THE UNIVERSITY
of EDINBURGH

Coarse Preferences: Representation, Elicitation, and Decision Making

Pavlos Andreadis

THE UNIVERSITY
of EDINBURGH

Thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy in Informatics
to the
University of Edinburgh — 2018

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, either in whole or in part, in any previous application for a degree. Except where otherwise acknowledged, the work presented is entirely my own.

Pavlos Andreadis

November 2018

Abstract

In this thesis we present a theory for *learning* and *inference* of user preferences with a novel hierarchical representation that captures *preferential indifference*. Such models of 'Coarse Preferences' represent the space of solutions with a uni-dimensional, discrete latent space of *'categories'*. This results in a partitioning of the space of solutions into *preferential equivalence classes*. This hierarchical model significantly reduces the computational burden of learning and inference, with improvements both in computation time and convergence behaviour with respect to number of samples. We argue that this Coarse Preferences model facilitates the efficient solution of previously computationally prohibitive recommendation procedures. The new problem of *'coordination through set recommendation'* is one such procedure where we formulate an optimisation problem by leveraging the factored nature of our representation. Furthermore, we show how an on-line learning algorithm can be used for the efficient solution of this problem. Other benefits of our proposed model include increased quality of recommendations in *Recommender Systems* applications, in domains where users' behaviour is consistent with such a hierarchical preference structure. We evaluate the usefulness of our proposed model and algorithms through experiments with two recommendation domains - a clothing retailer's online interface, and a popular movie database. Our experimental results demonstrate computational gains over state of the art methods that use an additive decomposition of preferences in on-line active learning for recommendation.

Acknowledgements

I would like to thank my supervisors, Subramanian Ramamoorthy and Michael Rovatsos, for their well-thought-out advice and consistent support throughout the entirety of my doctoral studies. I am grateful, not only for their efforts put into my studies, but also their trust in allowing me further teaching and support roles, and, ultimately, in allowing me to study under them at the University of Edinburgh.

I would also like to thank Ya'akov (Kobi) Gal, of the Ben-Gurion University of the Negev, for his wise outside perspective while acting as my yearly external reviewer. A big thank you should also be extended to my colleagues at the EU FP7 SmartSociety project, and especially to my friends Dimitris Diochnos and Sofia Ceppi, with whom I have spent many hours, both happy and frustrating.

I would like to extend my thanks to the people at Mallzee, and especially Martina Pugliese for her help and insight.

Edinburgh has a spirit of its own, with which I am very much in love. A spirit only enhanced by that of the amazing people I have come to know and befriend since first setting foot here, one summer of 2013. And I would never have seen it if it wasn't for my friends Nikos Thrapsaniotakis and Despina Koukoula.

This list would not be complete if I did not extend a thank you to my family for their

love and support. Which brings me to the most important part:

This work is dedicated to my nephews Evangelos, Pavlos, *and* Orpheas.
May you grow up to be men worthy of respect and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Preference Elicitation is the process of selecting and presenting user(s) with queries such that their responses can be used to update our belief over a model of the user(s)'s preferences (Fishburn, 1999). Essentially, it is the application of Active Learning (Rubens *et al.*, 2015) in the domain of human preferences, and can therefore be used in a Recommender System (Jannach *et al.*, 2010). The space of queries to consider when selecting which to present to the user is at least as large as the number of solutions available for recommendation, and can often be a continuous space, especially when solutions are generated according to certain criteria rather than being selected from an available set.

Typically, preference models do not explicitly capture a user's indifference among certain alternatives and, until learning has converged, do not provide information about such a relation between those solutions. Explicitly learning such indifference relations would allow a Recommender System to treat corresponding alternatives as interchangeable (other criteria withstanding) and potentially significantly reduce the effective space of available queries.

Though some work is present in the literature that groups solutions together before making a recommendation, it either assumes that users are capable of indicating these groups directly, or does not consider the constraints on-line learning applies. Specifically, that the grouping needs to be the same for all users if we are to maximise computational gains, that the utility users have for each group should not be constrained to be the same, and that the groups should not be defined over predetermined sets of items but a more general model.

The theory of *Coarse Preferences*, as developed in this thesis, addresses the representation of user preferences with utility functions defined over a space of disjoint categories of solutions, even when this is shared by all users. This restricts the space of queries, during preference elicitation, and items for recommendation to those produced from the space of categories, leading to significant computational gains, especially when considering scaling our algorithms to large or continuous item spaces, as well as an increase in recommendation quality in the tested applications.

1.1 Online and Multi-user Recommendation

In 1992 Goldberg *et al.* presented the first recommender system (Resnick and Varian, 1997). Two years later Resnick *et al.* (1994) would write about "a deceptively simple idea", that "people who agreed in their subjective evaluation... are likely to agree again in the future". Fifteen years later, Netflix announced the winners of their million dollar competition (Bell *et al.*, 2009), marking what is considered broadly to be the moment that pushed Recommender Systems research into the academic and societal mainstream (Jannach *et al.*, 2010; Ricci *et al.*, 2010). The winning BellKor team (Bell *et al.*, 2007) deployed a variation on *Collaborative Filtering*, a technique based on the similarities between users' ratings, which up to this day remains the go-to approach.

Generally, recommender systems aim to learn representations of what their users prefer, and then use these to inform their decision of what, if any, items they will present their user with. The intention behind such a *recommendation* is often to provide the user with something they will *like*. This is the case with the online clothings retailer scenario we will be examining across chapters (and solve in Chapter 4). The relevant application recommends one item of clothing at a time to its user, who then either *accepts* or *rejects* this recommendation. Users expect the system to adapt its recommendations during a single visit, calling therefore for learning their preferences *online*, i.e. by incorporating evidence effectively.

Recommendations can also be the interaction of choice in more complex decision problems, involving multiple criteria of optimality and users. One such scenario is our running example of *coordination through set recommendation* over a popular movie data set (which we solve in Chapter 5). Here, each of a number of users is presented with a personalised set of movie recommendations, from which they select one according to their preferences. However, whether they get to watch that movie or not depends on the selections of all the users, since movies are only shown if a predefined quota of users is met. Such interactions happen repeatedly, though not necessarily with the same exact users, calling for online learning procedures, if any improvement is to be made across sessions. Recommender systems have mostly shied away from such multi-user coordination scenarios, which are typically handled by Graph Theory as a *matching* problem (e.g. in Gusfield and Irving (1989); Mourad Baïou (2002); Dickerson *et al.* (2012)), where the system would match users to movies, not allowing for empowering the user with choice, or for the system to learn from that choice.

Both of our examined scenarios feature the need for an online learning procedure, and it is for this reason that we will not focus on collaborative filtering approaches, which typically require a large offline computation in order to update the learned preference model. We will instead look into the area of Decision Theory, and

specifically preference elicitation, which has a long history of addressing the learning of preferences, in an online, typically active, manner. Such models explicitly represent our belief over a user's preferences over some representation of items, which, as argued in Viappiani and Boutilier (2009b) is vital in determining the most appropriate recommendations, especially in *set recommendation*, the recommendation of sets of items.

With *online* learning we refer to learning procedures that update their model incrementally, by incorporating evidence as it is encountered, rather than by running a learning procedure of the entirety of evidence encountered so far, as is the case with *offline* learning. *Active* learning refers to learning procedures where the system has some control over which possible interactions to make available to users, and selects among these using some optimisation criterion or heuristic that relates to the quality of information inferred from the interaction.

Despite these advantages, preference elicitation has not seen widespread use in recommender systems, likely because of the computational costs associated with it (Boutilier, 2002), in terms of both computational time, and quick convergence to accurate representations. The latter is significantly aggravated in online *active* learning scenarios, where we present queries strategically, aiming to maximise the information we get from the user's response, and even more so when most modern recommender systems feature large, dynamic catalogues of available items or, even worse, need to recommend services that are composed 'on the fly'. Thus, a big benefit of these approaches is effectively lost.

If we had access to user preference models that are not only accurate, but actually allow us to reduce the effective cardinality of the space of items for recommendation, then this should allow preference elicitation to see much more mainstream use, especially given all the other benefits this approach has to offer. The hierarchical model presented in this work tackles these computational issues, enabling active online learning, and

the optimisation of complex multi-user problems. Furthermore, our learning procedure achieves significantly better accuracy of predictions for both of our examined scenarios.

1.2 The Cold-Start Problem

Two related issues that recommender systems designers need to consider are the *cold-start* item and user problems. Both problems refer to the absence of information on the interaction of users with items; either pre-existing users with new items (item cold-start) or new users with pre-existing items (user cold-start). Models that are solely based on the similarity between users' behaviour, will not have any information on which to base their recommendations in such scenarios.

'*In practice these problems cover important cases: new users should not be scared away by getting bad or no recommendations in the beginning, and new items should not have to wait until they are found and taken up by users by chance.*' (Marinho *et al.*, 2012).

Since preference elicitation learns *personalised* preference models, the *item* cold-start problem is easily handled, as long as the model is defined over a vector description of items. However, the user cold-start problem typically requires representing users as members of one of a set of *user types*, in order to handle new arrivals Chajewska *et al.* (1998).

As we will see in Chapters 3, 4, and 5, our procedures naturally generalise across users, without forfeiting the advantage of individual user profiles.

1.3 Hierarchical Factored Optimisation

In Chapters 2 and 5 of this thesis, we will see that there are important coordination/multiagent resource allocation problems that can be posed as problems of recommendation. Such problems involve making optimal decisions while considering multiple objective functions, derived from different users as well as from system specifications. Most work in recommender systems applications focusses on user satisfaction, typically modelled as the user's evaluation of their recommendation (though other metrics related to user satisfaction have also been used (Ricci *et al.*, 2010)).

If optimising for user satisfaction is a criterion we do not wish to compromise over, then it is not clear how to best formulate a multi-criteria problem that would also consider system level criteria. Typical approaches to solving such multi-criteria problems is either the construction of a multi-objective function for optimisation, which would represent a tradeoff between users' and system's preferences, or the re-definition of most objectives as constraints, with some bound over their value, and the subsequent optimisation over one of, or a combination of, the criteria (Keeney and Raiffa, 1993; Mardani *et al.*, 2015).

These two approaches represent two distinct types of problems: In the first case, we assume that we can give an explicit weighting of different criteria, typically linear. An example would be maximising a weighted sum of users' *social welfare*, typically the sum of their individual rewards, and expected profit.

In the second case, we are conceding that certain criteria will only be satisfied, while only a single criterion (or function of criteria) will be optimised for. This could mean, for example, minimising the number of resources used by the system, while guaranteeing a minimum expected evaluation for each user.

None of these approaches maps directly to the problem of considering a system criterion while not having to tradeoff in user satisfaction. This would require a hierarchical process, by which maximising user evaluations outputs a space of optimal recommendations.

If the hypothesis space for our user models allows for identifying spaces of optimal solutions, then we could introduce additional criteria without negatively impacting user satisfaction, and therefore important considerations such as *user retention* (Keiningham *et al.*, 2007).

1.4 Motivation

This work studies the potential benefits of explicitly modelling users' preferential indifference between solutions, in decision problems whose prime component is one or more users' preference functions, and with a focus on problems of recommendation and, more specifically, the online (active) learning of preferences for recommendation. The latter focus is mirrored in the examples and applications reviewed throughout the thesis.

This choice of motivation stems not only from an interest in improving the state of the art in the contemporary relevant field of Recommender Systems, but also from the desire to expand the scope of the field into more computationally complex problems, especially those involving decisions across many, strongly co-dependent, users.

The intuition driving this research is that the explicit modelling of indifference will allow for a principled way of *ignoring* solutions and that, if the set, or in some cases the more general 'space' of solutions, ignored in this way is *large* enough, then a *significant*

amount of computational time can be saved during inference, and methods for learning such models of preferences will converge faster to good representations.

A complementary view of our model is that of spaces of equivalent, interchangeable solutions, and Chapter 6 gives a brief analysis of how this can increase the robustness of applications where re-computation of optimal solutions is not an option.

1.5 Relation to Categorical Thinking

The concept of indifference in decision problems relates to the well-studied in Cognitive Science phenomenon of *Categorisation*. Specifically, as groups of items or solutions are treated as equivalent for one or more problems addressed by the user, he can be said to be indifferent between alternative solutions in that group or *category*.

It is important to make clear here, that the thesis does not attempt to propose a model of *how* users might form these categories, or even that the categories learnt through our procedures *are* the ones held in an individual user's mind. However, since users are known to map alternatives to categories, we find this to be a good indication of the possibility of learning a *good enough* mapping.

Though it is possible that some of the work presented in Chapter 3 can be utilised in solidifying this claim, we do not attempt to do so in this work. Instead, the thesis aims at providing tools that *work*, and that can be used in rationalising decisions made with them, under the understanding that their models reflect user behaviour and wants, to the best of the decision maker's knowledge.

1.6 Coarse Preferences

We present a model of user preferences which gives rankings of outcomes by utilising a latent space of *categories*, each one representing a subspace of the original outcome space, such that they collectively partition this space, while having no overlapping elements. The resulting mapping from the space of outcomes, or solutions, to that of categories summarises the original space, allowing us to define a preference function of potentially much smaller complexity.

The gains in complexity, as well as the overall usefulness of the model, depends on to what extent such a *coarse* representation can be learned, on its domain's cardinality, and on how accurately it can represent user behaviour. Chapters 4 and 5 demonstrate that these conditions can be met, by providing both typical and novel recommendation scenarios, with data drawn from real user interactions on online platforms.

What makes the model presented in this work of particular interest, is that even though it is defined by drastically fewer parameters than a preference model with variable-based decomposition, it manages significantly better performance in terms of accuracy for both scenarios examined. In Chapter 4 we take advantage of this to maximise both the utility of recommendations to the user and the profit achieved by the system. In Chapter 5 the increased accuracy translates to better performance in terms of social welfare.

Inference with *coarse preferences*, as we term preference functions defined by the proposed approach, is significantly faster when compared to previous models for representing preferences; such as those based on additive independence and generalised additive independence.

