it interacts, and makes autonomous decisions.

The vast majority of music-oriented robotics falls under this category. Robotic agents are autonomous systems which need to sense their environments, make decisions, and perform complex continuous control in order to achieve their goals. They may either need to play music alone, as in the work of Solis et al. on a robotic saxophone player [365], or with humans, as in the work of Hoffman et al. on a robotic marimba player [152] and that of Peterson et al. on a robotic flute player [309], but their tasks still involve complex sensing and continuous control. Of course, not only physical robots serve as agents - autonomous accompaniment frameworks such as those proposed by Thom [384] and Raphael [327] which I mentioned previously may certainly be considered autonomous agents. For a fairly recent survey of the state of the art in robotic musicianship, see [46].

Another family of approaches which we define as agent based are multiagent systems, where multiple autonomous, reactive components cooperate in order to perform a musical task. These approaches have been primarily utilized for music generation tasks. Examples include the swarm approach of Blackwell, previously mentioned in the context of music tasks. Blackwell modeled music through particle swarms which generate music through forces of attraction and repulsion [38]. A somewhat similar approach can be seen in the more recent work of Albin et al., who utilized local properties in planar multi-robot configurations for decentralized real time algorithmic music generation [7].

Lastly, it is worth noting that some approaches have directly applied reinforcement learning, which is an agent-based learning paradigm, for various musical tasks. Beyond the work on reinforcement-learning based approaches that was presented in this thesis (particularly in Chapter 3), in recent years others have also considered sequential decision-making as a paradigm for music-related tasks. Cont et al. apply a reinforcement learning model for anticipatory musical style imitation [67]. Wang et al. considered music recommendation as a multi-armed bandit problem, a concept closely related to the RL literature, with the explicit purpose of efficiently balancing exploration and exploitation when suggesting songs to listeners [410]. And quite recently, Dorfer et al. framed score-following as a reinforcement learning task, a sensible approach given that changes in an agent estimation of its position in the score affect its expectation over future score position [91]. In that paper the authors also had the interesting insight that once the agent is trained, it does not need a reward function in order to generate predictions, an observation that would pave the road for other applications of reinforcement learning in similar situations.

To summarize, in this section I have reviewed the wide and varied range of artificial intelligence disciplines utilized in the context of music-related tasks. It is indeed apparent that nearly all major developments in artificial intelligence research have found their way to music applications and domains. In the next section I will address one of the primary challenges of music AI research - how do we evaluate algorithmic performance in music-related tasks?

8.6 Evaluation Methods for Music AI

Having delved into the vicissitudes of the music and AI literature, one should also consider the various evaluation metrics used in assessing success and failure in tackling the varied research questions previously mentioned. In this section we discuss the various approaches observed in the literature for evaluating performance on various musical tasks. Evaluation is often a challenge when it comes to the application of AI for music. Many musical tasks are inherently fuzzy and subjective, and on the face of it, any tasks that are aimed towards humans, be they music recommendation or affective performance, ultimately rely on human feedback as the most reliable (and perhaps the only) measure for success. An additional source of complication stems from the inherently sequential nature of music. In the case of image scene understanding, for instance, a person is able

to perceive, recognize and annotate relatively quickly. Unlike visual data, music is experienced and processed sequentially in time, and often without being afforded the luxury of skipping information or "speed auditing". For these reasons, data from human participants is expensive to obtain, and various other methods have been employed in addition to it, depending on the task. We now briefly discuss such methods in this section. A visual illustration of the breakdown of evaluation method can be seen in Figure 8.5.

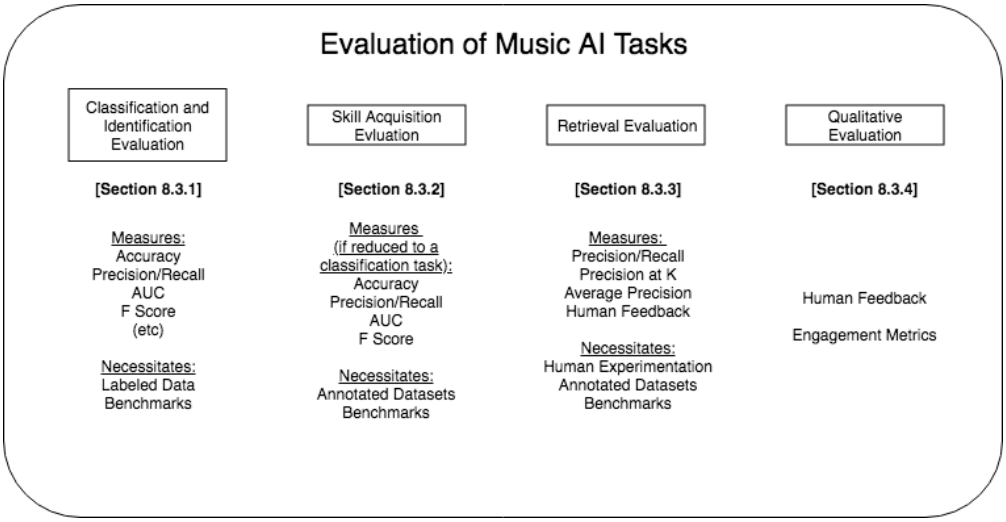

Figure 8.5: Visual high-level overview of evaluation methods used in music AI research.

8.6.1 Evaluation of Classification Tasks

One of the primary reasons why classification tasks have been popular in music informatics is its relative ease of evaluation. Given that a labeled dataset exists, evaluation can rely on the traditional evaluation metrics used in supervised learning, such as overall accuracy, AUC, F-scores etc [278]. Some challenge may still lie in obtaining labeled examples. For certain tasks, such as classification by genre or composer, labels can easily be assigned automatically. For other tasks, such as guitar playing technique classification, getting label information is more difficult. In such cases, collecting hand-annotated data is a common solution [331, 372]. Alternatively speculative labels may be inferred in some cases [115]. Overall, for multiple tasks ranging from sentiment analysis and tagging in music to structure inference, preexisting hand-annotated datasets, such as the Mazurka project for performance analysis [69] or the various existing MIREX datasets [92] serve as necessary benchmarks.

8.6.2 Evaluation of Skill Acquisition Tasks

Skill acquisition (or music understanding) tasks, per our definition from Section 8.3, are generally more difficult than traditional classification, and as such tend to be more difficult to evaluate.

For tasks such as music segmentation, structural analysis and motif identification, for instance, no trivial way to obtain ground truth information exists, and therefore most commonly researchers have relied on hand-annotated datasets for evaluation (as previously discussed in the context of classification tasks).

In certain contexts, in which the underlying skill is learned to facilitate a more complicated task, such as better genre classification, evaluation can be done directly on the final task. This observation holds for many of the aforementioned MIREX tasks, such as key detection and audio downbeat estimation, see the MIREX website for a most current list of tasks and benchmarks.[16]

In certain contexts, such as informative music performance, direct human evaluation has been applied, commonly in a comparative format (of the two performances, which one was more expressive?) [205].

8.6.3 Evaluation of Retrieval Tasks

Like skill acquisition, retrieval tasks are nontrivial for evaluation. For example, they often rely on some notion of similarity among musical pieces, which is often a subjective and relative concept. Even when ground truth exists (for instance, in the form of playlists designed by humans [261]), deducing similarity or commonalities in taste is not immediate.

For music recommendation systems, for instance, the best and most reliable evaluation method is through human experimentation (as was done in Chapter 3, for instance), which is a difficult and time consuming process. Some researchers have gone around this by leveraging preexisting datasets as a surrogate for human preference [261]. Various different methods have been suggested to use limited existing data to impute speculative information regarding success or failure in the underlying task. For instance, in the context of playlist recommendation, it has been suggested that if a given person likes both artists A and B, then having songs by these two artists in the same playlist is considered a success [57, 413]. In other tasks, such as mood analyis, particularly for retrieval purposes, given that certain songs by an artist have been labeled as "moody", assigning this label to other songs by that artist could be considered a success. These methods can be noisy and have obvious limitations (for instance, song style can vary considerably even for songs by the same artist). However, in a recent paper, Craw et al. have shown that preexisting datasets in combination with information extracted from social media can serve as a reasonable approximation for evaluating the effectiveness of different music recommenders, validating their approach via a human study [73].

8.6.4 Qualitative Evaluation

Some music tasks, such as music generation, are very difficult to evaluate even with human feedback. For instance, the fact that 20 out of 100 human subjects liked or didn't like a song isn't in itself

[16] https://www.music-ir.org/mirex/wiki/MIREX_HOME

sufficient evidence for the quality of that song. For this reason, some researchers in the past have relied on qualitative evaluation by experts as a benchmark for performance. While such evaluation is foreign to the world of machine learning and artificial intelligence, it is in line with how culture in general is often evaluated. Another common approach aims for verisimilitude. In the case of style imitation, this approach has some legitimacy, though to the best of my knowledge very few if any recent algorithmic composition algorithms have been put to the test rigorously (i.e. having a large group of people try and differentiate between algorithmic compositions in the style of a given composer and pieces by that composer himself). If I were to speculate, I'd cautiously suggest that in most cases, even in light of recent, truly impressive advances in the field of generative music models (such as the work of Huang et al. [161]), the differentiation between an actual composition by a renowned composer and an algorithmic one is either trivially easy (for experts in particular) or meaningless (for laymen, who would not be able to tell much less professional-sounding algorithmic approximation from actual human compositions). To conclude, despite much progress both in research and in analysis, the question of how to evaluate algorithmic composition in general remains an open problem.

8.7 Summary & Discussion: Open Problems

In this chapter I have reviewed an extremely large body of work involving both AI research and music-related tasks. I have proposed an overall taxonomy of music AI problems, breaking them down by the core nature of the task, by the input type, and by the algorithmic technique employed. I have then proceeded to map out the current state of the art, focusing on research from the past 20 years, relating a wide array of concrete exemplars to the proposed taxonomy.

This panoramic overview of music AI research reveals a dizzyingly complex picture, spanning disciplines and paradigms. On the one hand it feels as though almost any conceivable task has been attempted and any plausible technique has been employed. For some tasks, like key identification [285] or beat detection [98], the current levels of performance are high enough to allow for other tasks to rely on them as lower-level skills (for instance, key identification or beat and note extraction in the service of algorithmic accompaniment [322], or score following [291]). On the other hand, while the research community has been able to make significant strides on many music-related tasks spanning the gamut from extracting chords and notes to structure analysis to playlist recommendation to music synthesis, the more elusive goal of "music understanding" - as I proposed in Section 8 - is still largely unsolved. While we have been able to impart AI with the ability to identify many different building blocks necessary for music understanding, such as recognizing notes, chords, beats, motifs, sentiment (to some extent) and how these relate to more abstract things like listener preferences. But we have yet to teach AI to make sense of all these disparate sources of information; to ground their cultural and semiotic significance; understand the core characteristics of an individual's taste in music (and how it related to one's background, sense of identity etc); to know what a given

chord means to a listener in a given setting; to understand what makes a piece by Telemann banal to modern ears and a piece by Bach a work of timeless genius; or to understand what people listen for when they listen to rock music vs. when they listen to a piano sonata by Beethoven.