An interesting aspect of our approach, is that it naturally lends itself to factored

optimisation problems. Where other representations would lead to peaks of optimal solutions, our model results in entire subspaces of optimal solutions, between which the user, by definition, is indifferent.

Though the benefits described above are significant in the single user case (see Chapter 4), it is in multi-criteria, multi-user problems that most of their potential lies (see Chapter 5). By providing a coarse mapping that is consistent with all users over which a decision needs to be made, we can scale the benefits up. The computational benefits, in particular, are such that they enable otherwise prohibitively slow procedures to be run online. Learning for the problem of coordination through set recommendation for example, would not be practically feasible without our proposed model.

The main contributions of this thesis are:

- A decision theoretic model with explicit representation of preferential indifference, which allows for the significant speedup of inference;

- An algorithmic approach for learning such preference models which achieves significantly better performance when compared to the relevant state of the art preference elicitation procedure, in single and multi-user recommendation problems, in terms of computational time and accuracy of predictions;

- Two sets of experiments demonstrating the validity of our claims and providing insight into how our model behaves under different circumstances.

In addition to these contributions, our thesis provides details on how to incorporate our model into multi-user recommendation problems, especially when there is interdependency between their solutions. We also introduce the new problem of coordination

through set recommendation, which exemplifies how reduced computational time can enable novel applications. Furthermore, we provide a new recommender system data set extracted from the online retailer Mallzee (Andreadis, 2016).

1.7 Thesis Overview

Chapter 2 provides the necessary background for framing the contributions in the following chapters. It outlines the area of recommender systems, with an emphasis on preference elicitation, as it relates to problems of online learning of preferences.

The chapter develops preference elicitation from the point of view of Decision Theory, and emphasises preference representations that take the form of utility functions, and how they can be used for addressing decision problems over single or multiple users.

Chapter 3 proposes our model of *coarse preferences* and examines its use for inference. We prove its compatibility with the von Neumann and Morgenstern (1953) Expected Utility Theorem, and examine the problem of inference for single and multiple users, and user types. The latter two require specialised treatment in order to maintain certain desirable properties. Among those, computational time reduction at scale, and quicker convergence to good solutions.

Chapter 4 develops an online learning procedure for coarse preferences, involving an important one time offline procedure, and applies it to a preference elicitation problem in a recommender system defined over a real-world online retailer's data set. The offline procedure learns the latent space of categories, over which the online procedure elicits users' preferences. A common latent space is learnt for all users, including new arrivals.

The procedure outperforms the state of the art in online utility based preference elicitation procedures, both computationally and in terms of quality of recommendation, leading to more satisfied users and greater financial profit.

Chapter 5 presents the computationally demanding problem of *coordination through set recommendation*, and shows how online learning is enabled in the setting by our approach. The chapter gives special consideration to how the factorisation of the optimisation problem allows for scaling up the benefits described in the previous chapter.

Chapter 6 concludes, summarising positive outcomes, as well as listing any concerns, such as the offline computation and non-personalised nature of our learnt coarse mappings. Directions for future work are provided, with emphasis on tackling these concerns, and ideas on how to best utilise coarse preferences.

Chapter 2

Background

Recommender systems are automated procedures and techniques for suggesting items to users. This chapter presents the traditional approaches to recommender systems, along with some more recent work, and outlines basic concepts and procedures. Since we are interested in the online, potentially active, learning of users' preferences, we will focus on approaches that allow for this, and only give a brief outlining of other recommender system approaches. At the end of the chapter's introduction we present two case studies which exemplify this focus, acting as examples for both single and multi-user recommendation in online settings.

2.1 Introduction

Largely because of the expansion of the Internet, people are often presented with amounts and type of information that are hard to handle efficiently and in time. Recommender systems have been developed in order to address these issues. Mahmood and Ricci (2009) define a Recommender System (RS) as an *intelligent application*

which assists users in their information-seeking tasks, by suggesting those items that best suit their needs and preferences. Ricci *et al.* (2010) redefine recommender systems as *software tools and techniques providing suggestions for items, to be of use to a user*.

The above definitions make use of the term *item*, which is commonly used in the literature to refer to what the system recommends to the user. Most recommender systems tend to focus on a specific type of item (e.g. books, news, or medical procedures) and feature a design, user interface, and techniques customized for that application. In this thesis we will use the terms *possible outcome*, *candidate solution* and *item* interchangeably, except where noted, to refer to the available options the user has to select from, regardless of what has been recommended.

An RS is in many ways an automated salesman; it probes the user for information, compares them to previous customers, establishes their preferences, shows and suggests items and, as its motivation should be, tries to sell them something. What item(s) it attempts to promote will depend on its expectations over the user's behaviour, and its owners goals and motivations. These are often more complex or altruistic than what a price tag might indicate. Of course, recommender systems are not limited in use to online shops and have applications in areas such as e-learning (Zaíane, 2002) and healthcare (Duan *et al.*, 2011), among others.

Traditional recommender systems tend to ignore the system's goals and act as a predictor of whether, and sometimes how much, a user would prefer an item. As mentioned above, the two main motivations for someone to make use of a RS are *information overload* and the possible *lack of competence or experience* in the area of application (Ricci *et al.*, 2010). Herlocker *et al.* (2004b) define eleven popular tasks that a RS can assist in implementing. These include *finding some/all good items*, *just browsing*, *self-expression*, *helping/influencing others* and *finding a credible recommender*. This diversity of use for a RS means that setting one up can be arbitrarily complex. The approaches described below are all built around providing

good recommendations given known information, with issues such as the exploration of alternatives by the user and further enrichment/updating of her profile only implicitly addressed.

Jannach *et al.* (2010) divide recommender systems into three main categories: Collaborative, Content-based, and Knowledge-based, further classifying real-world applications as hybridizations based on the original classification by Burke (2007). Before presenting the different approaches, we outline two running examples which the thesis directly addresses across chapters. As the different recommendation approaches are described, we will consider how each can potentially tackle these problems. Consistent with the focus of this thesis, these examples require the online, computationally efficient learning of user preferences.

2.1.1 Case Studies

The thesis is concerned with the online learning of user preferences, in the form of a utility function, and the inference of optimal decisions given such a model, especially with dynamic or compositional available item spaces for recommendation. With *dynamic*, we refer to item spaces that are not constant in time, such that any preference model that is explicitly defined over a set of items (e.g. standard collaborative filtering) would have to be periodically reinstantiated. A good example would be an internet shop where products are added and removed by independent agents with little or no central control by the system. For example Amazon Linden *et al.* (2003). With *compositional*, we refer to item spaces that are not known during offline computations, and are rather constructed by the interactions of the system and users within a given, potentially dynamic, context. A good example would be a ridesharing application, where the space of available recommendations could be defined by attributes such as departure time, risk of delay, and distance travelled (Agatz *et al.*, 2011, 2012; Andreadis *et al.*, 2016)

Though such items can be described by a set of attributes, with bounds over the domain of each attribute, the specific items available only become known as the interaction of the user(s) with the system takes place. The thesis is more specifically concerned with making learning and inference procedures over such problems more computationally efficient. With that in mind, we present two representative learning and inference scenarios, which will guide our analysis of the literature and later chapters. Our case studies, one of which involves single-user active learning and the other one the coordination of multiple users, both require adaptation to users' preferences after every interaction and, therefore, an online solution that will compute in time. This differs from the typical Recommender System application where a rating is predicted from models computed offline.

Case study 1: Clothing recommender system

Our first case study concerns the sequential recommendation of items of clothing, with binary user feedback. The system maintains a dynamic library of items, each characterised by assignments to a set of parameters such as type of clothing, primary colour, and intended gender. Interacting with the user consists of presenting him/her with an item, which the user proceeds to like or dislike. In the case of a 'like', the item is stored in a list for future user reference. We will assume that items in this list have a predetermined chance of leading to a sale of the item. This pattern repeats itself until the user decides to terminate the session. Our aim, as a system, is to present the user with items such that they, in anticipation of their responses, allow for quickly learning what items the user prefers, thus improving our future recommendation, in the sense that either the user will like them, or profits from sales are maximised. The scenario is an example of the sequential rating of presented items, which can be represented as *explicit global value queries* and which are addressed in Section 2.5.11.

This scenario is an instantiation of the typical preference elicitation problem, in which

the system sequentially selects a query to present to the user, with the aim of learning an accurate model of his/her preferences. This problem is examined in detail in Chapter 4 where we make use of the Mallzee dataset (Andreadis, 2016) which was procured for the purposes of this study.

Case study 2: Movie viewing recommender system

Our second case study examines a novel application of recommender systems for the coordination of user choices. Here, users join an online service that allows them to watch a movie together with other users. Each user has preferences over what movie they would like to watch and is incentivised to join a 'movie viewing' either for the social aspect or because it is the only way to access some limited feature, such as live commentary and discussion with the movie director. The available movies and type of service can vary dynamically, and further parameters such as time of showing might be taken into account. Upon joining a session on the application, users are each presented with a personalised list of movie showings, from which they each get to select their preferred option. Constraints over system resources mean that movie showings will only occur if a minimum number of users selects the corresponding option. The system's task is to optimally construct the personalised recommendation sets for all users in the session, given limited knowledge of the users' preferences. The criterion we will focus on is that of maximising expected utilitarian *social welfare*, i.e. the expected sum of users' individual utilities from joining a viewing. As each user selects a viewing from the set of recommendations, they are indicating a preference for that viewing above the others in the list. We will treat this interaction as the user responding to an *explicit global set recommendation*, as defined in Section 2.5.11. Updating our beliefs over user preferences online is important, since users can sequentially join a number of sessions before the system has had a chance to make any offline computations. However, because of the complexity of actively selecting the lists to

present the users with, we will not consider the active learning of users' preferences in this scenario.

This scenario is an example of a system that goes beyond the standard single-user RS that are typically encountered in the literature, and that considers interdependencies between user responses to their individual recommendations, specifically as concerns the outcome for each individual user and the system. Our primary concern as regards this type of problem is how to develop efficient RS approaches for it. However, to contextualise our contributions, we provide an overview of multi-agent resource allocation in Section 2.7. We further look into this problem in Chapter 5 where we make use of the MovieLens 20M Dataset (Harper and Konstan, 2015).

Running example

At this point we will start with a running example of someone using the Mallzee app (`Mallzee.com`) to browse for clothes. We select the Mallzee scenario for this purpose because the item descriptors have a clear real-world meaning, unlike the tag percentages of the MovieLens scenario.

Consider the user *Jon* logging in to the Mallzee app and sequentially being presented with individual items of clothing, on which he is to decide whether to swipe left for *reject* or right for *accept*. We will assume that Jon makes his decision by rating each presented item on a continuous scale of 0 to 1, and then comparing this number with an internalised threshold of, for example, 0.6. If the rating is equal to or above that threshold, the item is accepted. Otherwise, it is rejected.

As the first item is presented to Jon, he is asked to swipe on a *black*, *unisex scarf* of *brand AB*, for the retail *price* of 26 *pounds*, which includes *no discount*, and that is *out of stock*. All items presented to Jon will have a similar 8 parameter description by which

he must make a decision on the item. As we go through this chapter, and specifically in Section 2.5, we will provide examples of how Jon might make this decision given different preference models. We then meet Jon one last time in Chapter 3.

2.2 Collaborative Recommendation

Collaborative Recommender Systems (CRS) are built around the idea of exploiting an existing user community in order to provide recommendations. Past behaviour of similar users (or over similar items) is used to make a prediction of the user's interests. The technology behind CRS has matured significantly in recent years due to its widespread industrial use and, largely, because of the 2006 announcement of the Netflix contest (Bell *et al.*, 2007, 2009). Techniques in the area are often called *Collaborative Filtering* (CF) since they involve filtering for the best items given implicit collaboration between users. Beyond the stationarity of users' preferences, these approaches further assume that similarity between user preferences is preserved. CRS are by far the most well studied techniques in the field, largely because of their successful online application and the availability of real-world benchmarks with a simple data structure, namely a ratings matrix. Techniques in the field have been employed by Netflix (Bell *et al.*, 2007), Amazon.com (Linden *et al.*, 2003) and Google Hofmann (2004), among others. Two general approaches to CF can be distinguished: *nearest neighbour recommendation* and *matrix factorization*.

2.2.1 Nearest neighbour recommendation

Nearest Neighbour Recommendation (NNR) can be categorised into *user-based*, and *item-based*, the latter being one of the earliest methods developed (Maltz and Ehrlich, 1995; Schafer *et al.*, 2007). For the *user-based* case, consider a set of users $I = \{i\} \forall i \in$

$1,...,n$, a list of items $S = \{s\} \forall s \in 1,...,m$ and a $n \times m$ matrix of user-item interactions M with numeric values $m_{j,i} \in L \subseteq \Re$ (generally a ratings matrix). We identify a list of users I_i that had similar preferences to the *active user i* in the past. These similar users are often referred to as *peer users* or *nearest neighbours*. The active user is the user whose preferences we are currently investigating. For every item $s \in S$ that the user has not yet rated, a prediction is computed from the ratings for s assigned by the peer users.

A similarity metric is used for determining the set of peer users, with the *Pearson's correlation coefficient* being a common choice:

$$\text{sim}(i,i') = \frac{\sum_{s \in S}(m_{i,s} - \overline{m_i})(m_{i',s} - \overline{m_{i'}})}{\sqrt{\sum_{s \in S}(m_{i,s} - \overline{m_i})^2} \sqrt{\sum_{s \in S}(m_{i',s} - \overline{m_{i'}})^2}}, \forall i,i' \in I, \tag{2.1}$$

where $\overline{m_i}$ is the average rating given by user i. Eq. 2.1 gives values in $[-1,1]$ where the extrema indicate *strong negative* and *strong positive* correlation, respectively. The Pearson correlation coefficient explicitly takes users' different rating scales into account by subtracting the average from each user's ratings. The literature makes use of other similarity metrics such as *adjusted cosine similarity*, *Spearman's rank correlation coefficient* and the *mean squared difference* measure. An empirical study by Herlocker *et al.* (1999), however, shows that for user-based RS the Pearson coefficient outperforms other user comparison measures. Regardless, Pearson's measure does not take into account that similar preferences in some items might be more indicative of user similarity. In particular, agreement on controversial items should be a better predictor. *Inverse user frequency* (Breese *et al.*, 1998), and *variance weighting* (Herlocker *et al.*, 1999) are two approaches towards addressing this problem. Another factor not taken into account in Eq. 2.1 is the number of co-ratings between users. Herlocker *et al.* (1999) address this problem by introducing *significance weighting*.

Having computed the similarities between users one needs to define i's neighbourhood

I_i and the extent to which each user in the set is going to be taken into consideration when determining their rating prediction. Approaches include using a threshold value h to filter out distant users and/or limiting the size of I_i and then using a formula that takes user similarity into account:

$$\text{pred}(i,s) = \overline{m_i} + \frac{\sum_{i' \in I} \text{sim}(i,i')(m_{i,s} - \overline{m_b})}{\sum_{i' \in I} \text{sim}(i,i')} \tag{2.2}$$

Herlocker *et al.* (1999), Anand and Mobasher (2005) and Herlocker *et al.* (2004a) discuss the prediction trade-offs with user coverage and noise, in selecting a threshold value and neighbourhood size respectively.

We can now theoretically compute all item rating predictions for user i and recommend those with the highest value. However, real-world applications will maintain large rating databases with high sparsity, since every user will only get to rate a limited number of items. Moreover, it is unclear how to recommend items to users with no ratings, or how to recommend items that have not been rated. These problems are commonly referred to in the literature as *cold-start user* and *item*, or *new user* and *item*, problems, respectively.

Real-time predictions using the user-based approach are in most practical applications impossible, due to the large size and dynamic nature of the user set I. An alternative approach to CF is using *item-based NNR* where similarities between items become the basis for predicting user preferences. The standard similarity metric in item-based recommendation approaches, in contrast to the user-based case, is the *cosine similarity measure* (Jannach *et al.*, 2010), which measures the similarity between the rating vectors m of two users based on the angle between them:

$$\text{sim}(m_i, m_{i'}) = \frac{m_i \cdot m_{i'}}{|v_a|\,|v_b|} \tag{2.3}$$

Eq. 2.3 gets values in $[0,1]$, with 1 indicating strong similarity.

Item-based CF has been used by Amazon.com Linden *et al.* (2003) in their product recommendations. User-based CF does not scale well for large number of items and user, something which item-based applications circumvent by pre-computing the *item similarity matrix* offline. The prediction is made efficient in real-time since we only need access items that have been rated by the user. Though offline pre-computation is theoretically possible for the user-based case, the low number of overlapping ratings between users means that additional ratings may quickly influence the similarity value between users Sarwar *et al.* (2001). Both cases however, do not handle the new item cold-start problem.

2.2.2 Matrix factorization

Matrix Factorization (MF) constitutes an alternative approach to neighbourhood-based recommenders by transforming both item and user data to the same latent factor space. Essentially, this approach tries to explain ratings as the result of factors inferred from user feedback. The identified factors might or might not correspond to specific item or user properties (Koren *et al.*, 2009). One approach to MF, *Singular Value Decomposition* (SVM) (Golub and Kahan, 1965), was proposed for use in information retrieval by Deerwester *et al.* (1990) and later used in recommender systems (Sarwar *et al.*, 2000b; Goldberg *et al.*, 2004; Canny, 2002):

2.2.3 Data sparsity and the cold-start problem

Users typically provide ratings for only a few items causing rating matrices in real world applications to be very sparse. The literature includes various proposal for dealing with this, of which Matrix Factorisation is one. Pazzani (1999) presents a hybrid RS which exploits demographic user information when constructing a neighbourhood.

Huang *et al.* (2004) describe a graph-based method where recommendations are selected based on the paths between users and items. Breese *et al.* (1998) propose *default voting* when comparing two users, where ratings existing only for one user are compared with a default value. More recently, Wang *et al.* (2006) proposed the combination of user and item-based NN approaches, while also exploiting information over *similar item ratings made by similar users*.

The cold-start problem can be viewed as a special case of the sparsity problem (Huang *et al.*, 2004), and is a shorthand for the questions: How to recommend items to users that have not yet rated any items, and how to recommend items that have not yet been rated or bought. Hybrid approaches have been developed to deal with these problems that make use of external information (Adomavicius and Tuzhilin, 2005). The new-user problem can be handled by requiring new users to rate a set of items with an Active-Learning approach Rashid *et al.* (2002), Goldberg *et al.* (2004).

2.2.4 Active learning collaborative filtering

Considering how to utilise CF for Case Study 1, we could devise a Decision Tree over a user's rating to specific items. By running an offline CF calculation for each possible path down the tree, we can instantly pull up the correct recommendation in response to a user's rating actions. Even though the number of model instances that would need to be stored increases exponentially as we go down the tree, it would be enough to only store the recommendation policy for the user. However, this policy would not be able to consider the interactions of the system with other users, and would therefore largely relinquish the primary benefit of collaborative recommendation. Building a decision tree over all users' interactions with the system would lead to combinatorially explosive memory requirements. It should therefore be obvious why such an approach would be even less appealing for Case Study 2. Lastly, there is no straightforward way

of adapting this approach to new items, since the strategy would have been computed offline, given a predetermined set of items.

A number of approaches have been developed for *actively* presenting users with items to rate, i.e. making an explicit informed decision of which item to present to the user for rating, so that collaborative filtering can provide better results. However, these approaches either define a probabilistic model over a predetermined set of items, making it incompatible with our assumption of a dynamic or compositional item space, or do not allow for an online update. Elahi *et al.* (2016) present a categorisation of active CF approaches in their survey of the area. In their conclusions, they point to the lack of research into active online CF learning problems (which they term *sequential*, as opposed to *batch*, active learning problems), emphasising that they would allow for the quick adaptation of recommendations to the user.

Boutilier *et al.* (2012) provide a general framework for performing sequential active learning for CF, while considering the *expected value of information* (see Section 2.5.11) of a recommendation. Unfortunately, though they provide techniques for reducing the time needed for updating the preference model, the model update computations are still offline. The preference model is also such that it is specified over a predetermined set of items, and it is not clear how to adapt this to a dynamic or compositional item portfolio.

2.2.5 Other approaches

Other approaches to collaborative filtering include *association rule mining* (Sarwar *et al.*, 2000a; Lin, 2002), a technique used to identify rule-like relationship patterns in data, and *slope one predictors* (Lemire and Maclachlan, 2005), where a simple linear ratings relationship is estimated between item pairs. Association rule mining usually

requires a predetermined set of items, though Lin (2002) extend this to rules over dynamic sets of items. However, this approach does not lend itself to problems with compositional item-spaces. The method proposed in Lemire and Maclachlan (2005) does allow for updating the model online, even as new items are added to it, however this also does not account for compositional item spaces.

Chee *et al.* (2001) and Xue *et al.* (2005) propose the use of a k-means clustering algorithm for partitioning users into homogeneous groups. These procedures do not handle the new item and new user problems well, and do not allow for compositional item spaces. Chee *et al.* (2001) specifically propose a decision tree for partitioning users according to their ratings of a set of items. Jin *et al.* (2006) give a comprehensive survey of different probabilistic approaches and mixture models. However, these models are learnt offline by use of the Expectation-Maximisation algorithm.

2.3 Content-based Recommendation

Content-based recommenders construct a model of user preferences by analysing their set of previously rated items. This structured representation of user interests is then used when deciding upon the next recommendation. The process can be summarized as a comparison of user and item attributes, with similarity based metrics being the most common approach.

The set of items, or products, available for recommendation is often referred to as product *catalogues*. Items in the catalogue can be represented by a set of attributes, or features. When a user's preferences are described using the same set of features, the recommendation process is one of comparing item and user attributes. User profiles can be constructed by explicitly asking for their preferences, or requiring that they rate a set of items.

The act of recommendation involves judging the similarity between previously positively rated items and catalogue items. Similarity could be binary, in where items are checked for sharing specific properties that have been favoured in the past, or a similarity, or *overlap*, of involved *keywords* could be computed. In the latter case, the Dice coefficient could be used as a similarity metric:

$$\frac{2 \times |keywords(p_i) \cap keywords(p_j)|}{keywords(p_i) + |keywords(p_j)|}. \tag{2.4}$$

This, and other measures, can be found in detail in Baeza-Yates and Ribeiro-Neto (1999), Maimon and Rokach (2005) and Zanker *et al.* (2006).

2.3.1 Vector space model and TF-IDF

Content-based recommenders were originally developed for recommending text-based items such as webpages or e-mails. Information about items, or documents, is therefore rarely in the form of an attribute vector, but needs to be extracted from the text itself. One could list all words present in all documents and define a binary variable indicating its presence or absence from each item. Such an approach, however, would assign the same importance to each word in a document. Furthermore, longer documents will tend to have a larger overlap, which would end in them being recommended more often.

The *Term Frequency - Inverse Document Frequency* (TF -IDF) (Salton *et al.*, 1975) approach is an established technique in the field for dealing with these issues above. Documents are encoded into multidimensional Euclidean space vectors. Dimensions correspond to the keywords, or *terms* or *tokens*, in the documents. Coordinates in each dimension are calculated as the product of *term frequency* and *inverse document frequency*.

TF represents the frequency of term appearance in a document and takes the document length into account, so as to minimize its effect on the recommendation. Several schemes have been proposed for normalization (e.g. in Chakrabarti (2002), Adomavicius and Tuzhilin (2005), Pazzani and Billsus (2007)). Various improvements have been proposed for the original vector space model. Perhaps the most straightforward approach entails the removal of all propositions, articles and other *stop-words*. *Stemming*, or *conflation*, is another technique where common-root words are replaced by a single variant. A similar approach can be applied for *synonyms*. Such approaches, though useful, risk stripping words of their discerning qualities and, therefore, matching irrelevant documents (Chakrabarti, 2002). *Size cut-offs* is another approach, where only a pre-specified number of most informative words is used (Pazzani and Billsus, 1997). Lastly, one could define *phrases* as terms, with the intuition that they will be more descriptive than single words (Chakrabarti, 2002), and is a basic approach for trying to capture *context*.

2.3.2 Similarity-based retrieval

Whereas collaborative filtering searches for items that similar users have liked in the past, content-based approaches recommend items that are similar to those the user has liked in the past. Similarity-based retrieval refers to techniques employing the vector space model. The most common approaches are relevance feedback model, and nearest neighbours.

The similarity of user and item attribute vectors can estimated by use of cosine similarity measure (Eq. 2.3). The prediction for an unobserved item can then be made by *vote* of the k most similar items (Allan *et al.*, 1998). Approaches, besides limiting the size k of the neighbourhood, include a minimum similarity threshold, weighting items according to their similarity, and binary transformation of ratings. Billsus *et al.*

(2000) implements a k nearest neighbour approach (kNN) where the system maintains separate user profiles for short- and long-term interests. kNN-based methods are simple to implement, adapt quickly to recent changes and require only a small number of ratings to function. However, experiments show that pure kNN approaches often suffer from low predictive accuracy (Jannach *et al.*, 2010).

Rocchio's relevance feedback method was developed for the information retrieval system SMART (Salton, 1971). Besides querying the system, SMART users could also provide feedback on the relevance of retrieved items through rating. The method employs a loop where the initial query q_i is repeatedly refined to q_{i+1} by weighted addition of the vectors of relevant documents and subtraction of non-relevant documents. In order to accomplish this, rated document are first separated into two groups D^+ and D^-, representing relevant and non-relevant documents, respectively. The proposed formula was

$$q_{i+1} = \alpha\, q_i + \beta\Big(\frac{1}{|D^+|}\sum_{d^+\in D^+} d^+\Big) - \gamma\Big(\frac{1}{|D^-|}\sum_{d^-\in D^-} d^-\Big), \tag{2.5}$$

where α, β and γ are parameters chosen by the system designer. Though intuitive, this model is not theoretically sound, as pointed out by Pazzani and Billsus (2007). Experimental results however show that the feedback mechanism improves results (Koenemann and Belkin, 1996). Salton and Buckley (1997), and Buckley *et al.* (1994) provide experimental evaluations for different parameter variations.

2.3.3 Classification approaches

An alternative to the vector space model of the previous paragraph, is to view the recommendation task as a classification problem. This formulations allows for use

of various *supervised learning* procedures. *Supervised learning* refers to machine learning tasks relying on labelled training data (Mehryar *et al.*, 2012).

Probabilistic classification methods were the most prominent developed in early text classification applications. Based on naive Bayes, these assumed conditional independence with respect to term occurrences. Pazzani and Billsus (1997) give a straightforward application of naive Bayes for binary classification. Though the core assumption of conditional independence does not hold in practice, naive Bayes has been shown to lead to good results (McCallum and Nigam, 1998). An advantage of the approach is that it is easily updated and learning complexity is linear in the number of examples. (Pazzani and Billsus, 1997). The *multinomial* and *Bernoulli* models have been proposed for modelling documents in text classification (Pazzani and Billsus, 2007; Manning *et al.*, 2008). The latter treats the document as a binary vector over terms whereas the former takes the number of occurrences into account.

Another *machine learning* approach is to compute a linear classifier $w \cdot x = b$, where x is a vector representation of a document, and w and b are the parameters to be learned. Many text classifiers fall into this category, including the naive Bayes and Rocchio methods (Manning *et al.*, 2008) (but not the kNN method). *Support vector machines* (Joachims, 1998) and the Widrow-Hoff algorithm (Widrow and Stearns, 1985) are other examples. The correct method to use varies with the application, and a comparative evaluation can be found in Yang and Liu (1999).