To isolate the gaps more concretely, there has been an over-emphasis in the literature on isolating small, encapsulated tasks and focusing on them, without enough consideration of how different tasks connect to some end-goal vision of artificial intelligence for music. Despite their existence (as surveyed in this chapter, particularly under the category of agent-based techniques), there is a relative dearth of music AI *systems*, entities that perform multiple music-related tasks over time, and connect music sensing, understanding and interaction. Two concrete aspects of the research affected by this overall gap is that there has not been much research on music AI systems over time, or on how such systems can practically interact with other agents and with humans and explicitly reason about their perceptions and intentions.

These gaps, as identified in this chapter, highlight the contributions of this thesis. Contribution 2 through 6 all revolve around these two issues: explicitly addressing the temporal, sequential elements that are inherent to music AI tasks through sequential decision-making paradigms, and explicitly reasoning about humans as well as other agents in the context of music-related tasks.

If we envision a future where intelligent artificial agents interact with humans, we would like to make this interaction as natural as possible. We would therefore like to give AI the ability to understand and communicate within cultural settings, by correctly modeling and interpreting human perception and responses. Such progress would have many real world practical benefits, from recommender systems and business intelligence to negotiations and personalized human-computer interaction.

Beyond its practical usefulness, having AI tackle complex cultural domains, which require advanced cognitive skills, would signify a meaningful breakthrough for AI research in general. The research presented in this dissertation towards sequential decision-making and learning social agents in the music domain makes significant progress towards this goal.

9. Conclusion and Future Work

As of December 2018, Spotify, which only a few years would have been considered a small niche company, is valued by investors at over 22 billion US Dollars[17] and has over 80 million paying subscribers.[18] Robotic musicianship has evolved from isolated novelty to a growing field of research with audience-inspiring outcomes [151]. Between 2002 and 2017 the number of members in the International Society for Music Information Retrieval has grown from about 500 to well over 2000.[19] As the field of artificial intelligence for music-related tasks continues to grow, so grows the need for better, more robust learning *systems* that are able to more deeply understand people's responses, preferences, and intentions to music. Such learning systems are not only valuable in the specific contexts in which understanding music, and human psychology with respect to music, is valuable, but also has broader implications regarding learning systems in general. From reasoning about implicit mental states in human counterparts and how they are affected by surrounding stimuli, to temporal adaptation to changing preferences and circumstances, to online context-aware recommendation, music serves as an excellent testbed for a wider, broader array of more general questions involving continual learning, system adaptation, and social agents.

This thesis aims to directly target two key aspects of this type of learning systems: online adaptation through sequential decision-making, and modeling people's preferences and responses with respect to music.

9.1 Thesis Contributions

To recap, the contributions of this thesis are as follows:

- **Contribution 1: Delineating a taxonomy of musical intelligence.** As the body of work which involves both AI and music grows, there is a growing need to put this work in shared context and delineate a clear vision of the current state of the art - what types of tasks exist, on which input, using which techniques, and how they all fit together within a unified framework. This step is crucial if we wish to progress towards complicated integrated systems which involve AI in music-related domains. Indeed, in Chapter 8 of this thesis I provide an extensive overview of the literature and propose such a taxonomy, breaking down previous work by task, input type and technique. This contribution serves both to contextualize the other contributions of this thesis and the gaps they address, but it is also intended to be useful as a standalone, providing a good primer for the current state of affairs in this field for

[17] https://seekingalpha.com/article/4229513-spotify-valuation-problem
[18] https://www.statista.com/statistics/244995/number-of-paying-spotify-subscribers/
[19] https://www.ismir.net/stats.php

AI researchers interested in music as a domain, and music informatics researchers looking for a good source to review the type of AI techniques they could use.

- **Contribution 2: Studying the role of sequential decision making in music recommendation.** As I establish in Chapter 3 of this thesis, one domain which benefits from reframing its core problem as one involving intelligent agents, is that of music playlist recommendation. Playlist recommendation, the task of recommending songs to an individual listener in sequence in real time, naturally blends together preference elicitation and music analysis, and has been historically modeled as a supervised learning task, without too much emphasis on learning a preference over transition dynamics between songs. In Chapter 3 I discuss how reformulating this problem as a sequential decision-making task, on all it entails (chiefly, learning a transition preference on top of song preference on the fly, and learning how to plan effectively in the complex playlist space) can have a substantial impact on listener experience.

- **Contribution 3: Studying the ability of an artificial agent to model the impact of music on people's behavior.** If we envision a future in which AI systems and people interact seamlessly in social environments, these AI systems would need to be able to predict and reason about people's responses to environmental cues, and musical cues are no exception. In Chapter 5 I study how music impacts people's decision making in two different contexts: emotional classification and quantitative, risky decision-making, and propose a framework for modeling this impact in a way that can be leveraged by an artificial agent. The results presented in this chapter indicate that music does indeed impact people's decision making in nontrivial ways, and that this impact can be modeled effectively in terms of how specific decision-making components are affected. Lastly I show how different properties of the music are correlated with different facets of the decision-making process. Beyond their implications in human-agent interaction, these results may prove to be a valuable contribution to our broader understanding of human cognition.

- **Contribution 4: Designing agents that are able to meaningfully interact with people in musical environments.** Prior to this thesis, not much work has been done to establish how an agent could leverage musical information in the environment to better interact with a person in settings which require explicit reasoning about intentions. Building on Contribution 3, in Chapter 6 I present an agent that is able to carry out meaningful interaction with people, crossing an intersection in simulation, directly modeling people's intentions as part of its environment representation. I show how considering music directly helps the agent perform better and more safely than not doing so. Furthermore, to further bolster this aspect of the thesis contributions, in Chapter 7 I study multiagent collaboration in the context of existing preferences - an extremely natural setting in the case of interactive music agents. The results in Chapter 7 are an encouraging necessary step towards more

organic and general agent-agent and agent-person music-based collaboration.

- **Contribution 5: Introducing general algorithms for tracking changes in user preference.** It is a fundamental challenge in continually learning systems that the underlying structure of the data changes over time. Recommendation platforms are no exception. If anything, they are even more susceptible to these changes, given how collective preferences are subjected to wide shifts in style and fashion, and how individual tastes may warp and evolve over time. Indeed, in Chapter 4 I frame this problem as a particular case of concept drift, and formulate the model retraining problem as a sequential decision making task. I introduce algorithms for model management over time in the face of changing data distributions, with direct applications to music recommendation, as well as many other domains.

- **Contribution 6: Showing the applicability of insights gathered from music domains in other types of content-based interaction**. One of the main tenets of this thesis is that while the design of better music agents and solving problems specific to music domains are both worthy and meaningful pursuits, such contributions have benefits and uses that transcend the musical domain exclusively. Indeed, in multiple chapters in this thesis I show concrete examples in which approaches and insights pertaining to music domains have useful applications in other contexts. Concretely, in Chapter 4 I show how continual model adaptation can be utilized on an AT&T dataset that involves customer complaints. In Chapter 7 I show how a multiagent learning setting in which agents have individual preferences has applications in the classic reinforcement learning problem called the pursuit domain. These examples, along with the generalizability of many of the other results (such as the online recommendation framework propsed in Chapter 3), are all indicative of the utility of ideas and insights cultivated in music domains, outside the domains in which they were initially formulated.

9.2 Future Work

The topics covered in this thesis are extensive, and as such, offer many opportunities for further research and extensions. In this section I review such possibilities by topic.

9.2.1 Playlist Recommendation

- *Combining Online Adaptation with Collaborative Filtering* - While the online adaptation approach proposed in Chapter 3 offers a way to mitigate the cold start problem, the framework presented in that chapter could certainly benefit from large scale historical data. A collaborative filtering approach could be extremely useful for training initial priors that are better than the initial uniform model used by DJ-MC. Furthermore, one could envision ways to

generalize information across users *in online adaptation* given a sufficiently large platform such that multiple sessions are happening at once.

- *Listener Prototype Discovery* - Following up on the idea presented above, it could be beneficial to learn mutliple prior listener profiles in advance, and then to consider listeners as distributions over these profiles (this idea is somewhat akin to the k experts model [17]), which could then be adapted online, but more robustly. This vein of thought also ties in to the emerging literature on contextual MDPs, a framework that could also prove beneficial in this setting [133].

- *Mood and Activity Recognition* - Another hybrid approach for connecting personalization and online adaptation with large scale prior learning is through mood and activity recognition. Explicitly identifying listener mood (for instance, by adapting topic models and connecting them with music sentiment analysis) from the context of currently preferred music, rather than relying on the online adaptation to recover this information implicitly, is likely to improve system performance.

- *Extending to Other Domains* - While DJ-MC was particularly designed for music recommendation, the general architecture is largely agnostic of domain specifics, and is amenable, at least in theory, to any sequential data that can be modeled via DJ-MC's binned representation. Therefore, it would be extremely interesting to apply this architecture in other domains in which sequentially recommending content makes sense (such as news items, video streaming services, or personalized television programming).

9.2.2 Adaptive Model Management

- *More Realistic Action Spaces* - While considering three possible actions for model adaptation as studied in Chapter 4, namely keeping the old model, training a new model, and transfer learning from the old model to a new one, lead to interesting results, any real world system should benefit from a richer action space representation that enables more refined decisions involving *how* a new model should be trained or transfer learning should be employed. For instance, one could envision a case in which transfer learning can take place using any past trained model rather than just the most recent one, or that different model learning choices regarding data and architecture could be made. These are interesting expansions to the existing framework that could lead to significantly improved results, but could also require algorithm adaptation due to the increased meta-learning complexity.

- *Multi-Model Adaptation* - The architecture proposed in Chapter 4 can control any number of models by simply learning a retraining policy separately for each underlying model. However, this solution introduces great inefficiency, especially given the possibility of multiple models

reusing underlying data. The question of how to efficiently leverage commonalities between the models could be the focus of interesting and hopefully useful follow-up work.