Decision trees (Quinlan, 1993) and *rule induction* (Cohen, 1995) are two approaches to generating an *explicit decision model*. Decision trees compute an easily explained representation but work best with a small number of features. Pazzani and Billsus (1997) show that this approach leads to poor results. Rule induction, on the other hand, have been applied with some success on e-mail classifiers. Koprinska *et al.* (2007) provide a recent survey of e-mail classification techniques, where *random forests* perform particularly well.

2.3.4 Summary

Most content-based recommendation approaches lie in the field of information retrieval. User feedback is translated to a profile which is then compared with available documents when deciding on a recommendation. Unlike collaborative filtering, these techniques do not make use of a user community but rather examine the content of previously encountered items. Though avoiding the new item problem, new users need to pass through a process of, implicit or explicit, elicitation; an often time consuming procedure. The literature is not clear on the distinction between content-based and knowledge-based recommenders. The difference most emphasized is that of automatic feature extraction (content-based) versus externally provided information (knowledge-based).

2.4 Knowledge-based Recommendation

Knowledge-Based (KB) recommender systems are developed to handle situations where there is little, or no, access to historical data, and user preferences and requirements are defined explicitly. Formulating requirements in this manner is not typical of collaborative and content-based approaches. Due to their non-reliance on past transactions, in the form of ratings, KB systems do not exhibit the other approaches' ramp-up problems. Recommendations are calculated independently for each user, either from data in the form of *similarities* between user requirements and item specifications or *recommendation rules*. Where the approaches discussed previously in this chapter focus on information filtering, KB systems take a more interactive approach. This has resulted in their characterization as *conversational systems* (Burke, 2000). However,

the knowledge acquisition phase in KB recommenders represents an effective bottleneck. This involves eliciting preferences from users and/or converting domain expert knowledge into a formal representation.

Knowledge-based RS can be roughly divided into *constraint-based* (Felfernig and Burke, 2008) and *case-based* (Burke, 2000) systems. Though both approaches rely on the user specifying his requirements through some interaction with the system, they differ in the way they compute the recommendations. Case-based recommenders make use of similarity measures where constraint-based systems enforce a set of, more or less strict, rules, as predefined in *recommender knowledge bases*.

Constraint-based recommendation is generally represented as a constraint satisfaction problem (Felfernig and Burke, 2008; Zanker *et al.*, 2010) that can be solved by a constraint solver or in the form of a *conjunctive query* (Jannach, 2006), executed and solved by a database engine. A *Constraint Satisfaction Problem* (CSP) (Jannach *et al.*, 2010; Tsang, 1993) can be described by a tuple (V, D, C) where V is a set of variables, D is a set of finite domains for these variables, and C is a set of constraints that describes the combinations of values the variables can simultaneously take. A solution to a CSP corresponds to a value assignment to each variable in V such that all constraints are satisfied. Defining the user's needs and preferences in the above constraints requires an elicitation process of varied complexity, depending on the method used for the application. Approaches to this include, user maintained profiles, session specific fill-out forms and conversational recommendation dialogues.

A *case-based* recommendation approach retrieves items using similarity measures that describe to which extent item properties match the user's requirements. We can similarly define a case-based recommendation problem by a tuple (V, D, R) where V is a set of variables, D is a set of finite domains for these variables, and R is a set of preferred assignments to the variables in V.

Earlier versions of case-based recommenders followed a *query-based* approach, in which users specified their requirements until a target item had been identified (Burke, 2002). The main drawback of such approaches is the amount of interaction needed, which motivated the development of *browsing-based* approaches, mostly based on the methodology of critiquing. *Critiquing* (Burke, 2000; Burke *et al.*, 1997) is the process by which users specify their change requests in the form of goals that have not been satisfied by the current item under consideration (either an initial item or the latest recommendation). Critiques can be specified in the level of attributes or abstract dimensions. Approaches exist that combine query-based and browsing-based item retrieval (Burke, 2002).

Variations to critiquing include *Compound critiquing*, where critiques can be related to more than one property and *Dynamic critiquing* (Reilly *et al.*, 2007) where generic descriptions of differences, or *patterns*, between recommended and candidate items are used for deriving the critiques. Dynamic critiques are calculated with use of associated rule mining (Agrawal and Srikant, 1994). Reilly *et al.* (2007) present an alteration of the recommendation procedure, where a *compatibility score* is computed representing the percentage of compound critiques that have already been selected and are consistent with the candidate item. This is then used to estimate the *quality* of a potential recommendation.

The approaches mentioned above show a deterioration of performance when many similar items exist in a small outcome space area.McCarthy *et al.* (2005) proposed to diversify critiquing by taking a *support* value into account. With this approach, items with low support of and overlap with already presented critiques are preferred.

Utility-based recommendation can be seen as a specific type of knowledge-based recommendation (Jannach *et al.*, 2010). It is often applied in combination with constraint (Felfernig *et al.*, 2006) or case-based recommendation (Reilly *et al.*, 2007).

The next section gives a detailed analysis of preference elicitation and decision making with utility functions, which encompasses also the task of recommendation.

2.5 Eliciting User Preferences

Preference Elicitation (PE) refers to the interaction of a system with one or more users, in order to learn a model of their preferences over some space of solutions or outcomes; essentially a ranking function defined over this space. Though there is work on learning other types of ranking functions, we emphasise the learning of *utility functions* since this allows us to compare different users' utilities without further assumptions, and compute expected evaluations of items. Before presenting the state-of-the-art in computational Preference Elicitation, we will provide the necessary decision theoretic background. This involves a presentation of important concepts regarding preferences, their representation, most relevantly through a utility function, and elicitation strategies aimed at the latter. Since Coarse Preferences can be understood as introducing an alternative decompositional model for preference functions, we will give special emphasis to such models.

2.5.1 Preferences

In order for us to present the basic notions of preferences and utility functions we first need to define a formulation of the decision problem (von Neumann and Morgenstern, 1953; Keeney and Raiffa, 1993; Braziunas, 2006). Assume a *decision maker*, or user, i, that needs to select an alternative, or action, $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of alternatives. The action's resulting outcome $o \in O$, out of the set of outcomes O, depends on the world state $\theta \in \Theta$, where Θ is the set of all possible world states. A consequence function $c : \mathcal{A} \times \Theta \rightarrow O$ maps each action and world state to a resulting outcome. The

user aims at selecting an action a^* leading to the best outcomes. If the world state θ is known, then an alternative maps directly to an outcome. In this case the problem translates directly to the selection of an optimal outcome o^*.

The user i maintains a preference function over O that provides a ranking, complete or partial, over the elements in O. This preference function might be modelled as a *value*, or *utility* function $u : O \to \mathbb{R}$ denoting the desirability of each outcome. The outcome space might itself be multidimensional and it is, in fact, this case that we will be more interested in.

Let us first consider preferences under a known world state θ, or *preferences under certainty*, as referred to in the literature (Keeney and Raiffa, 1993). Preferences over outcomes completely determine the optimal outcome(s), in the sense that a rational decision maker would choose one of the most preferred outcomes.

Assume a set O of outcomes. A weak preference $o^1 \succeq o^2$ of outcome o^1 over o^2 is a binary relation indicating that the user *weakly* prefers outcome o^1 to o^2, or, in other words, o^1 is at least as good as o^2. The *week preference relation* is typically expected to satisfy the following properties (von Neumann and Morgenstern, 1953), if the preference is to be considered *rational*:

$$\text{Comparability:} \forall o^1, o^2 \in O,\ o^1 \succeq o^2 \vee o^2 \succeq o^1 \tag{2.6}$$

$$\text{Transitivity:} \forall o^1, o^2, o^3 \in O,\ o^1 \succeq o^2 \wedge o^2 \succeq o^3 \Rightarrow o^1 \succeq o^3 \tag{2.7}$$

Based on the above, we can also define the binary relations of indifference $\sim$ and strict preference $\succ$. $o^1 \succ o^2$ indicates that o^1 is strictly preferred to o^2, and $o^1 \sim o^2$ that the

user is indifferent between o^1 and o^2. Formally:

$$o^1 \sim o^2 \Leftrightarrow o^1 \succeq o^2 \wedge o^2 \succeq o^1 \tag{2.8}$$

$$o^1 \succ o^2 \Leftrightarrow o^2 \not\succeq o^1 \tag{2.9}$$

Fishburn (1970) lists the following properties for a binary relation R on set O:

$$\textit{reflexive: if } oRo, \forall o \in O, \tag{2.10}$$

$$\textit{irreflexive: if } \neg oRo, \forall o \in O, \tag{2.11}$$

$$\textit{symmetric: if } o^1Ro^2 \Rightarrow o^2Ro^1,\ \forall o^1, o^2 \in O, \tag{2.12}$$

$$\textit{asymmetric: if } o^1Ro^2 \Rightarrow \neg o^2Ro^1,\ \forall o^1, o^2 \in O, \tag{2.13}$$

$$\textit{antisymmetric: if } o^1Ro^2 \wedge o^2Ro^1 \Rightarrow o^1 = o^2,\ \forall o^1, o^2 \in O, \tag{2.14}$$

$$\textit{transitive: if } o^1Ro^2 \wedge o^2Ro^3 \Rightarrow o^1Ro^3, \forall o^1, o^2, o^3 \in O, \tag{2.15}$$

$$\textit{negatively transtitive: if } \neg o^1Ro^2 \wedge \neg o^2Ro^3 \Rightarrow \neg o^1Ro^3,\ \forall o^1, o^2, o^3 \in O, \tag{2.16}$$

$$\textit{connected or complete: if } o^1Ro^2 \vee o^2Ro^1 (\text{possibly both}),\ \forall o^1, o^2 \in O, \tag{2.17}$$

$$\textit{weakly connected: if } o^1 \neq o^2 \Rightarrow o^1Ro^2 \vee o^2Ro^1,\ \forall o^1, o^2 \in O. \tag{2.18}$$

An asymmetric binary relation is irreflexive. An irreflexive and transitive binary relation is asymmetric. Also, a relation R is negatively transitive if and only if $\forall o^1, o^2, o^3 \in O$:

$$o^1Ro^2 \Rightarrow o^1Ro^3 \vee o^3Ro^2. \tag{2.19}$$

Weak preference is an asymmetric and negatively transitive relation, a *total preorder* or *weak order*, over the set of outcomes O. Strict preference is asymmetric and transitive, a *strict order* or weakly connected weak order, while indifference is reflexive, asymmetric and transitive, an *equivalence relation* (Fishburn, 1968).

Weak preferences can be represented compactly by a numerical function. An *ordinal value function* $u : O \to \mathbb{R}$ represents, or agrees, with the ordering $\succeq$ if and only if $\forall o^1, o^2 \in O$:

$$u(o^1) \geq u(o^2) \Leftrightarrow o^1 \succeq o^2 \tag{2.20}$$

A representation theorem gives necessary and sufficient conditions under which some qualitative relations can be represented by a numerical ranking, or scale. In the case of weak preferences, an agreeing ordinal value function can always be constructed if the outcome set O is finite or countably large. If O is uncountably large, then an agreeing ordinal value function exists if and only if O has a countable, order dense subset with respect to $\succeq$. $\Omega \subseteq O$ is order dense with respect to $\succeq$ if $\forall o^1, o^3 \in O$ such that $o^1 \succeq o^3$, there exists $o^2 \in O$ such that $o^1 \succeq o^2 \succeq o^3$ (Fishburn, 1999).

2.5.2 Preferences under uncertainty and utility functions

Ordinal value functions are unique up to *strictly increasing transformations*. They contain only preference ranking information, and any linear combination of scale values is without meaning. In order to make such comparisons, a necessity for preferences under uncertainty, the *expected utility* representation theorem (von Neumann and Morgenstern, 1953) is needed.

We begin giving the necessary definition for the von Neumann and Morgenstern (1953)

expected utility representation theorem with the concept of a lottery: A *simple lottery*, with outcome o_i realized with probability p_i, is denoted:

$$l = \langle p_1, o^1; p^2, o^2; ...; p_n, o^n \rangle. \tag{2.21}$$

Outcomes with zero probability are typically omitted from the representation while the often encountered case of two possible outcomes is abbreviated $\langle p, o^1; 1-p, o^2 \rangle \equiv \langle o^1, p, o^2 \rangle$. A *compound lottery* is a lottery with outcomes that are themselves simple lotteries: $l' = \langle p_1, l'_1; ...; p_k, l'_k \rangle$. It can be reduced to an equivalent simple lottery with the assumption that the user's preferences do not change between the two representations.

A rational decision maker is assumed to have a complete and transitive preference ranking $\succeq$ over the set of simple lotteries L. We can use a utility function $u : L \to \mathbb{R}$ to represent the users preferences over simple lotteries as long as the *continuity axiom* holds:

$$\textit{Continuity}\text{: for } p, q \in (0,1) \textit{ and } \forall l_1, l_2, l_3 \in L, \tag{2.22}$$

$$l_1 \succ l_2 \succ l_3 \Rightarrow \langle l_1, p, l_3 \rangle \succ l_2 \succ \langle l_1, q, l_3 \rangle. \tag{2.23}$$

The *independence axiom* that follows is necessary for the existence of a *linear* utility function:

$$\textit{Independence}\text{: for } p, q \in (0,1) \textit{ and } \forall l_1, l_2, l_3 \in L, \tag{2.24}$$

$$l_1 \succ l_2 \Rightarrow \langle l_1, p, l_3 \rangle \succ \langle l_2, p, l_3 \rangle. \tag{2.25}$$

The von Neumann and Morgenstern (1953) *expected utility* representation theorem states that,

Theorem 2.5.1 (Expected utility representation theorem)**.** If and only if the weak preference relation on simple lotteries is complete, transitive, and satisfies the continuity and independence axioms, then there is an *expected* or *linear* utility function $u : L \to \mathbb{R}$ which represents $\succeq$.

The utility function has the following properties:

$$u(l_1) \geq u(l_2) \Leftrightarrow l_1 \succeq l_2, \tag{2.26}$$

$$u(\langle l_1, p, l_2 \rangle) = p\, u(l_1) + (1-p)\, u(l_2),\ \forall l_1, l_2 \in L \text{ and } p \in [0,1]. \tag{2.27}$$

Preference relations over lotteries can be extended to outcomes by noting $l^o = \langle 1, x \rangle \equiv o$ and $u(o) = u(l^o)$. Due to the linearity of $u(\cdot)$, it can be shown through induction that:

$$u(l) = u(\langle p_1, o^1; p_2, o^2; ...; p_n, o^n \rangle) = \sum_{i=1}^{n} p_i\, u(o^i). \tag{2.28}$$

Eq. 2.28 allows for representing preferences over lotteries with a utility function over a finite set of outcomes.

2.5.3 Multiattribute preferences

Outcomes are seldom monolithic objects in practice. They exhibit internal structure that can usually be represented to a sufficient degree by a set of attributes. For each outcome o, each variable, or attribute, $x_j \in V = \{x_1, ..., x_m\}$ is assigned a value from its domain X_j, such that we have for the set of all outcomes $O \subseteq \mathcal{X} = \times_{j=1}^{m} X_j$, where $\mathcal{X}$ is the outcome space. O is best seen as the set of assignments to attributes that make sense or exist in the union of world states, and are not to be confused with the set of

feasible outcomes. The latter would be the set of outcomes that is accessible through the user's actions given the world state.

Given a variable set $A \subseteq V$, we define $X_A = \times_{j \in A} X_j$ to be the partial outcome space restricted to attributes in A, and $O_A \subseteq X_A$ as the set of partial outcomes available from that space. The vector o_A will similarly denote a *partial outcome* with assignments to the space X_A. When $A = V$, o_A is the *complete outcome* o (Braziunas, 2006).

If o_A and o_B are assignments to disjoint sets X_A and X_B, respectively ($X_A \cap X_B = \emptyset$), we denote their combination by $o_A o_B$. If $X_A \cup X_B = V$, we call $o_A o_B$ a *completion* of assignment o_A. We will use $\mathrm{Comp}(o_A)$ to denote the set of completions of o_A. For any outcome o^i, $x_j^{o^i} \in X_j$ denotes the assignment to variable x_j by o^i.

2.5.4 Preferential independence

The complete outcome space $\mathcal{X}$ represents an exponential number of possible assignments. Assessing the user's preferences directly over this space is usually infeasible. However, given sufficient preference structure, a function can be represented succinctly. When preferences over an attribute set A are independent of another set B, we say that A is preferentially independent of B. More specifically (Keeney and Raiffa, 1993):

Definition 2.5.1. Preferential Independence A set of variables A is *preferentially independent* of its complement $B = V - A$ iff, $\forall o_A^1, o_A^2 \in X_A$ and $\forall o_B^1, o_B^2 \in X_B$, we have

$$o_A^1 o_B^1 \succeq o_A^2 o_B^1 \iff o_A^1 o_B^2 \succeq o_A^2 o_B^2. \tag{2.29}$$

In other words, preferences over assignments to A, with all other variable assignments

fixed, are the same regardless of their specific assignment. If Eq. 2.29 holds, then we say that o_A^1 is preferred to o_A^2 *ceteris paribus*. Preferential independence is not a symmetric relation. Conditional preference independence is similarly defined (Fishburn, 1999):

Definition 2.5.2 (Conditional Preferential Independence)**.** Assume nonempty variable sets A, B and C that partition V. A is *conditionally preferentially independent* (CPI) of B given an assignment o_C to C iff, $\forall o_A^1, o_A^2 \in X_A$ and $\forall o_B^1, o_B^2 \in X_B$, we have

$$o_A^1 o_B^1 o_C \succeq o_A^2 o_B^1 o_C \iff o_A^1 o_B^2 o_C \succeq o_A^2 o_B^2 o_C. \tag{2.30}$$

In other words, A is preferentially independent of B given the specific assignment o_C to C. If this independence holds for all possible assignments to C, then A is *conditionally preferentially independent* of B given the set of variables C.

Ceteris paribus statements do not generally define a single preference ordering as we will see in section 2.5.8. The above definitions of preferential independence will be useful in our below definitions of Additive and General Additive Independence.

2.5.5 Utility independence

Preferential independence can be extended to preferences over lotteries, leading to the notion of *utility independence* (Keeney and Raiffa, 1993):

Definition 2.5.3. Utility Independence Let A_1, A_2 be disjoint sets of variables such that $V = A_1 \cup A_2$. A_1 is utility independent of A_2 if, for any assignments $o_{A_2}^1, o_{A_2}^2 \in X_{A_2}$, and for any pair of lotteries $l_1^{A_1}, l_2^{A_1}$ over X_{A_1}, we have that:

$$l_1^{A_1} o_{A_2}^1 \succeq l_2^{A_1} o_{A_2}^1 \iff l_1^{A_1} o_{A_2}^2 \succeq l_2^{A_1} o_{A_2}^2. \tag{2.31}$$

A generalization of preferential independence, utility independence is also not symmetric. The following theorem gives the form of a utility independent decomposition:

Theorem 2.5.2. Utility Independent Decomposition A set A_1 is *utility independent* of $A_2 = V - A_1$ for u, if and only if it has the form:

$$u(V) = f(A_2) + g(A_2)y(A_1). \tag{2.32}$$

It is obvious from the above, that the final form of the decomposed function will depend on the specific utility independences assumed.

2.5.6 Additive independence

An additively independent decomposition allows for a utility function over disjoint sets of variables; including, of course, the case where each set is a singleton. We give the definition and decomposition theorem below as adapted from Koller and Friedman (2009):

Definition 2.5.4. Additive Independence Let $A_1, ..., A_k$ be disjoint sets of variables such that $V = \cup_i A_i$. $A_1, ..., A_k$ are *additively independent* (AI) for an underlying utility function u if, for any two probability distributions Pr_1 and Pr_2 over $\mathcal{X} = \text{Dom}(V)$ that have the same marginals on each of the sets of variables A_i, u has the same expected value under Pr_1 and Pr_2.

Additive independence is strictly stronger than utility independence: Given two additive independent subsets $A_1 \cup A_2 = V$, we have that A_1 is utility independent of A_2 and *vice versa*. Additive independence is equivalent to decomposing a utility function u into the sum of subutilities u_i over A_i:

Theorem 2.5.3. Additive Independent Utility Functions Let $A_1,...,A_k$ be disjoint sets of variables such that $V = \cup_i A_i$. $A_1,...,A_k$ are AI for an underlying utility function u if and only if u can be written as:

$$u(V) = \sum_{i=1}^{k} u_i(A_i). \tag{2.33}$$

Jon considers the item in front of him carefully. He particularly likes the colour black, and assigns that a partial utility of 0.3. He likes scarves and values that assignment to the parameter *type of clothing* at 0.24. The *intended gender* never has an impact on Jon, who assigns that attribute a partial utility of 0. He is content with prices being listed in *pounds*, partial utility of 0.02, but finds the price a bit steep, -0.15, the *brand* indifferent, -0.04, the lack of a *discount* troublesome, -0.11, and the fact that it is out of *stock* prohibitive, -0.25. Jon therefore values this item at 0.01, which is below his threshold of 0.6, and therefore rejects the item.

2.5.7 Generalised additive independence

We are now ready to give the definition for GAI-decomposition (Boutilier *et al.*, 2001):

Definition 2.5.5. General Additive Independence A Let $A_1,...,A_k$ be sets of *not necessarily disjoint* variables such that $V = \cup_i A_i$. $A_1,...,A_k$ are *generalized additive independent* (GAI) for an underlying utility function u if, for any two probability distributions Pr_1 and Pr_2 over $\mathcal{X} = \mathrm{Dom}(V)$ that have the same marginals on each of the sets of variables A_i, u has the same expected value under Pr_1 and Pr_2.

In other words, the expected value of u is not affected by correlations between the A_i.

It depends only on the marginal distributions over each set A_i. The definition above is equivalent to the following (Bacchus and Grove, 1995; Gonzales and Perny, 2004):

Theorem 2.5.4. General Additive Independent Utility Functions Let $A_1,...,A_k$ be sets of *not necessarily disjoint* variables such that $V = \cup_i A_i$. $A_1,...,A_k$ are GAI for an underlying utility function u if and only if u can be written as:

$$u(V) = \sum_{i=1}^{k} u_i(A_i). \tag{2.34}$$

The *degree* of a GAI-decomposition is the maximum arity of sub-utilities and its *cardinality* is the number k of sub-utilities.

Jon always considers the *colour* along with the *type* of clothing. He really likes black scarves and assigns to that a utility of 0.75. He is never interested in the *intended gender*, to which he assigns a partial utility of 0. Though the *currency* is in pounds, the lack of a *discount* is particularly troublesome, given that the price is too steep, and he assigns to the combination of those 3 parameters a partial utility of -0.40. Jon would never consider waiting for a scarf of *brand AB* to *restock* and therefore assigns the assignment to this variable subset a partial utility of -0.30. Jon therefore attributes a utility of 0.05 to this item, which he promptly swipes left on.

2.5.8 Conditional preference networks

Conditional *ceteris paribus*[1] preference statements capture conditional preferences; the preference over an attribute assignment or combination of assignments is dependent upon those of one or more other attributes. Doyle *et al.* (1991) introduced the *logic of relative desire* to treat preferences under a *ceteris paribus* assumption. His work bears

[1] all other things being equal (Latin)

resemblance to von Wright (1963)'s logic of preferences, supporting more complicated inferences, however. Perhaps the first and most important attempt at a compact representation of *ceteris paribus* statements, exploiting preferential independence, lies in the work of Boutilier *et al.* (1997) (see also Boutilier *et al.* (2004a)) and their *Conditional Preference Networks* (CP-nets):

Definition 2.5.6 (Conditional Preference Networks)**.** A *CP-net* over variables $V = \{x_1,...,x_m\}$ is a directed graph G over $x_1,...,x_m$ whose nodes are annotated with conditional preference tables $CPT(x_i)$ for each $x_i \in V$. Each conditional preference table $CPT(x_i)$ associates a total order $\succ^i_{o_U}$ with each instantiation o_U of x_i's parents $Pa(x_i) = U$.

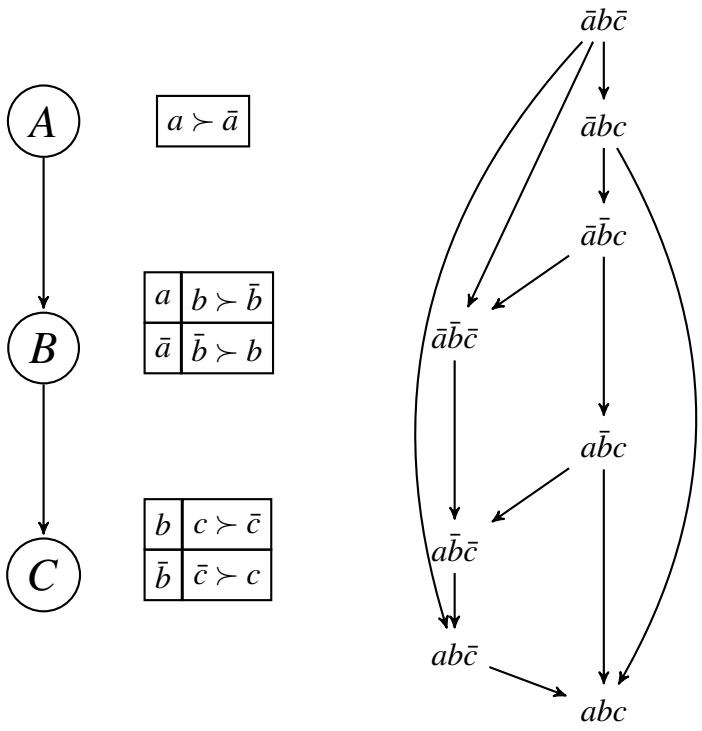


Figure 2.1: A simple CP-net (left) and the induced preference graph (right) as adapted from Boutilier *et al.* (2004a).

Each CP-net can be translated to a corresponding preference graph by use of the CPTs that accompany each node. *Preference graphs* dictate the dominance relations among outcomes. A CP-net's CPTs impose a set of preference constraints and a preference

ranking that does not violate these is said to satisfy the network. A CP-net is *satisfiable* if there is some preference ranking that satisfies it. Every acyclic CP-net is satisfiable, and CP-nets can be satisfied by more than one preference ranking.

Fig. 2.1 shows, on the left, a simple binary variable CP-net. In this example, preferences over assignments to B are dependent on the value of A, and preferences over assignments to C are dependent on the value of B. Moreover, when determining preferences over outcomes, the value of A is more important, has *higher priority*, than that of B, with the latter being more important than the value of C. The preference graph on the right half of Fig. 2.1 shows the dominance relations among outcomes; the head of an arc dominates its tail.

CP-nets can be used to perform comparisons between full outcomes, partial outcome optimization, and outcome ordering. Partial outcome optimization is the task of determining the optimal completion to a partial assignment. The latter two can be accomplished in polynomial time, however, dominance testing is PSPACE-complete, with polynomial time algorithms existing for the special case of tree and polytree structured networks. Chevaleyre *et al.* (2011)'s work consists a detailed theoretical analysis of the problem of learning a CP-Net. Though no general method of learning (beyond brute-force search) is given, they provide learnability results for the cases of passive and active learning.

UCP-networks Boutilier *et al.* (2001) are a directed graphical representation of utility functions that combines CP-nets and GAI. The utility function is decomposed by the network into a number of additive factors. As in CP-nets, a node is conditionally dependent on the set of tails whose arcs end in it.

Definition 2.5.7. UCP-Networks Let $u(x_1, ..., x_m)$ be a utility function with induced preference relation $\succeq$. A *UCP-network for u* is a Directed Acyclic Graph (DAG) G over

$x_1, ..., x_m$ and a quantification (i.e. a set of factors $f_i(x_i, U_i)$ where U_i are the parents of x_i) s.t:

$u(x_1, ..., x_m) = \sum_i f_i(x_i, U_i)$

The DAG G is a valid CP-network for $\succeq$; i.e., $\succeq$ satisfies conditional preference independence $CPI(x_i, U_i, Z_i)$ for each x_i, where $Z_i = V - (U_i \cup \{x_i\})$.