- *Improving Robustness with more Advanced Policy Learning* - While variations on value iteration and approximate value iteration techniques have their uses, it is definitely possible that other control techniques (such as actor-critic methods or policy search in general) would yield improved results. In fact, even the Fitted Value Iteration approach presented in Chapter 4 wasn't optimized, and proposing and analyzing methods for automatic optimization of the meta-learning architecture could lead to great improvements in performance.

9.3 Modeling the Impact of Music on Human Behavior and Improving Agent-Person Interaction with Music Stimuli

- *Extending to other Tasks* - In Chapters 5 and 6 we have considered, in total, three different task that involve human reasoning with music stimuli in the background: classifying words by their emotional content, accepting or rejecting bets, and driving through a simulated intersection. Interestingly, while music affected people's behavior in all three scenarios, the effects were markedly different from one setting to the other, implying that though music has an impact on multiple decision-making cognitive systems, the nature of this impact is specific to each decision component. Therefore, it is of interest to continue studying the effect of music on a larger set of tasks involving different types of reasoning and decision-making, and to try to identify patterns and commonalities once a clearer picture emerges.

- *Moving Beyond Drift-Diffusion* - While the Drift Diffusion Model (DDM), used extensively in Chapter 5, and indeed the entire class of Decision Field Theory models [50] are useful for breaking down binary decision-making processes to their putative underlying components, it is by and large limited to binary decisions. If we do wish to extend our understanding of how music impacts underlying mechanisms, different models must be used. Therefore, it would be extremely interesting to perform similar analyses to those done using the DDM with other models. For instance, one may consider value-based decision making models [323]. These models borrow heavily from the reinforcement learning literature and conceptualize human decision processes from a reward maximization perspective. Such models could shed a new light on how music affects people's behavior in the context of nonbinary and sequential decisions.

- *Employing a Larger Song Set* - The analyses done in Chapters 5 and 6 which involve breaking down the interrelations between aspects of the music and their impact on behavior hint at the possibility of complex, factored relationships between the properties of music and its impact on behavior. To establish such a relationship more clearly, this analysis must be

extended using a much larger song set, such that a spectrum could be laid out in the different dimensions of song abstraction (describing properties such as tempo, loudness, timbre, chord prevalence etc.) and mapped onto corresponding behavior changes. Such an investigation could also lay the groundwork for a much more refined model of the impact of music an agent could utilize, similar to what was suggested in Chapter 6.

9.4 Multiagent Cooperation in the Face of Preexisting Preferences

- *Exploring a Wider Variety of Preexisting Preferences, Agent Mixtures and Environments* - The analysis done in Chapter 7 is telling, but there is certainly room for additional, more exhaustive explorations, both with respect to the agent's preferences and the underlying architectures they model. Indeed, we would also like to move beyond the two domains studied in Chapter 7, namely multiagent chord synthesis and the pursuit domain. Extending this work to new domains, both music and non-music related, would give us a better understanding of the commonalities and core dynamics engendered by this model of multiagent learning, and would give rise to better multiagent learning frameworks in the face of preexisting biases and agent preferences.

- *The Role of Preferences In Shared Goal Learning* - The work carried out in Chapter 7 assumed preexisting preferences were a given, and studied how these preferences affected shared goal learning. However, the results presented in this chapter indicate that such preexisting preferences can indeed be beneficial towards learning faster and better with respect to the share task. This encouraging outcome raises the question: if we *could* optimize the agents' preexisting preferences, which preexisting preferences would be optimal, and under which circumstances? This research direction connects preference learning and multiagent collaboration with concepts from the transfer and curriculum learning literature, and has ramifications well beyond multiagent music generation.

- *Extending the Work to Human-Agent Collaboration* - one of the initial catalysts of this thread of research was the realization that without better understanding how artificial agents could learn to collaborate musically given a simplified model of intrinsic preference, it would be difficult to teach agents to reason about human musical preferences, which are far more difficult. It is therefore one of the long term research goals set by this thesis to explore this type of person-agent collaboration directly.

9.5 Concluding Remarks

Over the past decade, both reinforcement learning and music informatics have evolved from niche fields to bustling areas of research, of wide interest to industry, public media, and scholars coming

from varied disciplines. Broadly speaking, this dissertation connects the otherwise by-and-large disparate threads of reinforcement learning and music AI research by identifying facets of music informatics that require a sequential decision-making approach, and studying these problems through the lens of the reinforcement learning paradigm. In this dissertation I establish that not only is this a constructive approach for music informatics, I also provide evidence that insights from music domains are indeed generalizable to other, non-music-related tasks, implying this type of work, though in itself of interest to many, holds potential that goes beyond its music applications. I feel fortunate to have been studying problems relating to both these worlds at such an exciting time, and I hope my thesis work will inspire others to continue studying the vast challenges that remain to be explored at the intersection of music and AI research.

Appendix - Glossary

Computational Terms

term	**meaning**
π	policy
$Q(s, a)$	state-action value function
$R(s, a)$	state-action reward function
$V(s)$	state value function
AVI	Approximate Value Iteration
bootstrap resampling	repeated resampling with replacement.
DDetM	Drift Detection Method
DDM	Drift-Diffusion Model
DQN	Deep-Q Network
FVI	Fitted Value Iteration
GAIL	Generative Adversarial Imitation Learning
HDDM	Hoeffding Drift Detection Method
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
MIR	Music Information Retrieval
PPO	Proximal Policy Optimization
RT	Response Time
UCT	Upper Confidence Bounds on Trees

Musical Terms

term	**meaning**
beat	basic unit of time
chord	concurrent set of notes
interval	a step (either sequential or concurrent between
loudness	amplitude of audible sound
major chord	a chord based on a major third interval
minor chord	a chord based on a minor third interval
note	sustained sound with a specific pitch
pitch	the perceived base frequency of a note
playlist	ordered sequence of songs
tempo	speed or pace of a given music
timbre	perceived sound quality of a given note

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM. 108

[2] S. Abdallah and M. Plumbley. Information dynamics: patterns of expectation and surprise in the perception of music. *Connection Science*, 21(2-3):89–117, 2009. 135

[3] S. A. Abdallah and M. D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *5th International Conference on Music Information Retrieval (ISMIR)*, pages 318–325, 2004. 152

[4] S. Abdoli. Iranian traditional music dastgah classification. In *ISMIR*, pages 275–280, 2011. 146, 168

[5] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005. 12, 134

[6] T. E. Ahonen, K. Lemström, and S. Linkola. Compression-based similarity measures in symbolic, polyphonic music. In *ISMIR*, pages 91–96. Citeseer, 2011. 148

[7] A. Albin, G. Weinberg, and M. Egerstedt. Musical abstractions in distributed multi-robot systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 451–458. IEEE, 2012. 169

[8] M. A. Alonso, G. Richard, and B. David. Tempo and beat estimation of musical signals. In *ISMIR*, 2004. 154

[9] Y. Anan, K. Hatano, H. Bannai, M. Takeda, and K. Satoh. Polyphonic music classification on symbolic data using dissimilarity functions. In *ISMIR*, pages 229–234, 2012. 145, 147, 160

[10] J. Anderson. A provisional history of spectral music. *Contemporary Music Review*, 19(2):7–22, 2000. 135

[11] R. Andre-Obrecht. A new statistical approach for the automatic segmentation of continuous speech signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(1):29–40, 1988. 155

[12] A. Anglade, R. Ramirez, S. Dixon, et al. Genre classification using harmony rules induced from automatic chord transcriptions. In *ISMIR*, pages 669–674, 2009. 167

[13] C. Antila and J. Cumming. The vis framework: Analyzing counterpoint in large datasets. In *ISMIR*, pages 71–76, 2014. 160

[14] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni. Gender, genre, and writing style in formal written texts. *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-*, 23(3):321–346, 2003. 134

[15] T. Arjannikov and J. Z. Zhang. An association-based approach to genre classification in music. In *ISMIR*, pages 95–100, 2014. 145

[16] G. Assayag and S. Dubnov. Using factor oracles for machine improvisation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):604–610, 2004. 158

[17] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002. 34, 177

[18] S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld. Teamwork with limited knowledge of teammates. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, July 2013. 103, 124

[19] L. Barrington, R. Oda, and G. Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *International Symposium on Music Information Retrieval*, 2009. 3, 12

[20] R. Basili, A. Serafini, and A. Stellato. Classification of musical genre: a machine learning approach. In *ISMIR*, 2004. 167

[21] E. Batlle and P. Cano. Automatic segmentation for music classification using competitive hidden markov models. 2000. 150, 164

[22] B. Bel and B. Vecchione. Computational musicology. *Computers and the Humanities*, 27(1):1–5, 1993. 149

[23] A. Bellet, J. F. Bernabeu, A. Habrard, and M. Sebban. Learning discriminative tree edit similarities for linear classification—application to melody recognition. *Neurocomputing*, 214:155–161, 2016. 148

[24] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on speech and audio processing*, 13(5):1035–1047, 2005. 154

[25] E. Benetos, A. Holzapfel, and Y. Stylianou. Pitched instrument onset detection based on auditory spectra. In *ISMIR*, pages 105–110. International Society for Music Information Retrieval, 2009. 154

[26] E. Benetos, M. Kotti, and C. Kotropoulos. Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006. 144

[27] A. Berenzweig, B. Logan, D. P. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004. 19

[28] A. L. Berenzweig and D. P. Ellis. Locating singing voice segments within music signals. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 119–122. IEEE, 2001. 149

[29] T. Berg-Kirkpatrick, J. Andreas, and D. Klein. Unsupervised transcription of piano music. In *Advances in neural information processing systems*, pages 1538–1546, 2014. 152, 165

[30] M. Bergeron and D. Conklin. Structured polyphonic patterns. In *ISMIR*, pages 69–74, 2008. 155, 168

[31] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994. 155

[32] T. Bertin-Mahieux and D. P. Ellis. Large-scale cover song recognition using the 2d fourier transform magnitude. In *ISMIR*, pages 241–246, 2012. 153

[33] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR*, volume 2, page 10, 2011. xvi, 15, 17, 38, 52

[34] W. L. Berz. Working memory in music: A theoretical model. *Music Perception*, pages 353–364, 1995. 16

[35] M. Betser, P. Collen, and J.-B. Rault. Audio identification using sinusoidal modeling and application to jingle detection. In *ISMIR*, pages 139–142, 2007. 147

[36] G. Bickerman, S. Bosley, P. Swire, and R. M. Keller. Learning to create jazz melodies using deep belief nets. 2010. 10

[37] J. A. Biles. Improvizing with genetic algorithms: Genjam. In *Evolutionary Computer Music*, pages 137–169. Springer, 2007. 167