Jon values *black* clothes for 0.5 partial utility. Conditioned on the item being black, Jon is happy that the item is a *scarf*, for an additional 0.25 utility. He does not consider the item's *intended gender*. He adds 0.02 for the price being in *pounds*, but the *price* tag is too great for it to be in pounds, -0.32, and conditioned on this, the lack of a discount adds insult to injury for another -0.10. Jon is indifferent towards the *brand AB*, -0.04, and conditioned on the brand, he finds that the item is *out of stock* unacceptable, -0.36. Jon assigns a utility of 0.05 to the scarf, which is below his threshold of 0.6, and swipes left on the item, rejecting it.

CP-nets have been extended to deal with hard (Boutilier *et al.*, 2004b) and soft constraints (Prestwich *et al.*, 2005). TCP-nets (Brafman and Domshlak, 2002) are a generalization of CP-nets, introducing conditional importance relations among variables. Lastly, Wilson (2004) proposes a logic of conditional preferences that generalizes CP-nets, allowing for stronger conditional preference statements.

2.5.9 Issues with variable-based decompositional models

AI, GAI, and CP-nets are examples of models of preferential indifference defined over the set of variables describing the item space. If we assume utility queries, then the time used for performing active learning with such models will be linear in the space of items considered, provided assignments to the variable space uniquely identify each item. If we were to consider pair-wise comparison or list queries, then the time required

will increase exponentially with the number of items considered. Though this is the accepted norm, it is a consequence of these models not having an explicit representation of the preferential indifference of users for different combinations of assignments to the variables.

Explicitly representing the preferential indifference relation (see Eq. 2.8) would allow for a principled way of 'ignoring' some of the items in our available set of items, since these will be redundant when making a decision over the item space, effectively defining *categories* of items. Active learning is just one such decision problem.

This reduction in the number of actions, or items, that need to be considered indicates a hierarchical decision making model; we first establish which items the user is indifferent between, then proceed to select one of these. Though there is no criterion differentiating the items in this subset when only a single user's preferences are considered, we will see in later chapters how this can enable the use of further criteria.

Another concern relating to the absense of an explicit representation of preferential indifference in variable-based decompositional models, is that even if there are subsets of the item space that are equally preferred by a user, it is improbable that this will be represented in our current belief of the user's preference model. Specifically, we cannot generally expect for items which the user equally prefers to have the same expected utility before our belief over their model has converged to the true model. Preferential indifference therefore represents an issue of both computational time and accuracy.

2.5.10 Decision making criteria

Complete knowledge of a decision maker's utility function makes recommending an optimal outcome, more or less, trivial. It will usually be the case, though, that preference elicitation will not be exhaustive, and decisions will be made with only

partial knowledge of the user's preferences. A common approach is to maintain a space of possible utility functions, and use this to make a decision. Below we present some approaches in the literature for handling uncertainty over the user's utility function.

Strict uncertainty

Situations characterized by *strict uncertainty* present us with a viable set of utility function U, but not with a probability distribution indicating the likelihood of each of them being the *true* utility function u. This prevents us from using an expectation over u in order to make or recommend a decision a.

Criteria for selecting optimal outcomes under strict uncertainty optimise some worst or best case, or a compromise between worst and best cases. Examples of such criteria are the *Maximin return* (Wald, 1950) with examples of use in Boutilier *et al.* (2003a) and Salo and Hämäläinen (2004); *Hurwicz's optimism-pessimism index* (French, 1986) which generalizes maximin, maximax and the *central values* criterion (Salo and Hämäläinen, 2001); and *Minimax regret* (Savage, 1951) with example applications in the works of Boutilier *et al.* (2001), Boutilier *et al.* (2003a), Boutilier *et al.* (2003b), Wang and Boutilier (2003), Boutilier *et al.* (2004c), Boutilier *et al.* (2005), Patrascu *et al.* (2005). Minimax regret, in particular, fails to satisfy the axiom of *independence of irrelevant alternatives*, or *binary independence* axiom, that states that the ranking between two alternatives must be independent of other alternatives. However, it has been shown that human decision makers rarely adhere to this principle (Tversky and Kahneman, 1981).

Expected utility

Uncertainty over a decision maker's utility function can be modelled with a distribution over utility functions. Decisions are then made based on the expectation over each action's or outcome's utility. Boutilier (2003) refers to this expectation as *Expected* Expected Utility (EEU), indicating the fact that decisions are made over expectations of lotteries' expected outcomes. He points out in this work, that decisions under EEU are sensitive to the precise representation of the utility function: EEU is sensitive to positive affine transformations, while it is not always the case that different utility functions can be meaningfully compared and combined.

The expected utility of a decision or action a, $EU(a,u)$, is equivalent to the utility of the lottery induced by the action as is given by Eq. 2.28. The expected utility of an action a, given a density b over the set of utility functions $U \in \mathbb{R}^{|O|}$. is:

$$EU(a,b) = \int EU(a,u)b(u)\mathrm{d}u. \tag{2.35}$$

The optimal decision in such a belief state b is then $\mathrm{argmax}_a EU(a,b)$. Unfortunately, this decision rule is not invariant to utility transformations, and the question arises of which utility function to use.

Boutilier (2003) shows how one can assure commensurability of candidate utility functions, by assuming the existence of a known best and worst outcome $o_\top$ and $o_\bot$, respectively, and demanding $o_\top \succ o_\bot$. Utility functions adhering to this restriction are called *extremum equivalent* and those defined over the same $o_\top, o_\bot$ are essentially put on the same scale. These outcomes must not vary and their utilities must remain constant (values $u(o_\top) = 1$ and $u(o_\bot) = 0$ is a common option). Issues arise when these outcomes are selected from a reachable, *local*, subset of a larger, *global*, outcome space, and additional assumptions are needed when the latter is the case.

2.5.11 Preference elicitation

Preference Elicitation (PE) can be seen as the interviewing of the user by the system (even when that is done implicitly). When modelling an elicitation procedure, we need to decide over the questions, or *queries*, asked. What type of queries are we going to use, and which specific query should we ask in each circumstance. A good overview can be found in Braziunas and Boutilier (2009). This paragraph presents some commonly used query types, and follows with a range of decision criteria for selecting specific queries.

Query types

Queries are the means by which a system elicits information from the decision maker or user. A query is usually represented by the set of all possible responses Q, and the user presented with it is tasked with providing a response $q \in Q$.

The queries below are stated as *explicit* questions to a user. Of course, queries can be *implicit*, in that the user's behaviour when faced with options Q in the environment can be translated to the selection of a response q. These can range from clicking on a button or link, to spending an increased amount of time in an area or page. Alternatively, users might be asked to critique a presented behaviour. Related fields include *inverse reinforcement learning* (Ng and Russell, 2000; Chajewska *et al.*, 2001), and *revealed preferences* in Economics (Mas-Colell *et al.*, 1995). Mathematically, there is no distinction in the handling of explicit and implicit queries, as long as we can assume that our queries are consistently either one or the other. If the user is conciously aware of being asked a question (i.e. explicit queries) in some cases but not others, then we might want to consider using different response models for each

case and/or utility functions. The experiments in this thesis present queries to users in a consistent manner.

Another distinction between query applications is that of *local* and *global* queries. Global queries present the user with a choice between complete outcomes, regardless of any structure they might follow. In contrast, local queries involve comparisons between partial outcomes, as described in section 2.5.3. Though the queries that follow are presented over complete outcomes, the generalization is straightforward. An issue arising is the comparison between partial outcomes over different partial outcome spaces. This is something that needs to be addressed in any elicitation of decomposed utility functions. A relevant issue, one of interest in the area of Bounded Rationality (Simon, 1972; Tversky and Kahneman, 1981; Kahneman and Tversky, 1984; Gigerenzer and Selten, 2003), is the ease with which people can compare outcomes with more than a few variables. See for example Green and Srinivasan (1978), and Miller (1956).

Query types prevalent in the PE literature include gamble comparisons, where the user is asked to select between an outcome and a lottery, and pairwise comparisons, where the selection is between two outcomes. Other query types encountered in the literature include *membership* and equivalence queries (Boutilier *et al.*, 2010; Koriche and Zanuttini, 2010; Lahaie and Parkes, 2004), and *value queries* (Zinkevich *et al.*, 2003; Shawe-Taylor and Singer, 2004).

Standard Gamble Comparison (SGC) queries present the user with a choice between a certain outcome o and a lottery $\lambda = \langle o_\top, l, o_\bot \rangle$, that results in the best possible outcome $o_\top$ with a probability of l, and the worst outcome $o_\bot$, otherwise (Keeney and Raiffa, 1993). We denote a standard gamble query as $Q_{ol} = \{o, \langle o_\top, l, o_\bot \rangle\}$. If we assume normalization $u(o_\bot) = 0$ and $u(o_\top) = 1$ then $u(\lambda) = l$. The query is usually posed as whether the user *prefers the certain outcome to the gamble*. Resolving indifference is

not straightforward, but responses *yes* and *no* are usually translated to $o \succeq \lambda$ and $o \prec \lambda$, respectively.

A more general query, where the lottery outcomes are not necessarily extremes, is simply called a *gamble query*. Farquhar (1984) described gamble queries for every combination of outcomes and lotteries. One variation is the *probability equivalence query* (or *direct utility query*) where the user is asked to provide the probability l (equivalent to a utility value with the above assumptions) at which she would be indifferent between the two options (see e.g. Keeney and Raiffa (1993), and Gonzales and Perny (2004)). As noted however in Braziunas (2006), comparing an outcome and a lottery might be psychologically easier than directly assigning a utility. Lastly, one can ask for *bounds* over a utility value, as in Boutilier *et al.* (2003b) and Regan and Boutilier (2009).

For recent applications of SGC queries see Chajewska and Koller (2000); Boutilier *et al.* (2001); Boutilier (2002); Wang and Boutilier (2003); Boutilier *et al.* (2003b, 2005); Braziunas and Boutilier (2005), among others.

A *pairwise comparison query* (or *order query*) presents the user with a pair of outcomes and asks of her to indicate their preference relation. It can be denoted as $Q_{o_1 o_2} = \{o_1 \succ o_2, o_1 \prec o_2, o_1 \sim o_2\}$, with one response for each possible relation type.

Pairwise comparison queries are known to require low cognitive load of users (Conitzer, 2007); something which is expected to reduce noise in the elicitation process. If the two outcomes differ greatly in utility, the user can answer with ease, otherwise, confusion comes into play.

The Bradley-Terry model of confusion (Bradley and Terry, 1952) can be used to model such noise for strict preferences while extensions exist for the indifference preference. Unfortunately, pairwise comparison queries are not as informative as value queries,

which directly reveal information on a solution, though they are central in conjoint analysis and multiattribute utility theory (e.g. in Braziunas (2006)). For a recent application see for example Guo and Sanner (2010).

When indifference is not an option, then pairwise comparison queries can be seen as a special case of *choice queries* (Viappiani and Boutilier, 2010), where the user is asked to select one item o^* from a presented set O. This can then be modelled as the user having responded with $o^* \succ o$ to each of the queries $Q_{o^* o} = \{o^* \succ o, o^* \prec o\}\ \forall o \in O - \{o^*\}$. Choice queries can be presented to the user by making a *set recommendation* from which the user can select one alternative.

Value queries, involve asking the user to rate a presented item given a rating scale or range. Most online rating systems, including those considered in Collaborative Filtering, can be modelled in this way.

Query selection criteria

Having selected what query types are applicable, the question arises of how to select individual queries to present the user with. Below we outline three general approaches, and point to some recent applications.

The *Minimax regret* criterion can be used to consecutively bound the distance from the optimum as elicitation proceeds. Methods employing this criterion have been applied in auctions (Wang and Boutilier, 2003), combinatorial auctions (Boutilier *et al.*, 2004b), constrained configuration problems (Boutilier *et al.*, 2003b, 2005), and autonomic computing (Boutilier *et al.*, 2003a; Patrascu *et al.*, 2005).

When the minimax regret decision criterion is employed, there is no access to probabilistic information over the utility function. Queries aim at reducing the space of

possible utility functions and, in so doing, minimize the minimax regret of decisions. Queries can continue, e.g., until possible regret reaches an acceptable level, or elicitation becomes too expensive. Their selection involves anticipating the decision maker's response to each query, and is done according to some criterion such as the query with the best worst-case response, or with the maximum average or expected improvement (Wang and Boutilier, 2003).

Boutilier *et al.* (2003b) address the problem of selecting the best out of a set of configurations as encoded by hard constraints. Preferences are modelled through a GAI utility function, where sub-utilities are imprecisely specified by bounds. Boutilier *et al.* (2005) compare different GAI elicitation strategies, using *bound* queries; questions over the relative place of the true value compared to that in the query. Boutilier *et al.* (2006) expand on the the two papers above, while Patrascu *et al.* (2005) examine the elicitation of monotonically non-decreasing utility functions over single-variable domains.

Wang and Boutilier (2003) introduce and provide empirical results for different myopic elicitation strategies, considering an AI function decomposition. Braziunas and Boutilier (2007) review key semantic issues and propose a variety of query classes ans strategies, while Viappiani and Boutilier (2009a,b) address the problem of optimal recommendation sets, introducing the criterion of *set-wise minimax regret*. Lastly, Braziunas and Boutilier (2010) test the effectiveness of regret-based elicitation, as well as acceptance and comprehension, in an empirical user study, showing promising results.

The literature presents many examples of decisions that directly aim at reducing the uncertainty over the utility function. The set U of possible utility functions is often represented as a convex polytope over the function parameters. Queries bisect this polytope by addition of a linear constraint, and are selected through use of various heuristics. Iyengar *et al.* (2001) considers the resulting polytopes' volume, while Ghosh and Kalagnanam (2003), and Toubia *et al.* (2004) also consider their shape. The

following approaches aim to minimize the number of queries, but fail to account for the tradeoff between decision quality and elicitation costs. Iyengar *et al.* (2001) present the Q-Eval algorithm, a process of pairwise comparison queries for the addition of linear constraints on the weight space. Queries are selected so as to be the closest to bisecting the space of weights. The centreof the space is defined as the prime analytic center; the point maximizing the sum of log distances to the hyperplanes defining the region. Ghosh and Kalagnanam (2003) employ a similar algorithm where the center of the weight region is determined by a sampling procedure employing Markovian random-walk. They then ask the query whose corresponding hyperplane is orthogonal to the longest line segment contained in the weight region. Holloway and White (2003) model sequential elicitation of additive utility functions as a Partially Observable Markov Decision Process, with uncertainty over functions represented by linear constraints on the weights. Abbas (2004) presents an elicitation algorithm for single-variable utility functions with probabilistic uncertainty. Queries are selected myopically, so as to minimize the entropy of the distribution over the function. Toubia *et al.* (2003), and Toubia *et al.* (2004)'s are the first works considering adaptive elicitation through *conjoint analysis*. Conjoint analysis is a set of techniques for measuring consumer trade-offs among multi-attribute preferences and was first introduced by Green and Rao (1971).

Use of the probabilistic uncertainty over a utility function, allows for balancing possible gain of information from asking a query, with any related costs or decision rewards. Each response to a query leads to a different belief state over the user's utility function. Allowing for sequences of queries transforms the problem of elicitation into a sequential decision process. Of course, using this procedure makes for significantly more difficult computations. We start by defining the (myopic) expected value of information before proceeding to describe the sequential equivalent.

Myopic EVOI Assume a set of queries $Q = \{q_i : i = \{1,...,n\}\}$, and a set of possible

responses for each query $q_i \in Q$, $R_i = \{r_j^i : j = \{1,...,m\}\}$. The *Expected Value Of Information* (EVOI) is commonly used to determine which query to make (Chajewska *et al.*, 2000) and is a measure of the utility of asking a query given a user's utility function u. The user's responses are often allowed to be noisy, and her responses modelled by a *probabilistic response model*:

$$Pr(r_j^i \,|\, q_i, b) = \int_{u \in U} Pr(r_j^i \,|\, q_i, u) b(u) \mathrm{d}u, \tag{2.36}$$

where b is a density over the set of possible utility functions U. Queries, q_i, are often assigned a cost c_i, which is aimed to model computational expenses, cognitive burden on the user, etc. A Bayesian formulation of the elicitation process takes these costs into account, comparing them with any gain in utility. Denote $EU(o,b)$ as the expected utility of outcome o given the distribution b over u. The *maximum expected utility* of belief state b is then $MEU(b) = \max_{o \in O} EU(o,b)$. A response r to a query q updates our belief b according to the Baye's rule:

$$b^r(u) = b(u|r) = \frac{Pr(r|u)b(u)}{Pr(r|b)} \tag{2.37}$$

Calculating the value of a query requires weighting according to the probability of each response. The *Expected Posterior Utility* (EPU) of a query q_i is:

$$EPU(q_i, b) = \sum_{r \in R^i} Pr(r|q_i, b) MEU(b^r), \tag{2.38}$$

with which we can compute the EVOI of a query q_i from:

$$EVOI(q_i, b) = EPU(q_i, b) - MEU(b). \tag{2.39}$$

A *myopically*, or greedy, optimal strategy always selects the query which maximises

Eq. 2.39 minus any query costs. A sequentially optimal strategy would take any future queries into account and plan accordingly. Chajewska *et al.* (2000) is arguably the first paper to consider adaptive Bayesian elicitation. Example applications of Myopic EVOI can be found in Chajewska and Koller (2000) and Braziunas and Boutilier (2005). The latter (see also Braziunas and Boutilier (2006) for a summary) uses a GAI structure to ease EVOI computations. Users answer local queries, that equate to gamble queries, updating the belief over local utilities. A few global queries are asked in the end, and the results are fed to an algorithmic procedure that takes advantage of the GAI structure. Viappiani and Boutilier (2010) consider the EVOI of *choice* queries, where the user has to select an item out of a recommended set. Different user response models were considered, and optimality was shown for noiseless and constant-noise models.

One way of extending myopic EVOI is to perform a multistage lookahead, though such an approach might be of limited benefit, due to the requirement of online computation. Boutilier (2002) proposes instead to model PE as a Partially Observable Markov Decision Process (POMDP), in order to take the value of possible future queries into account when deciding on the current query. The state space of this POMDP is the set of utility functions U, while actions are either queries or single-item recommendations. There is no transition of states, since u is static, and a belief over U is updated according to the user's responses to the queries. The reward function for recommendations is equal to their expected utility, while a cost is assumed for queries. Solving the preference elicitation POMDP is computationally hard; the state space is continuous and multi-dimensional and is approximated by a uniform or truncated Gaussian mixture model. The POMDP is solved by asynchronous value iteration. Doshi and Roy (2008) explain how preference elicitation POMDPs are often *permutable*; they are symmetric with regard to the transition, reward, and observation functions. Given this assumption, they propose an algorithm that allows for an exponentially reduced number of belief points for approximation, for the same solution quality. Even though a PE POMDP

can be expected to only be approximately permutable, their experiments show that computations are significantly sped up.

Eliciting additive independent utility models

Considering our two case studies, Guo and Sanner (2010) represents the current state-of-the-art in the online elicitation of additive independent utility functions when using pairwise comparison queries, and without making use of user descriptors. Their work makes use of the TrueSkill algorithm (Herbrich *et al.*, 2007). Essentially an instantiation of the *Belief Propagation* algorithm (Murphy, 2012) combined with EVOI computations, the procedure accomplishes significant speed-ups compared to previous work. It is also easy to adapt for use with value queries. Figure 2.2 shows the factor graph over which the TrueSkill algorithm runs in order to compute an update to the user model after he/she responds to a pair-wise comparison query. By instead running a Belief Propagation algorithm over the graph in Figure 2.3 we can compute the update to our model given a value query.

In both cases, we are factoring our belief over the user's preferences into a set of Conditional Gaussians (CG); one for each parameter describing the items. Conditioned on a discrete assignment to the related parameter, each CG represents our belief over the users' partial utility from the specific assignment to that parameter as a Gaussian. The sum of each such Gaussian then represents our expectations over the user's evaluation of that item. Figure 2.2 is constructed by the combination of the representations for two compared items, and the factor comparing the user's evaluation over them. Figure 2.3 is comprised of the factors representing the evaluated item and the factor for that evaluation.

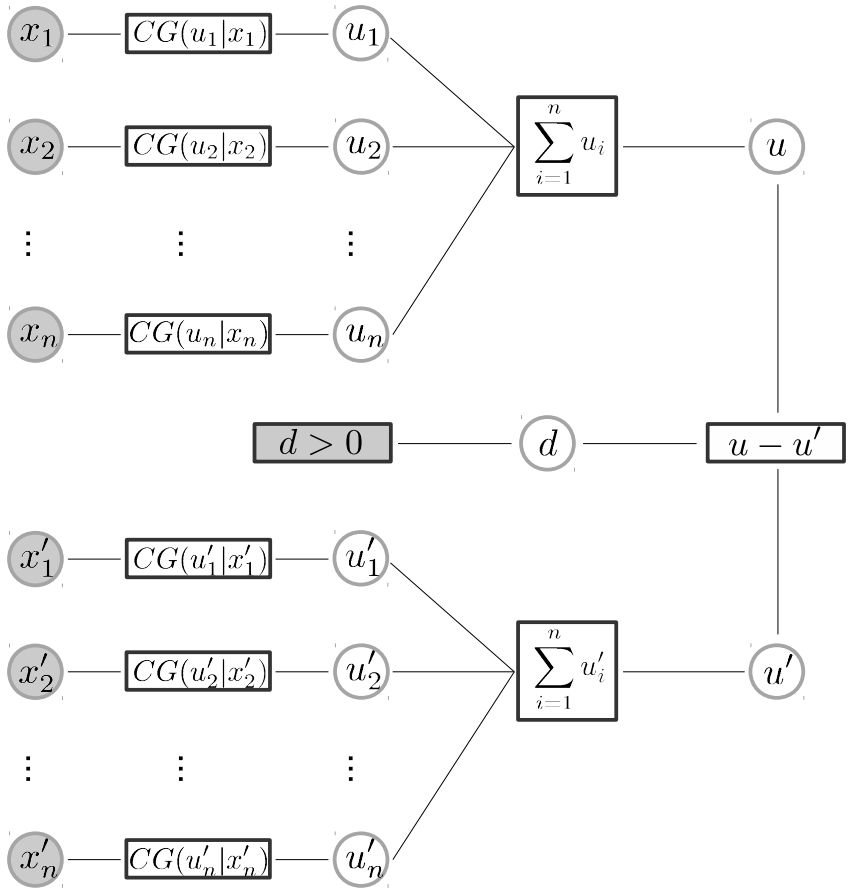


Figure 2.2: Factor graph for updating an additive model of user preferences for pairwise comparison queries. Known quantities are shaded grey.

2.5.12 Gaussian Processes for preference elicitation

Gaussian Processes (Rasmussen, 2006) have also recently been used for preference elicitation. State of the art approaches either assume a predefined constant set of items (Abbasnejad *et al.*, 2013; Vanchinathan *et al.*, 2014), or incorporate user vector descriptions in their model (Guo *et al.*, 2010), which we do not have access to in either of our data sets. A notable exception is the work by Houlsby *et al.* (2012). Nguyen *et al.* (2014) present an interesting variation that incorporates contextual information, such as the time of day. Similarly, Bohnert *et al.* (2009) make use of spatial information in

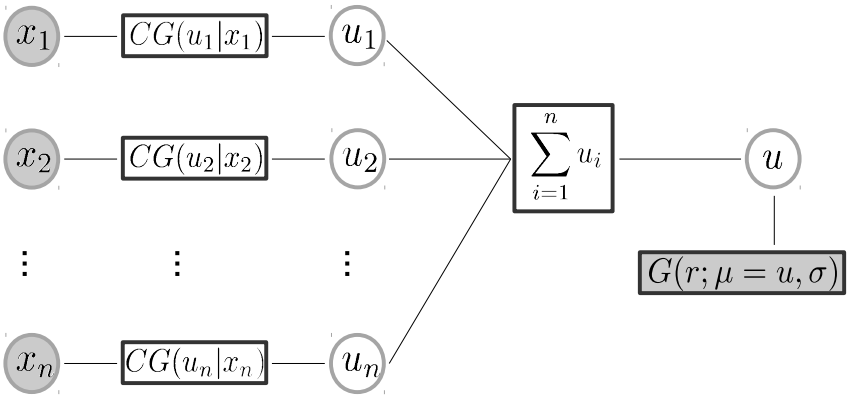


Figure 2.3: Factor graph for updating an additive model of user preferences for value queries. Known quantities are shaded grey.

order to inform their recommendations. Gaussian Processes are kernel-based methods and are updated offline.

2.6 Preferential Indifference and Categorisation

Categorizing individual items into groups is well documented in the literature (see e.g. Wilson and Keil (1999), and Rosch and Lloyd (1978)). Categorization refers to the process by which individual entities are treated as equivalent, and is one of the most fundamental and pervasive cognitive activities (Wilson and Keil, 1999). Though there is a lot of of work on *similarity-based* categorization, this approach has been criticised as unobjective (Goodman, 1972; Medin, 1989).[2] An alternative approach is *theory-based* categorization where *concepts*, defined by features and properties, act as mental representations of categories (Komatsu, 1992). *Coarse thinking* (Mullainathan, 2002, 2008) models decision makers that predict outcomes by using the probability distribution of the most likely category, rather than employing a Bayesian model.

[2]See Decock and Douven (2011) however for a recent counter-position.

The notion of *concept* (Angluin, 1988; Haussler, 1989) is relevant to our approach towards coarse preferences. Concepts are essentially subsets of the outcome space that satisfy some specified formulas. The problem of *concept learning* involves the identification of a concept given a sequence of labelled samples or an interactive querying procedure. Recently, Boutilier *et al.* (2009a) presented an online feature elicitation procedure, where uncertainty over concepts was reduced enough to make an optimal decision. They assumed a known utility function over concepts, which they refer to as *subjective features*, and outcomes described by catalogue features. The system constructs a mixed integer program from the answers to a series of membership queries, with use of the *minimax regret* criterion. Boutilier *et al.* (2010) [3] expand on this by introducing uncertainty over the utility function. Though we are interested in eliciting utility functions over a coarser representation of the outcome domain, which a set of concepts could be, we do not assume that the user is able to label these subspaces.

Our approach to coarse preferences is related to Bjorndahl *et al.* (2013)'s Language-based games. A generalization of psychological games (Geanakoplos *et al.*, 1989), their approach can capture dependent preferences (Kőszegi and Rabin, 2006) as well as coarse beliefs (Mullainathan, 2002) or categorical thinking. Bjorndahl *et al.* (2013)'s central concept relies on *utility* being defined, not on outcomes, but on situations; a collection of statements about, or *descriptions* of, the game. The set of all admissible descriptions comprises the *underlying language* of the game, on which the utility of a user is completely dependent.

Another problem that relates to coarse preferences is that of preferences over *value ranges*. Öztürk *et al.* (2011) model preferences over n-point *intervals*; intervals that allow for $n-2$ intermediate points between the two edges. Interval's are allowed to partially overlap, their relative position defining the strength of any preference relation, with a complete disjunction providing the strongest relation. Farfel and

[3] See also Boutilier *et al.* (2009b) for a preliminary version.

Conitzer (2011) describe the problem of eliciting truthful preferences over every users' single most preferred value range, when the there is shared knowledge of consequent aggregation. Users are modelled with *peaked* preferences, where points closer to the respective interval are preferred. Strugeon (2014) builds on a problem of group decision making, to describe how indifference over value ranges can ease the elicitation process. Unfortunately, the elicitation is trivial, assuming that users can explicitly give their complete preference structure to the system, the focus of the paper being on aggregation through self-reported ranges.