[38] T. Blackwell. Swarm music: improvised music with multi-swarms. *Artificial Intelligence and the Simulation of Behaviour, University of Wales*, 2003. 10, 142, 158, 169

[39] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. 141

[40] A. J. Blood and R. J. Zatorre. Intensely pleasurable responses to music correlate with activity in brain regions implicated in reward and emotion. *Proceedings of the National Academy of Sciences*, 98(20):11818–11823, 2001. 64

[41] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, pages 255–261, 2016. 166

[42] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2012 ieee international conference on*, pages 121–124. IEEE, 2012. 152

[43] G. Born. *Rationalizing culture: IRCAM, Boulez, and the institutionalization of the musical avant-garde*. Univ of California Press, 1995. 135

[44] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012. 152

[45] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer, 2013. 151, 166

[46] M. Bretan and G. Weinberg. A survey of robotic musicianship. *Communications of the ACM*, 59(5):100–109, 2016. 168

[47] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2014. 42

[48] J. A. Burgoyne and L. K. Saul. Learning harmonic relationships in digital audio with dirichlet-based hidden markov models. In *ISMIR*, pages 438–443, 2005. 151, 164

[49] J. J. Burred and A. Lerch. A hierarchical approach to automatic musical genre classification. In *Proc. 6th Int. Conf. on Digital Audio Effects' 03*, 2003. 1

[50] J. R. Busemeyer and J. G. Johnson. Computational models of decision making. *Blackwell handbook of judgment and decision making*, pages 133–154, 2004. 178

[51] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008. 103

[52] N. Casagrande, D. Eck, and B. Kégl. Frame-level audio feature extraction using adaboost. In *ISMIR*, pages 345–350, 2005. 167

[53] M. Cebrián, M. Alfonseca, and A. Ortega. The normalized compression distance is resistant to noise. *IEEE Transactions on Information Theory*, 53(5):1895–1900, 2007. 148

[54] J. Chen, J. Yuan, H. Huang, C. Chen, and H. Li. Music-induced mood modulates the strength of emotional negativity bias: An erp study. *Neuroscience Letters*, 445(2):135–139, 2008. 64

[55] R. Chen and M. Li. Music structural segmentation by combining harmonic and timbral information. In *ISMIR*, pages 477–482, 2011. 135

[56] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia. Chord recognition using duration-explicit hidden markov models. In *ISMIR*, pages 445–450. Citeseer, 2012. 151, 164

[57] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722. ACM, 2012. 157, 171

[58] M. Chiosi and B. Freeman. AT&T's sdn controller implementation based on opendaylight. Open Daylight Summit, 7 2015. 41

[59] K. Choi, G. Fazekas, and M. Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016. 166

[60] C.-S. Chow, J. N. Tsitsiklis, et al. An optimal multigrid algorithm for discrete-time stochastic control. 1989. 46, 47

[61] D. Cliff. Hang the dj: Automatic sequencing and seamless mixing of dance-music tracks. *HP LABORATORIES TECHNICAL REPORT HPL*, (104), 2000. 12

[62] A. E. Coca and L. Zhao. Musical rhythmic pattern extraction using relevance of communities in networks. *Information Sciences*, 329:819–848, 2016. 155

[63] W. G. Cochran. Sampling techniques. 1977. *New York: John Wiley and Sons*, 1977. 47

[64] T. Collins. Discovery of repeated themes and sections. *Retrieved 4th May, http://www.musicir. org/mirex/wiki/2013: Discovery of Repeated Themes & Sections*, 2013. 156

[65] F. Colombo, S. P. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner. Algorithmic composition of melodies with deep recurrent neural networks. *arXiv preprint arXiv:1606.07251*, 2016. 158

[66] D. Conklin and I. H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995. 135

[67] A. Cont, S. Dubnov, and G. Assayag. Anticipatory model of musical style imitation using collaborative and competitive reinforcement learning. In *Workshop on Anticipatory Behavior in Adaptive Learning Systems*, pages 285–306. Springer, 2006. 142, 169

[68] N. Cook. *A guide to musical analysis*. Oxford University Press, 1994. 12, 111

[69] N. Cook. Performance analysis and chopin's mazurkas. *Musicae scientiae*, 11(2):183–207, 2007. 170

[70] D. Cope. *Experiments in musical intelligence*, volume 12. AR Editions Madison, WI, 1996. 1

[71] T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from ambiguously labeled images. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 919–926. IEEE, 2009. 134

[72] M. Crampes, J. Villerd, A. Emery, and S. Ranwez. Automatic playlist composition in a dynamic music landscape. In *Proceedings of the 2007 international workshop on Semantically aware document processing and indexing*, pages 15–20. ACM, 2007. 12

[73] S. Craw, B. Horsburgh, and S. Massie. Music recommenders: user evaluation without real users? AAAI/International Joint Conferences on Artificial Intelligence (IJCAI), 2015. 171

[74] M. S. Cuthbert, C. Ariza, and L. Friedland. Feature extraction and machine learning on symbolic music using the music21 toolkit. In *ISMIR*, pages 387–392, 2011. 160

[75] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 161

[76] R. B. Dannenberg. Music representation issues, techniques, and systems. *Computer Music Journal*, 17(3):20–30, 1993. 159

[77] R. B. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. 1997. 165

[78] K. Dautenhahn. Getting to know each other—artificial social intelligence for autonomous robots. *Robotics and autonomous systems*, 16(2-4):333–356, 1995. 135

[79] J. B. Davies and J. B. Davies. *The psychology of music*. Hutchinson London, 1978. 12, 16

[80] R. Davis, B. Buchanan, and E. Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial intelligence*, 8(1):15–45, 1977. 142

[81] W. B. De Haas, M. Rohrmeier, R. C. Veltkamp, and F. Wiering. Modeling harmonic similarity using a generative grammar of tonal harmony. In *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR)*, 2009. 168

[82] S. K. de l'Etoile. The effectiveness of music therapy in group psychotherapy for adults with mental illness. *The Arts in Psychotherapy*, 29(2):69–78, 2002. 64

[83] R. L. De Mantaras and J. L. Arcos. Ai and music: From composition to expressive performance. *AI magazine*, 23(3):43, 2002. 158

[84] D. Diakopoulos, O. Vallis, J. Hochenbaum, J. W. Murphy, and A. Kapur. 21st century electronica: Mir techniques for classification and performance. In *ISMIR*, pages 465–470, 2009. 158

[85] S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, pages 7999–8009, 2018. 158

[86] L. Dixon, R. Trigg, and M. Griffiths. An empirical investigation of music and gambling behaviour. *International Gambling Studies*, 7(3):315–326, 2007. 72, 80

[87] S. Dixon. An on-line time warping algorithm for tracking musical performances. In *IJCAI*, pages 1727–1728, 2005. 155

[88] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (TOG)*, 31(4):101, 2012. 134

[89] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proc. AAAI Conf. Artificial Intelligence*, 2018. 166

[90] S. Doraisamy, S. Golzari, N. Mohd, M. N. Sulaiman, and N. I. Udzir. A study on feature selection and classification techniques for automatic genre classification of traditional malay music. In *ISMIR*, pages 331–336, 2008. 135

[91] M. Dorfer, F. Henkel, and G. Widmer. Learning to listen, read, and follow: Score following as a reinforcement learning game. *arXiv preprint arXiv:1807.06391*, 2018. 169

[92] J. S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008. 170

[93] Z. Duan and D. Temperley. Note-level music transcription by maximum likelihood sampling. In *ISMIR*, pages 181–186. Citeseer, 2014. 152

[94] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine-learning methods for musical style modeling. *Computer*, 36(10):73–80, 2003. 144

[95] S. Dubnov, S. McAdams, and R. Reynolds. Structural and affective aspects of music from statistical audio signal analysis. *Journal of the American Society for Information Science and Technology*, 57(11):1526–1536, 2006. 1

[96] M. Duckham and L. Kulik. "simplest" paths: automated route selection for navigation. In *International Conference on Spatial Information Theory*, pages 169–185. Springer, 2003. 134

[97] S. Dupont and T. Ravet. Improved audio classification using a novel non-linear dimensionality reduction ensemble approach. In *ISMIR*, pages 287–292. Citeseer, 2013. 167

[98] S. Durand and S. Essid. Downbeat detection with conditional random fields and deep learned features. In *ISMIR*, pages 386–392, 2016. 166, 172

[99] A. S. Durey and M. A. Clements. Melody spotting using hidden markov models. In *ISMIR*, 2001. 164

[100] D. Eck, T. Bertin-Mahieux, and P. Lamere. Autotagging music using supervised machine learning. In *ISMIR*, pages 367–368, 2007. 147

[101] D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE, 2002. 158

[102] J. Edworthy and H. Waring. The effects of music tempo and loudness level on treadmill exercise. *Ergonomics*, 49(15):1597–1610, 2006. 25

[103] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994. 31

[104] M. Eichner, M. Wolff, and R. Hoffmann. Instrument classification using hidden markov models. *system*, 1(2):3, 2006. 144, 164

[105] D. P. Ellis. Identifying 'cover songs' with beat-synchronous chroma features. *MIREX 2006*, page 32, 2006. 153, 154

[106] D. P. Ellis and G. E. Poliner. Identifyingcover songs' with chroma features and dynamic programming beat tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1429. IEEE, 2007. 162

[107] B. Eric and N. De Freitas. " name that song!" a probabilistic approach to querying on music and text. In *Advances in neural information processing systems*, pages 1529–1536, 2003. 135

[108] J. D. Fernández and F. Vico. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013. 158

[109] A. Flexer, E. Pampalk, and G. Widmer. Novelty detection based on spectral similarity of songs. In *ISMIR*, pages 260–263, 2005. 148

[110] J. Foote. Visualizing music and audio using self-similarity. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80. ACM, 1999. 148

[111] R. Foucard, S. Essid, M. Lagrange, G. Richard, et al. Multi-scale temporal fusion by boosting for music classification. In *ISMIR*, pages 663–668, 2011. 167

[112] D. Fourer, J.-L. Rouas, P. Hanna, and M. Robine. Automatic timbre classification of ethnomusicological audio recordings. In *International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014. 144

[113] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer, 1996. 168

[114] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015. 42, 50

[115] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE transactions on multimedia*, 13(2):303–319, 2011. 149, 170

[116] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004. 41, 50

[117] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014. 41

[118] S. Garcıa-Dıez, M. Saerens, M. Senelle, and F. Fouss. A simple-cycles weighted kernel based on harmony structure for similarity retrieval. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 61–66, 2011. 144, 148

[119] T. George, E. Georg, and C. Perry. Automatic musical genre classification of audio signals. In *Proceedings of the 2nd international symposium on music information retrieval, Indiana*, 2001. 144

[120] O. Gillet and G. Richard. Supervised and unsupervised sequence modelling for drum transcription. In *ISMIR*, pages 219–224, 2007. 154

[121] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006. 153

[122] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), Dec. 2015. 41

[123] M. Good. Musicxml for notation and analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001. 160

[124] P. Good. Permutation, parametric, and bootstrap tests of hypotheses, 2005. 31

[125] G. J. Gordon. Stable function approximation in dynamic programming. In *Proceedings of the twelfth international conference on machine learning*, pages 261–268, 1995. 47, 55

[126] N. Griffith, P. M. Todd, et al. *Musical networks: Parallel distributed perception and performance*. MIT Press, 1999. 166

[127] M. Griffiths and J. Parke. The psychology of music in gambling environments: An observational research note. *Journal of Gambling Issues*, 2005. 72

[128] G. Grindlay and D. P. Ellis. A probabilistic subspace model for multi-instrument polyphonic transcription. In *ISMIR*, pages 21–26, 2010. 152

[129] P. Grosche, M. Müller, and C. S. Sapp. What makes beat tracking difficult? a case study on chopin mazurkas. In *ISMIR*, pages 649–654, 2010. 160

[130] P. Grosche, J. Serra, M. Müller, and J. L. Arcos. Structure-based audio fingerprinting for music retrieval. In *13th International Society for Music Information Retrieval Conference*, pages 55–60. FEUP Edições, 2012. 148

[131] M. Gruhne, K. Schmidt, and C. Dittmar. Detecting phonemes within the singing of polyphonic music. *Proceedings of ICoMCS December*, page 60, 2007. 164

[132] K. Gurney. *An introduction to neural networks*. CRC press, 1997. 141

[133] A. Hallak, D. Di Castro, and S. Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015. 177

[134] P. Hamel, M. E. Davies, K. Yoshii, and M. Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. In *ISMIR*, pages 9–14, 2013. 145

[135] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *ISMIR*, volume 10, pages 339–344. Utrecht, The Netherlands, 2010. 162

[136] B.-j. Han, S. Ho, R. B. Dannenberg, and E. Hwang. Smers: Music emotion recognition using support vector regression. 2009. 145, 163

[137] Y. Han and K. Lee. Hierarchical approach to detect common mistakes of beginner flute players. In *ISMIR*, pages 77–82, 2014. 161

[138] S. Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. 2003. 150

[139] M. R. Hasan, M. Jamil, M. Rahman, et al. Speaker identification using mel frequency cepstral coefficients. *variations*, 1(4), 2004. 161

[140] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998. 141

[141] M. Helen and T. Virtanen. Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In *Signal Processing Conference, 2005 13th European*, pages 1–4. IEEE, 2005. 164

[142] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. In *ISMIR*, volume 11, page 2011, 2011. 162

[143] W. Herlands, R. Der, Y. Greenberg, and S. Levin. A machine learning approach to musically meaningful homogeneous style classification. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. 145

[144] P. Herrera, X. Amatriain, E. Batlle, and X. Serra. Towards instrument segmentation for music content description: a critical review of instrument classification techniques. In *International symposium on music information retrieval ISMIR*, volume 290, 2000. 143

[145] P. Herrera-Boyer, G. Peeters, and S. Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1):3–21, 2003. 167

[146] L. A. Hiller and L. M. Isaacson. Experimental music: composition with an electronic computer. 1959. 135

[147] R. Hillewaere, B. Manderick, and D. Conklin. String quartet classification with monophonic models. In *ISMIR*, pages 537–542, 2010. 160

[148] R. Hillewaere, B. Manderick, and D. Conklin. String methods for folk tune genre classification. In *ISMIR*, volume 2012, page 13th, 2012. 145

[149] J. Ho and S. Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016. 108

[150] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 141

[151] G. Hoffman and G. Weinberg. Interactive improvisation with a robotic marimba player. *Autonomous Robots*, 31(2-3):133–153, 2011. 142, 158, 174

[152] G. Hoffman and G. Weinberg. Interactive improvisation with a robotic marimba player. *Autonomous Robots*, 31(2-3):133–153, 2011. 168

[153] M. D. Hoffman, D. M. Blei, and P. R. Cook. Content-based musical similarity computation using the hierarchical dirichlet process. In *ISMIR*, pages 349–354, 2008. 148, 165

[154] L. Hofmann-Engl. Towards a cognitive model of melodic similarity. In *ISMIR*, 2001. 148

[155] A. Holzapfel and Y. Stylianou. Beat tracking using group delay based onset detection. In *ISMIR-International Conference on Music Information Retrieval*, pages 653–658. ISMIR, 2008. 154

[156] B. Horsburgh, S. Craw, and S. Massie. Learning pseudo-tags to augment sparse tagging in hybrid music recommender systems. *Artificial Intelligence*, 219:25–39, 2015. 147

[157] P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565. IEEE, 2002. 152

[158] D. Hu and L. K. Saul. A probabilistic topic model for unsupervised learning of musical key-profiles. In *ISMIR*, pages 441–446. Citeseer, 2009. 165

[159] Y. Hu, D. Li, and M. Ogihara. Evaluation on feature importance for favorite song detection. In *ISMIR*, pages 323–328, 2013. 147

[160] Y. Hu and M. Ogihara. Nextone player: A music recommendation system based on user behavior. In *ISMIR*, pages 103–108, 2011. 147

[161] C. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck. An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281, 2018. 158, 172

[162] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 357–362. IEEE, 2012. 152, 166

[163] E. J. Humphrey, J. P. Bello, and Y. LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *ISMIR*, pages 403–408. Citeseer, 2012. 162, 166

[164] E. J. Humphrey, O. Nieto, and J. P. Bello. Data driven and discriminative projections for large-scale cover song identification. In *ISMIR*, pages 149–154, 2013. 153

[165] D. Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002. 160

[166] R. S. Jackendoff. Semantic interpretation in generative grammar. 1972. 135, 142

[167] P. Jancovic, M. Köküer, and W. Baptiste. Automatic transcription of ornamented irish traditional flute music using hidden markov models. In *ISMIR*, pages 756–762, 2015. 165

[168] A. Jansson, C. Raffel, and T. Weyde. This is my jam—data dump. 52

[169] J.-W. Jeong, V. A. Diwadkar, C. D. Chugani, P. Sinsoongsud, O. Muzik, M. E. Behen, H. T. Chugani, and D. C. Chugani. Congruence of happy and sad emotion in music and faces modifies cortical audiovisual activation. *NeuroImage*, 54(4):2973–2982, 2011. 62

[170] S. Jo and C. D. Yoo. Melody extraction from polyphonic audio based on particle filter. In *ISMIR*, pages 357–362. Citeseer, 2010. 152

[171] C. Joder, S. Essid, and G. Richard. Temporal integration for audio classification with application to musical instrument classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(1):174–186, 2009. 144

[172] J. John. Pandora and the music genome project. *Scientific Computing*, 23(10):40–41, 2006. 3

[173] Z. Juhász. Motive identification in 22 folksong corpora using dynamic time warping and self organizing maps. In *ISMIR*, pages 171–176, 2009. 156

[174] P. N. Juslin and D. Västfjäll. Emotional responses to music: The need to consider underlying mechanisms. *Behavioral and brain sciences*, 31(5):559–575, 2008. 135

[175] B. Kahnx, R. Ratner, and D. Kahneman. Patterns of hedonic consumption over time. *Marketing Letters*, 8(1):85–96, 1997. 12

[176] F. Kaiser and T. Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *ISMIR*, pages 429–434, 2010. 155, 167

[177] E. Kapanci and A. Pfeffer. Signal-to-score music transcription using graphical models. In *IJCAI*, pages 758–765. Citeseer, 2005. 152, 165

[178] P. Khandelwal, S. Barrett, and P. Stone. Leading the way: An efficient multi-robot guidance system. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 1625–1633. International Foundation for Autonomous Agents and Multiagent Systems, 2015. 35

[179] P. Khandelwal, E. Liebman, S. Niekum, and P. Stone. On the analysis of complex backup strategies in monte carlo tree search. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1319–1328, 2016. 35, 40

[180] F. J. Kiernan. Score-based style recognition using artificial neural networks. In *ISMIR*, 2000. 165

[181] M. G. Kirschenbaum. The remaking of reading: Data mining and the digital humanities. In *The National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation, Baltimore, MD*, 2007. 134

[182] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004. 42

[183] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000. 41

[184] P. Knees and M. Schedl. A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(1):2, 2013. 147

[185] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, 2006. 33, 34

[186] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000. 10

[187] K. Kosta, M. Marchini, and H. Purwins. Unsupervised chord-sequence generation from an audio example. In *ISMIR*, pages 481–486, 2012. 158

[188] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat synchronous features with recurrent neural networks. In *ISMIR*, pages 129–135, 2016. 166

[189] G. Kreitz and F. Niemela. Spotify–large scale, low latency, p2p music-on-demand streaming. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–10. IEEE, 2010. 3

[190] C. L. Krumhansl. Rhythm and pitch in music cognition. *Psychological bulletin*, 126(1):159, 2000. 3

[191] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Oxford University Press, 2001. 135

[192] C. L. Krumhansl. Music: A link between cognition and emotion. *Current Directions in Psychological Science*, 11(2):45–50, 2002. 64

[193] C. Kuhbandner and R. Pekrun. Joint effects of emotion and color on memory. *Emotion*, 13(3):375, 2013. 64

[194] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 127

[195] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001. 141

[196] M. Lagrange, A. Ozerov, and E. Vincent. Robust singer identification in polyphonic music using melody enhancement and uncertainty-based learning. In *13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012. 146, 150

[197] O. Lartillot. Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 229–235. IEEE, 2005. 156

[198] A. Latham. *The Oxford dictionary of musical terms*. Oxford University Press, 2004. 137

[199] Y. Lavner and D. Ruinskiy. A decision-tree-based algorithm for speech/music classification and segmentation. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:2, 2009. 167

[200] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997. 55

[201] M. Le Coz, H. Lachambre, L. Koenig, and R. Andre-Obrecht. A segmentation-based tempo induction method. In *ISMIR*, pages 27–32, 2010. 154

[202] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 141

[203] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015. 162

[204] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009. 162

[205] J. H. Lee. Crowdsourcing music similarity judgments using mechanical turk. In *ISMIR*, pages 183–188, 2010. 171

[206] K. Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *ICMC*, 2006. 153

[207] A. Lehtiniemi. Evaluating supermusic: streaming context-aware mobile music service. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 314–321. ACM, 2008. 37, 157

[208] F. Lerdahl and R. S. Jackendoff. *A generative theory of tonal music*. MIT press, 1985. 150, 168

[209] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2036–2043. IEEE, 2009. 136

[210] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289. ACM, 2003. 144

[211] D. Liang, M. D. Hoffman, and D. P. Ellis. Beta process sparse nonnegative matrix factorization for music. In *ISMIR*, pages 375–380, 2013. 167

[212] D. Liang, J. Paisley, D. Ellis, et al. Codebook-based scalable music tagging with poisson matrix factorization. In *ISMIR*, pages 167–172. Citeseer, 2014. 167

[213] D. Liang, M. Zhan, and D. P. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *ISMIR*, pages 295–301, 2015. 166

[214] W. Liang, S. Zhang, and B. Xu. A hierarchical approach for audio stream segmentation and classification. In *ISMIR*, pages 582–585, 2005. 150

[215] E. Liebman, B. Chor, and P. Stone. Representative selection in non metric datasets. *eprint arXiv:1502.07428*, 2015. 21

[216] E. Liebman, P. Khandelwal, M. Saar-Tsechansky, and P. Stone. Designing better playlists with monte carlo tree search. In *Twenty-Ninth IAAI Conference*, 2017. 13

[217] E. Liebman, E. Ornoy, and B. Chor. A phylogenetic approach to music performance analysis. *Journal of New Music Research*, 41(2):195–222, 2012. 135, 149, 150, 156

[218] E. Liebman, M. Saar-Tsechansky, and P. Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 591–599. International Foundation for Autonomous Agents and Multiagent Systems, 2015. 13, 38, 157

[219] E. Liebman, M. Saar-Tsechansky, and P. Stone. The right music at the right time: Adaptive personalized playlists based on sequence modeling. *MISQ*, to appear. 13

[220] E. Liebman, P. Stone, and C. N. White. How music alters decision making - impact of music stimuli on emotional classification. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 793–799, 2015. 79, 87

[221] E. Liebman, P. Stone, and C. N. White. How music alters decision making-impact of music stimuli on emotional classification. In *Proceedings of the 16th International Society of Music Information Retrieval Conference (ISMIR 2015)*, 2015. 60

[222] E. Liebman, P. Stone, and C. N. White. Impact of music on decision making in quantitative tasks. In *ISMIR*, pages 661–667, 2016. 60

[223] E. Liebman, C. N. White, and P. Stone. On the Impact of Music on Decision Making in Cooperative Tasks. In *19th International Society for Music Information retrieval Conference (ISMIR)*, September 2018. 81

[224] E. Liebman, E. Zavesky, and P. Stone. A stitch in time: Autonomous model management via reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018. 43

[225] I. Liu and B. Ramakrishnan. Bach in 2014: Music composition with recurrent neural network. *CoRR*, abs/1412.3191, 2014. 10

[226] I.-T. Liu and R. Randall. Predicting missing music components with bidirectional long short-term memory neural networks. In *ISMIR*, pages 225–231, 2016. 166

[227] B. Logan. Content-based playlist generation: Exploratory experiments. In *ISMIR*, 2002. 157

[228] B. Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pages 1–11, 2000. 161

[229] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *ICME*, pages 22–25, 2001. 37, 39, 157

[230] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 161

[231] G. Loy. Musicians make a standard: the midi phenomenon. *Computer Music Journal*, 9(4):8–26, 1985. 141, 160

[232] L. Lu, H. You, H. Zhang, et al. A new approach to query by humming in music retrieval. In *ICME*, pages 22–25, 2001. 1

[233] Q. Lu, X. Chen, D. Yang, and J. Wang. Boosting for multi-modal music emotion. In *11th International Society for Music Information and Retrieval Conference*, pages 105–105, 2010. 145

[234] H. Lukashevich and I. Fraunhofer. Towards quantitative measures of evaluating song segmentation. pages 375–380, 2008. 151

[235] P. MacAlpine, S. Barrett, D. Urieli, V. Vu, and P. Stone. Design and optimization of an omnidirectional humanoid walk: A winning approach at the robocup 2011 3d simulation competition. In *AAAI*, 2012. 10

[236] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967. 23, 36

[237] N. C. Maddage, C. Xu, and Y. Wang. A svm c based classification approach to musical audio. 2003. 164

[238] J. Madsen, B. Sand Jensen, and J. Larsen. Modeling temporal structure in music for emotion prediction using pairwise comparisons. 2014. 156

[239] S. T. Madsen and G. Widmer. Exploring pianist performance styles with evolutionary string matching. *International Journal on Artificial Intelligence Tools*, 15(04):495–513, 2006. 156

[240] S. T. Madsen and G. Widmer. Towards a computational model of melody identification in polyphonic music. In *IJCAI*, pages 459–464, 2007. 152, 160

[241] G. Maicas, G. Carneiro, A. P. Bradley, J. C. Nascimento, and I. Reid. Deep reinforcement learning for active breast lesion detection from dce-mri. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 665–673. Springer, 2017. 95

[242] F. Maillet, D. Eck, G. Desjardins, P. Lamere, et al. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*, pages 345–350, 2009. 157

[243] M. I. Mandel and D. P. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, pages 577–582, 2008. 163

[244] M. Mann, T. J. Cox, and F. F. Li. Music mood classification of television theme tunes. In *ISMIR*, pages 735–740, 2011. 146

[245] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis. Conditioning deep generative raw audio models for structured automatic music. *arXiv preprint arXiv:1806.09905*, 2018. 165

[246] A. Mardirossian and E. Chew. Music summarization via key distributions: Analyses of similarity assessment across variations. In *ISMIR*, pages 234–239, 2006. 135, 160

[247] M. Marolt. Probabilistic segmentation and labeling of ethnomusicological field recordings. In *ISMIR*, 2009. 150

[248] M. Marolt. Probabilistic segmentation and labeling of ethnomusicological field recordings. In *ISMIR*, pages 75–80, 2009. 161

[249] C. M. Marques, I. R. Guilherme, R. Y. Nakamura, and J. P. Papa. New trends in musical genre classification using optimum-path forest. In *ISMIR*, pages 699–704, 2011. 145

[250] J. Marques. *An automatic annotation system for audio data containing music*. PhD thesis, Massachusetts Institute of Technology, 1999. 143

[251] J. Marques and P. J. Moreno. A study of musical instrument classification using gaussian mixture models and support vector machines. *Cambridge Research Laboratory Technical Report Series CRL*, 4, 1999. 143, 163

[252] K. D. Martin. *Sound-source recognition: A theory and computational model*. PhD thesis, Massachusetts Institute of Technology, 1999. 143

[253] K. D. Martin and Y. E. Kim. Musical instrument identification: A pattern-recognition approach. 1998. 143

[254] T. Masuda, K. Yoshii, M. Goto, and S. Morishima. Spotting a query phrase from polyphonic music audio signals based on semi-supervised nonnegative matrix factorization. In *ISMIR*, pages 227–232, 2014. 167

[255] M. Mauch, K. Noland, and S. Dixon. Using musical structure to enhance automatic chord transcription. In *ISMIR*, pages 231–236, 2009. 151

[256] R. Mayer and A. Rauber. Musical genre classification by ensembles of audio and lyrics features. In *Proceedings of International Conference on Music Information Retrieval*, pages 675–680, 2011. 145

[257] J. McCormack. Grammar based music composition. *Complex systems*, 96:321–336, 1996. 168

[258] B. McFee, L. Barrington, and G. R. Lanckriet. Learning similarity from collaborative filters. In *ISMIR*, pages 345–350, 2010. 148

[259] B. McFee and G. R. Lanckriet. Heterogeneous embedding for subjective artist similarity. In *ISMIR*, pages 513–518, 2009. 148

[260] B. McFee and G. R. Lanckriet. Large-scale music similarity search with spatial trees. In *ISMIR*, pages 55–60, 2011. 148

[261] B. McFee and G. R. Lanckriet. The natural language of playlists. In *ISMIR*, pages 537–542, 2011. 157, 171

[262] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *ISMIR*, 2004. 1

[263] A. Meng, P. Ahrendt, J. Larsen, and L. K. Hansen. Temporal feature integration for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1654–1664, 2007. 144

[264] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.*, pages 41–48. Ieee, 1999. 144

[265] L. L. Minku and X. Yao. Ddd: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, 24(4):619–633, 2012. 42

[266] R. Miotto and N. Orio. A music identification system based on chroma indexing and statistical modeling. In *ISMIR*, pages 301–306, 2008. 147

[267] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 94, 95

[268] M. Molina-Solana, J. L. Arcos, and E. Gomez. Using expressive trends for identifying violin performers. In *ISMIR*, pages 495–500, 2008. 156

[269] H. Morita, S. Hashimoto, and S. Ohteru. A computer music system that follows a human conductor. *Computer*, 24(7):44–53, 1991. 11

[270] D. Morris, I. Simon, and S. Basu. Exposing parameters of a trained dynamic model for interactive music creation. In *AAAI*, pages 784–791, 2008. 164

[271] M. J. Mulder, E.-J. Wagenmakers, R. Ratcliff, W. Boekel, and B. U. Forstmann. Bias in the brain: a diffusion model analysis of prior probability and potential payoff. *The Journal of Neuroscience*, 32(7):2335–2343, 2012. 61

[272] M. Müller and M. Clausen. Transposition-invariant self-similarity matrices. In *ISMIR*, pages 47–50, 2007. 148

[273] M. Müller and S. Ewert. Joint structure analysis with applications to music annotation and synchronization. In *ISMIR*, pages 389–394, 2008. 155

[274] M. Müller, P. Grosche, and F. Wiering. Robust segmentation and annotation of folk song recordings. In *ISMIR*, pages 735–740, 2009. 150

[275] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *ISMIR*, volume 2005, page 6th, 2005. 148

[276] E. Muñoz, J. M. Cadenas, Y. S. Ong, and G. Acampora. Memetic music composition. *IEEE Transactions on Evolutionary Computation*, 20(1):1–15, 2016. 135

[277] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino. A robot singer with music recognition based on real-time beat tracking. In *ISMIR*, pages 199–204, 2008. 158

[278] K. Murphy. Machine learning: a probabilistic approach. *Massachusetts Institute of Technology*, pages 1–21, 2012. 170

[279] E. Nakamura, P. Cuvillier, A. Cont, N. Ono, and S. Sagayama. Autoregressive hidden semi-markov model of symbolic music performance for score following. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015. 165

[280] E. Nakamura, N. Ono, and S. Sagayama. Merged-output hmm for piano fingering of both hands. In *ISMIR*, pages 531–536, 2014. 165

[281] J. Nam, J. Ngiam, H. Lee, and M. Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *ISMIR*, pages 175–180, 2011. 152

[282] S. R. Ness, A. Theocharis, G. Tzanetakis, and L. G. Martins. Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 705–708. ACM, 2009. 164

[283] H.-W. Nienhuys and J. Nieuwenhuizen. Lilypond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, volume 1, pages 167–171, 2003. 160

[284] G. Nierhaus. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009. 157

[285] K. C. Noland and M. B. Sandler. Key estimation using a hidden markov model. In *ISMIR*, pages 121–126, 2006. 164, 172

[286] T. J. Noseworthy and K. Finlay. A comparison of ambient casino sound and music: Effects on dissociation and on perceptions of elapsed time while playing slot machines. *Journal of Gambling Studies*, 25(3):331–342, 2009. 72

[287] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM, 2005. 12

[288] M. Ogihara and T. Li. N-gram chord profiles for composer style representation. In *ISMIR*, pages 671–676, 2008. 151

[289] K. Okumura, S. Sako, and T. Kitamura. Stochastic modeling of a musical performance with expressive representations from the musical score. In *ISMIR*, pages 531–536. Citeseer, 2011. 156

[290] N. Oliver and L. Kreger-Stickles. Papa: Physiology and purpose-aware automatic playlist generation. In *Proc. 7th Int. Conf. Music Inf. Retrieval*, pages 250–253, 2006. 12

[291] T. Otsuka, K. Nakadai, T. Ogata, and H. G. Okuno. Incremental bayesian audio-to-score alignment with flexible harmonic structure models. In *ISMIR*, pages 525–530, 2011. 172

[292] A. M. Owen. The authorship of bach's cantata no. 15. *Music & Letters*, 41(1):28–32, 1960. 143

[293] F. Pachet and P. Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001. 168

[294] J.-F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR)*, number EPFL-CONF-83178, 2005. 151

[295] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 42

[296] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):576–588, 2010. 144

[297] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Sparse multi label linear embedding non-negative tensor factorization for automatic music tagging. In *Eighteenth European Signal Processing Conference*, pages 492–496, 2010. 166

[298] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Sparse multi-label linear embedding within nonnegative tensor factorization applied to music tagging. In *ISMIR*, pages 393–398, 2010. 145

[299] H. Papadopoulos and G. Tzanetakis. Modeling chord and key structure with markov logic. In *ISMIR*, pages 127–132, 2012. 164, 168

[300] S. Paquette, I. Peretz, and P. Belin. The "musical emotional bursts": a validated set of musical affect bursts to investigate auditory affective processing. *Frontiers in psychology*, 4, 2013. 64

[301] P. M. Pardalos and J. Xue. The maximum clique problem. *Journal of global Optimization*, 4(3):301–328, 1994. 39

[302] B. Pardo and W. Birmingham. Modeling form for on-line following of musical performances. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20, page 1018. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005. 155

[303] C. L. Parker. Applications of binary classification and adaptive boosting to the query-by-humming problem. In *ISMIR*, pages 245–251, 2005. 167

[304] R. Parry and I. Essa. Feature weighting for segmentation. *Proc. ICMIR*, pages 116–119, 2004. 150

[305] J. Paulus and A. Klapuri. Combining temporal and spectral features in hmm-based drum transcription. In *ISMIR*, pages 225–228, 2007. 154

[306] J. Paulus, M. Müller, and A. Klapuri. State of the art report: Audio-based music structure analysis. In *ISMIR*, pages 625–636, 2010. 149

[307] M. Pearce, D. Müllensiefen, and G. A. Wiggins. A comparison of statistical and rule-based models of melodic segmentation. In *ISMIR*, pages 89–94, 2008. 1, 135, 150

[308] G. Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *ISMIR*, pages 35–40, 2007. 155

[309] K. Petersen, J. Solis, and A. Takanishi. Musical-based interaction system for the waseda flutist robot. *Autonomous Robots*, 28(4):471–488, 2010. 168

[310] J. Pickens and C. S. Iliopoulos. Markov random fields and maximum entropy modeling for music information retrieval. In *ISMIR*, pages 207–214, 2005. 165

[311] J. C. Platt. Fast embedding of sparse similarity graphs. In *Advances in neural information processing systems*, pages 571–578, 2004. 148

[312] G. E. Poliner and D. P. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007(1):048317, 2006. 152

[313] T. Prätzlich and M. Müller. Freischütz digital: A case study for reference-based audio segmentation for operas. In *ISMIR*, pages 589–594, 2013. 150

[314] T. Prätzlich and M. Müller. Frame-level audio segmentation for abridged musical works. In *ISMIR*, pages 307–312, 2014. 150

[315] P. Proutskova and M. A. Casey. You call that singing? ensemble classification for multi-cultural collections of music recordings. In *ISMIR*, pages 759–764. Citeseer, 2009. 161

[316] D. Pye. Content-based methods for the management of digital music. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 4, pages 2437–2440. IEEE, 2000. 161

[317] D. Quick. Generating music using concepts from schenkerian analysis and chord spaces. Technical report, Tech. rep., Yale University, 2010. 135, 158, 168

[318] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 141, 151

[319] S. Raczynski, E. Vincent, F. Bimbot, and S. Sagayama. Multiple pitch transcription using dbn-based musicological models. In *2010 Int. Society for Music Information Retrieval Conf.(ISMIR)*, pages 363–368, 2010. 152

[320] R. Ramirez and A. Hazan. A tool for generating and explaining expressive music performances of monophonic jazz melodies. *International Journal on Artificial Intelligence Tools*, 15(04):673–691, 2006. 158

[321] R. Ramirez and A. Hazan. Inducing a generative expressive performance model using a sequential-covering genetic algorithm. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2159–2166. ACM, 2007. 158, 167

[322] M. Ramona, G. Cabral, and F. Pachet. Capturing a musician's groove: Generation of realistic accompaniments from single song recordings. In *IJCAI*, pages 4140–4142, 2015. 172

[323] A. Rangel, C. Camerer, and P. R. Montague. A framework for studying the neurobiology of value-based decision making. *Nature reviews neuroscience*, 9(7):545, 2008. 178

[324] C. Raphael. Automated rhythm transcription. In *ISMIR*, volume 2001, pages 99–107, 2001. 154

[325] C. Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *ISMIR*, pages 387–394, 2004. 165

[326] C. Raphael. A graphical model for recognizing sung melodies. In *ISMIR*, pages 658–663, 2005. 165

[327] C. Raphael. Demonstration of music plus one-a real-time system for automatic orchestral accompaniment. In *AAAI*, pages 1951–1952, 2006. 168

[328] R. Ratcliff and G. McKoon. The diffusion decision model: theory and data for two-choice decision tasks. *Neural computation*, 20(4):873–922, 2008. 61, 65, 73

[329] R. Ratcliff and F. Tuerlinckx. Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic bulletin & review*, 9(3):438–481, 2002. 65, 73

[330] A. Rauber, E. Pampalk, and D. Merkl. *Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity*. 2002. 160, 166

[331] L. Reboursière, O. Lähdeoja, T. Drugman, S. Dupont, C. Picard-Limpens, and N. Riche. Left and right-hand guitar playing techniques detection. In *NIME*, 2012. 170

[332] B. M. . M. M. . C. R. . D. L. . D. Repetto. Librosa. *https://github.com/bmcfee/librosa*, 2014. 68, 87

[333] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006. 141

[334] M. B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994. 42

[335] D. Rizo, P. J. Ponce de León, C. Pérez-Sancho, A. Pertusa, and J. Iñesta. A pattern recognition approach for melody track selection in midi files. 2006. 160

[336] M. E. Rodríguez López, A. Volk, and D. Bountouridis. Multi-strategy segmentation of melodies. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, pages 207–212. ISMIR press, 2014. 151

[337] M. Rohrmeier. A generative grammar approach to diatonic harmonic structure. In *Proceedings of the 4th sound and music computing conference*, pages 97–100, 2007. 168

[338] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635, 2011. 108

[339] H. Rump, S. Miyabe, E. Tsunoo, N. Ono, and S. Sagayama. Autoregressive mfcc models for genre classification improved by harmonic-percussion separation. In *ISMIR*, pages 87–92. Citeseer, 2010. 145

[340] S. Russell, P. Norvig, and A. Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995. 137

[341] P. Ruvolo and E. Eaton. Ella: An efficient lifelong learning algorithm. *ICML (1)*, 28:507–515, 2013. 42

[342] J. Salamon, B. Rocha, and E. Gómez. Musical genre classification using melody features extracted from polyphonic music signals. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 81–84. IEEE, 2012. 145

[343] C. S. Sapp. Online database of scores in the humdrum file format. In *ISMIR*, pages 664–665, 2005. 160

[344] C. S. Sapp. Comparative analysis of multiple musical performances. In *ISMIR*, pages 497–500, 2007. 156

[345] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001. 10

[346] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1331–1334. IEEE, 1997. 143

[347] J. Schluter and S. Bock. Improved musical onset detection with convolutional neural networks. In *Acoustics, speech and signal processing (icassp), 2014 ieee international conference on*, pages 6979–6983. IEEE, 2014. 154

[348] E. Schmidt and Y. Kim. Learning rhythm and melody features with deep belief networks. In *ISMIR*, pages 21–26, 2013. 165

[349] E. M. Schmidt and Y. E. Kim. Modeling musical emotion dynamics with conditional random fields. In *ISMIR*, pages 777–782, 2011. 165

[350] D. Schnitzer, A. Flexer, G. Widmer, and M. Gasser. Islands of gaussians: The self organizing map and gaussian music similarity features. 2010. 148

[351] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 41

[352] B. W. Schuller, C. Kozielski, F. Weninger, F. Eyben, G. Rigoll, et al. Vocalist gender recognition in recorded popular music. In *ISMIR*, pages 613–618, 2010. 161

[353] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015. 105

[354] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 10, 105

[355] J. Serra, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008. 153, 154

[356] J. Shawe-Taylor and A. Meng. An investigation of feature models for music genre classification using the support vector classifier. 2005. 163

[357] A. Sheh and D. P. Ellis. Chord segmentation and recognition using em-trained hidden markov models. 2003. 150, 164

[358] S. Sigtia, E. Benetos, S. Cherla, T. Weyde, A. Garcez, and S. Dixon. An rnn-based music language model for improving automatic music transcription. In *http://www. terasoft. com. tw/conf/ismir2014//proceedings% 5CISMIR2014_Proceedings. pdf*, pages 53–58. International Society for Music Information Retrieval, 2014. 152

[359] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014. 10

[360] S. Sinclair, M. Droettboom, and I. Fujinaga. Lilypond for pyscore: Approaching a universal translator for music notation. In *ISMIR*, pages 387–388, 2006. 160

[361] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010. 42

[362] M. Skalak, J. Han, and B. Pardo. Speeding melody search with vantage point trees. In *ISMIR*, pages 95–100, 2008. 147

[363] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In *International Symposium on Music Information Retrieval (ISMIR)*, 2008. 148

[364] T. F. Smith and M. S. Waterman. Comparison of biosequences. *Advances in applied mathematics*, 2(4):482–489, 1981. 153

[365] J. Solis, A. Takanishi, and K. Hashimoto. Development of an anthropomorphic saxophone-playing robot. In *Brain, Body and Machine*, pages 175–186. Springer, 2010. 168

[366] Y. Song, S. Dixon, and M. Pearce. Evaluation of musical features for emotion classification. In *ISMIR*, pages 523–528, 2012. 146

[367] Y. Song, S. Dixon, and M. Pearce. A survey of music recommendation systems and future perspectives. In *9th International Symposium on Computer Music Modeling and Retrieval*, 2012. 147

[368] J. Spenwyn, D. J. Barrett, and M. D. Griffiths. The role of light and music in gambling behaviour: An empirical pilot study. *International Journal of Mental Health and Addiction*, 8(1):107–118, 2010. 72

[369] E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the Association for Information Science and Technology*, 60(3):538–556, 2009. 134

[370] R. Stenzel and T. Kamps. Improving content-based similarity measures by training a collaborative model. In *ISMIR*, pages 264–271, 2005. 148

[371] P. Stone, G. A. Kaminka, S. Kraus, J. S. Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010. 103

[372] L. Su, L.-F. Yu, and Y.-H. Yang. Sparse cepstral, phase codes for guitar playing technique classification. In *ISMIR*, pages 9–14, 2014. 146, 170

[373] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. 13, 14, 43, 45

[374] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 8, 10, 34, 94, 104

[375] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000. 10, 105

[376] C. Szepesvári and R. Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd international conference on Machine learning*, pages 880–887. ACM, 2005. 56

[377] S.-L. Tan, P. Pfordresher, and R. Harré. *Psychology of music: From sound to significance*. Psychology Press, 2010. 16, 25

[378] M. Taramigkou, E. Bothos, K. Christidis, D. Apostolou, and G. Mentzas. Escape the bubble: Guided exploration of music preferences for serendipity and novelty. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 335–338. ACM, 2013. 37, 157

[379] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009. 42

[380] A. Tellegen, D. Watson, and L. A. Clark. On the dimensional and hierarchical structure of affect. *Psychological Science*, 10(4):297–303, 1999. 145

[381] R. E. Thayer. *The biopsychology of mood and arousal*. Oxford University Press, 1990. 145

[382] B. Thom. Bob: an interactive improvisational music companion. In *Proceedings of the fourth international conference on Autonomous agents*, pages 309–316. ACM, 2000. 142

[383] B. Thom. Unsupervised learning and interactive jazz/blues improvisation. In *AAAI/IAAI*, pages 652–657, 2000. 158

[384] B. Thom. Machine learning techniques for real-time improvisational solo trading. In *ICMC*, 2001. 158, 168

[385] M. Tiemann, S. Pauws, and F. Vignoli. Ensemble learning for hybrid music recommendation. In *ISMIR*, pages 179–180, 2007. 146, 167

[386] P. Toiviainen and T. Eerola. Classification of musical metre with autocorrelation and discriminant functions. In *ISMIR*, pages 351–357, 2005. 146

[387] N. Tokui, H. Iba, et al. Music composition with interactive evolutionary computation. In *Proceedings of the 3rd international conference on generative art*, volume 17, pages 215–226, 2000. 167

[388] S. M. Tom, C. R. Fox, C. Trepel, and R. A. Poldrack. The neural basis of loss aversion in decision-making under risk. *Science*, 315(5811):515–518, 2007. 72

[389] B. Tomasik, J. H. Kim, M. Ladlow, M. Augat, D. Tingle, R. Wicentowski, and D. Turnbull. Using regression to combine data sources for semantic music discovery. In *ISMIR*, pages 405–410, 2009. 161

[390] G. Tomlinson. Musicology, anthropology, history. In *The cultural study of music*, pages 81–94. Routledge, 2012. 149

[391] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009. 42

[392] C. J. Tralie and P. Bendich. Cover song identification with timbral shape sequences. 2015. 153

[393] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008. 145

[394] W.-H. Tsai, H.-M. Yu, H.-M. Wang, et al. Query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *ISMIR*, volume 5, pages 183–190, 2005. 160

[395] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106, 2004. 41

[396] D. Turnbull, G. R. Lanckriet, E. Pampalk, and M. Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *ISMIR*, pages 51–54, 2007. 167

[397] A. Tversky and D. Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323, 1992. 72

[398] G. Tzanetakis and F. Cook. A framework for audio analysis based on classification and temporal segmentation. In *EUROMICRO Conference, 1999. Proceedings. 25th*, volume 2, pages 61–67. IEEE, 1999. 10

[399] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002. 10

[400] D. Urieli and P. Stone. A learning agent for heat-pump thermostat control. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013. 22

[401] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013. 166

[402] S. Van Herwaarden, M. Grachten, and W. B. De Haas. Predicting expressive dynamics in piano performances using neural networks. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, pages 45–52. International Society for Music Information Retrieval, 2014. 156, 166

[403] P. van Kranenburg. Assessing disputed attributions for organ fugues in the js bach (bwv) catalogue. *Computing in Musicology*, 15:120–137, 2008. 142

[404] R. Vogl, M. Dorfer, and P. Knees. Recurrent neural networks for drum transcription. In *ISMIR*, pages 730–736, 2016. 166

[405] J. Von Neumann. Probabilistic logic. *California Institute Technology*, 1952. 142

[406] J. K. Vuoskoski and T. Eerola. The role of mood and personality in the perception of emotions represented by music. *Cortex*, 47(9):1099–1106, 2011. 64

[407] J.-C. Wang, H.-S. Lee, H.-M. Wang, and S.-K. Jeng. Learning the similarity of audio music in bag-of-frames representation from tagged music data. In *ISMIR*, pages 85–90, 2011. 148

[408] J.-C. Wang, M.-C. Yen, Y.-H. Yang, and H.-M. Wang. Automatic set list identification and song segmentation for full-length concert videos. In *ISMIR*, pages 239–244, 2014. 147

[409] X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: A reinforcement learning approach. *arXiv preprint arXiv:1311.6355*, 2013. 37, 157

[410] X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1):7, 2014. 169

[411] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *ISMIR*, 2004. 145

[412] K. West and S. Cox. Finding an optimal segmentation for audio genre classification. In *ISMIR*, pages 680–685, 2005. 167

[413] J. Weston, S. Bengio, and P. Hamel. Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. *Journal of New Music Research*, 40(4):337–348, 2011. 171

[414] C. N. White, A. Kapucu, D. Bruno, C. M. Rotello, and R. Ratcliff. Memory bias for negative emotional words in recognition memory is driven by effects of category membership. *Cognition & emotion*, 28(5):867–880, 2014. 65

[415] C. N. White, E. Liebman, and P. Stone. Decision mechanisms underlying mood-congruent emotional classification. *Cognition and Emotion*, 32(2):249–258, 2018. PMID: 28271732. 60

[416] C. N. White and R. A. Poldrack. Decomposing bias in different types of simple decisions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 40(2):385, 2014. 61, 62

[417] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996. 41

[418] M. Wolterman. Infrastructure-based collision warning using artificial intelligence, Jan. 8 2008. US Patent 7,317,406. 134

[419] I. Xenakis. Free stochastic music from the computer. programme of stochastic music in fortran. *Gravesaner Blätter*, 26:54–92, 1965. 135

[420] I. Xenakis. *Formalized music: thought and mathematics in composition*. Number 6. Pendragon Press, 1992. 135

[421] G. Xia, J. Tay, R. Dannenberg, and M. Veloso. Autonomous robot dancing driven by beats and emotions of music. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 205–212. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 158

[422] Z. Xing, X. Wang, and Y. Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *ISMIR*, pages 445–450, 2014. 147, 157

[423] C. Xu, N. C. Maddage, X. Shao, F. Cao, and Q. Tian. Musical genre classification using support vector machines. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–429. IEEE, 2003. 163

[424] T. B. Yakar, P. Sprechmann, R. Litman, A. M. Bronstein, and G. Sapiro. Bilevel sparse models for polyphonic music transcription. In *ISMIR*, pages 65–70, 2013. 152

[425] D. Yang and W.-S. Lee. Disambiguating music emotion using software agents. In *ISMIR*, volume 4, pages 218–223, 2004. 145

[426] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions. *arXiv preprint arXiv:1703.10847*, 2017. 160

[427] A. R. H. Yeshurun. Midi music genre classification by invariant features. 2006. 160

[428] K. Yoshii and M. Goto. Infinite latent harmonic allocation: A nonparametric bayesian approach to multipitch analysis. In *ISMIR*, pages 309–314, 2010. 151, 165

[429] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *ISMIR*, volume 6, page 7th, 2006. 146

[430] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Improving efficiency and scalability of model-based music recommender system based on incremental training. In *ISMIR*, pages 89–94, 2007. 146

[431] W. You and R. B. Dannenberg. Polyphonic music note onset detection using semi-supervised learning. In *ISMIR*, pages 279–282, 2007. 154

[432] L. A. Zadeh. Fuzzy logic and approximate reasoning. *Synthese*, 30(3-4):407–428, 1975. 142

[433] M. Zhao and S.-C. Zhu. Sisley the abstract painter. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 99–107. ACM, 2010. 134

[434] E. Zheleva, J. Guiver, E. Mendes Rodrigues, and N. Milić-Frayling. Statistical models of music-listening sessions in social media. In *Proceedings of the 19th international conference on World wide web*, pages 1019–1028. ACM, 2010. 157

[435] X. Zhou and A. Lerch. Chord detection using deep learning. In *Proceedings of the 16th ISMIR Conference*, volume 53, 2015. 152

[436] I. Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010. 41

[437] A. Zumbansen, I. Peretz, and S. Hébert. The combination of rhythm and pitch can account for the beneficial effect of melodic intonation therapy on connected speech improvements in broca's aphasia. *Frontiers in human neuroscience*, 8, 2014. 64