Lastly, Crès (2001) presents an analysis on the benefits of coarse preferences in individual preference aggregation for majority voting. Coarseness, here is defined as a partition of the single-dimensional outcome space into a preset number of categories. Users are indifferent between outcomes in the same category.

2.7 Multi-Agent Resource Allocation

Chapter 5 addresses our second case study of a *movie viewing recommender system*, solving a multi-agent coordination problem through recommending sets of items. Though the problem assumes non-strategic agents (in fact, the problem description is such that an agent would behave in the same manner regardless of whether or not they were strategic), we acknowledge that the deployment of such multi-agent recommender systems could benefit from tools in the Mechanism Design literature (Nisan and Ronen, 2001), particularly in the context of Multi-Agent Resource Allocation. We will therefore provide a brief overview of this area here, and reference this material when describing the mechanism in Chapter 5.

Multi-Agent Resource Allocation (MARA) is "*the process of distributing a number of*

items amongst a number of agents" (Chevaleyre *et al.*, 2006). Considering Recommender Systems, these *resources* would be the *items* we present as recommendations, while the *agents* would be addressed as *users*. Resources can be numerically *continuous* (e.g. gallons of water) or *discrete* (e.g. a movie, or house), and can either be *divisble* (e.g. one could get half a house) or not (e.g. we do not allow for watching half of a movie). A resource can also be *shareable* or not, depending on whether multiple agents can be allocated the same resource (e.g. multiple users could join a movie viewing, but multiple users would be unlikely to share a purchased item of clothing). Finally, resources/items can either be *static* or not, depending on whether its properties do not change during the interaction with the agents. For example, a piece of clothing might stop being discounted, a food item might perish, or a movie that has been recommended too many times might lose its appeal to certain demographics (though the latter might be best handled as part of the preference function).

2.7.1 Preference models and Social welfare

Though applications in MARA do make use of the preference models presented in Section 2.5 (Chevaleyre *et al.*, 2008; Cafaro *et al.*, 2013; Bouveret *et al.*, 2016), the characteristics of certain allocated items (also referred to as tasks or resources) in many applications have led to the development of other preference models. For example, *k-additive* utility functions (Chevaleyre *et al.*, 2004) provide evaluations for *bundles* of allocated items and are a generalisation of many models addressed in this chapter, while *weighted propositional formulas* represent preferences using logical operations (Lang, 2004). A comprehensive review can be found in Chevaleyre *et al.* (2006).

Ultimately, the goal of MARA is to produce an allocation of users to resources. Often, the aim is to compute an *optimal* allocation, according to some prespecified metric. The term *social welfare* is typically used to address a metric of optimality that captures

the quality of the allocation across the set of agents/users. The most common such metric, often used synonymously with social welfare (Wooldridge, 2009), is *utilitarian* social welfare, i.e. the sum of individual agent's utilities (Chevaleyre *et al.*, 2006). Variations of this include the *Nash product* (Darmann and Schauer, 2015), which is the product of individual agent's utilities, and *rank dictators* (Weiss, 2013) which is a family of functions that equates social welfare with the utility assigned to a specific agent. Examples of this are *egalitarian social welfare*, which is equal to the utility of the worst off agent, and *elitist social welfare*, which is equal to the utility of the best off agent. Utilitarian social welfare, rank dictators, and the Nash product are examples of *collective utility functions*, i.e. mappings from a vector of individual utilities to a collective utility (Chevaleyre *et al.*, 2006).

Game-theoretic concepts, such as *Pareto-optimality* (Pardalos *et al.*, 2008) are also viable. An allocation is Pareto optimal if it is not Pareto-dominated by any other allocation, i.e. if there is no other allocation that would make at least one player better off without hurting any other player. Another relevant criterion is *envy-freeness*, where every user is at least as happy with their item, as they would be with any of the items allocated to another user. Since envy-freeness and Pareto-optimality are not always achievable simultaneously, it is common to attempt to minimise some criterion of *envy*, such as the number of envious agents, or the average distance to the most envied agent (Lipton *et al.*, 2004).

2.8 Conclusion

Though we have focussed on the prediction of user ratings, recommender systems provide enough incentive for additional metrics (Ricci *et al.*, 2010). The *cold-start problem*, the bad performance of the system with new users and items, calls for evaluating the accuracy of predictions in such cases, perhaps as a trade-off with

normal performance. The *confidence* in a recommendation, e.g. the probability of a recommendation being correct, can be a useful thing to show to a user, increasing his *trust* in the system, i.e. the probability of accepting the recommendation. Lastly, the *novelty* and *serendipity*, or surprise factor, of a recommendation can be important in retaining users, while the *diversity* of recommendations might increase their usefulness.

Recommendation is typically described as the end goal of preference elicitation systems. This chapter presented the fundamentals of Decision Theory and Preference Elicitation, giving emphasis to the utility models commonly used in the literature for representing user preferences. However, updating even an additive model based on feedback can take significant computational time, which increases with the size of the vector description of items. Moreover, update procedures need to be repeated for each query-response combination when considering EVOI computations. This time can be prohibitive in real-time interactions with a user, even in the myopic EVOI case, as we will see in the next chapter. An explicit way of modelling the preferential indifference between alternatives would allow for 'ignoring' parts of the solutions space when considering queries and recommendations. Moreover, if this model represents such indifference in the form of 'categories' of solutions, we will be able to significantly reduce the time of a model update, which would no longer increase with solution space dimensionality. Following up on this observation, and motivated by research into how human's think *categorically*, the next chapter builds a theory of coarse preferences, focussed around a model of preferential independence that partitions the space of alternatives into equivalence classes, based on the user's preferences. We further provide motivation for utilising the model when its assumptions are compatible with user behaviour. Chapters 4, and 5 then provide experimental evidence for the context-dependent validity of these assumptions.

Chapter 3

A Decision Theoretic Model of Coarse Preferences

In Chapter 2 we examined the problem of making recommendations, especially as it concerns their online adaptation to user behaviour. We detailed how current models for the online learning of preferences are computationally intensive, and explained that might be a reason for their general lack of use in the Recommender Systems industry.

In this chapter we build on that insight in order to propose utility models defined over a discrete uni-variate latent space, in the hope that the reduced dimensionality will speed up both the computation time and the rate of convergence of learning procedures for preference models. We demonstrate how inference with these models can be significantly faster than with current approaches, and propose a decision theoretic model for their instantiation which we term *coarse preferences*. We examine how to compromise a set of coarse preferences models, such that decisions can be made over sets of users, or when the specific model for a user is unknown, e.g. during learning. Finally, we propose a generative model of coarse preferences, examining inference with a *space* of latent spaces.

3.1 Introduction

Preference Elicitation (PE), the active online learning of preferences, has a long history in decision support (Chen and Pu, 2004). However, as discussed in Chapter 2, overbearing computational costs have prohibited PE from seeing mainstream use in recommender systems. The primary objective of this thesis is to propose a model of preferences that will retain the qualities of personalisation, online learning, and explicit rational user behaviour, while reducing the complexity of inference and learning. Not only is it expected that this will make PE more attractive for modern recommender systems, but it should enable otherwise computationally prohibitive operations to take place. Coordination through set recommendation, first introduced in Chapter 2, and further analysed later on in Chapter 5, is one such problem.

In developing such a model, we will examine how mapping our space of solutions to a uni-variate latent space of categories can speed up inference. We then address an important problem of such a representation: they are personalised and do not generalise across users. This flaw would result in significant cold-start user and item problems. Further, it would render any benefits from inference void, whenever there is uncertainty over the latent space of categories. We amend this problem by showing how to combine individual mappings into a single latent space, valid across all users.

The last core section of the chapter examines how to use a generative model for representing the uncertainty over a user's latent space and preferences.

3.2 Categorisation in Inference with Utility Functions

This section considers the problem of inferring an optimal decision given a utility function representing a user's preferences. Except where noted, we consider the utility

function to be *known*, i.e. there is no uncertainty over what the user's preferences are. Though one would not be wrong in calling this search-based optimisation, the inclusion of uncertainty in this analysis would only affect the computational time of a lookup which, provided all densities are of the same type of distribution, can be safely assumed constant.

We refer to alternatives as outcomes, to emphasise that we are omitting the analysis of the action-outcome relation. This latter dependency, where present, will constrain the space of possible outcomes in a way that would not make it *practical* to simply maintain an explicit ranking over all outcomes, which one would then traverse when maximising for user utility.

We first consider the case where the utility function is a mapping from a discrete set of outcomes before addressing the more common case of a vector representation. In each case, we will address the effects of grouping outcomes or partial descriptions of outcomes into categories, and how it can expedite inference. The next section will then describe a formal theory for representing utility functions over such categories while Section 3.4 will address how to combine different such representations for inference with multiple users or user types.

3.2.1 Inference with non-decomposed utility functions

As a trivial example, we first consider *non-decomposed* utility functions $u_i : O \to \mathbb{R}$, for some user $i \in I$, where O is simply a finite set of outcomes $O = \{o^1, o^2, ..., o^n\}$, and the utility function for any user is defined by a lookup table where all utility values for each outcome are enumerated.

With no access to distributions, or more constrained bounds, over utility values, determining an optimal allocation will require a lookup for each outcome in O, and the

application of some ranking or maximisation procedure. A classification of outcomes into different categories according to their assigned utility we be represented through a table with rows defining preferential equivalence classes and columns defining inclusion (binary) or, alternatively, a list of included outcomes. Fig. 3.1 illustrates these two alternative representations.

	o^1	o^2	o^3	...	o^n
c^1	0	1	0	...	0
c^2	0	0	0	...	1
⋮	⋮	⋮	⋮	⋱	⋮
c^m	1	0	1	...	0

(a) Tabular representation.

c^1	$\{o^2, ...\}$
c^2	$\{..., o^n\}$
⋮	⋮
c^m	$\{o^1, o^3, ...\}$

(b) List representation.

Figure 3.1: An example of representations of preferential equivalence classes over outcomes, as defined by the corresponding categories. Each outcome is mapped to one category.

Deciding on an optimal outcome is equivalent to looking up one outcome for each equivalence class, for the case of one user. Assuming the user's preferences are such that we can split outcomes into $|C_i|$ groups, we will need $|C_i|$ lookups and one maximisation step. All other things being equal, optimising over the space of categories will obviously be preferable when $|C_i| < |O|$.

3.2.2 Inference with additive independent utility functions

Next, we assume that outcomes can be described by vectors of m variables $x = [x_1, x_2, ..., x_m]$, where $\mathbb{X} = \times_{j=1}^{m} \mathbb{X}_j$,. Further, we assume that the utility function of each user $i \in I$ can be written as $u_i(o) = u_i(x) = \sum_{j=1}^{m} u_{i,j}(x_j)$, i.e. through an *Additive Independent (AI) Decomposition* (Koller and Friedman, 2009) where the total utility

is the sum of individual utilities accrued for individual properties of the outcome. We will write $u_{i,j} : \mathbb{X}_j \to \mathbb{R}$ for the subutility function that maps each assignment x_j to a real value.

Assuming discrete and finite variable domains, these utility functions can be represented with m look up tables of size $|\mathbb{X}_j|$, $\forall j = \{1,...,m\}$ and for any user $i \in I$. The analysis now depends on whether O is equal to $\times_{j=1}^{m}\mathbb{X}_j$, or not. If the former is the case, we can decide independently on an assignment x_j, $\forall j = \{1,...,m\}$ because u_i is a sum of independent sub-utilities with no constraints over the individual space of assignments for each property. Choosing an assignment for one variable does not constraint our choice over the rest. We will therefore need to perform $|\mathbb{X}_j|$ lookups, for each variable j, in a manner similar to Sec. 3.2.1. This leaves us with a total of $\sum_{j=1}^{m}|\mathbb{X}_j|$ lookups and m maximisation steps.

In the general case, where $O \subseteq \times_{j=1}^{m}\mathbb{X}_j$, we can no longer independently optimise over each variable assignment, since not all combinations of assignments will belong to O. The simplest approach to finding $\text{argmax}_{o\in O}\, u_i(o)$, under these circumstances, is to evaluate $u_i(o)$ $\forall o \in O$ for any $i \in I$, similarly to 3.2.1, but with each utility value lookup replaced with m lookups and the summation of their values. This translates to $|O| \cdot m \leq \prod_{j=1}^{m}|\mathbb{X}_j| \cdot m$ lookups in total, and one maximisation over the entire set of all enumerated possible solutions.

Assume that, for a specific user, we could group assignments to each variable x_j into disjoint groups of equal partial utility values $c_{i,j} \in C_{i,j}$. Deciding on an optimal allocation for this user will entail looking up the subutilities of assignments to m variables, for each available category, with an upper bound of

$$m \cdot \prod_{j=1}^{m}|C_{i,j}| \tag{3.1}$$

lookups.

3.2.3 Inference with generalised additive independent utility functions

Generalising the previous section's model in order to allow subutilities to be defined over more than one outcome variable, we can define utility functions $u_i(o) = \sum_{l=1}^{k_i} u_{i,l}(x_{i,l})$, where $x_{i,l}$ are assignments to a set of variables $\mathcal{X}_{i,l} \subseteq \{x_1,...,x_m\}$, such that $\cup_{l=1}^{k_i} \mathcal{X}_{i,l} = \{x_1,...,x_m\}$. This type of decomposition implies that utility functions exhibit Generalized Additive Independence (GAI) (Boutilier *et al.*, 2001). In this case, variable sets that jointly affect overall utility are not necessarily disjoint. Therefore, unlike the AI case, we can not generally optimise independently over sets of 1 or more variables, even if $O = \times_{j=1}^{m} \mathbb{X}_j$ holds. Consequently, $\forall$ users $i \in I$, we will have to evaluate $u_i(o)$ $\forall o \in O$. This leads to $|O| \cdot k_i \leq \prod_{j=1}^{m} |\mathbb{X}_j| \cdot k_i$ value lookups, and one maximisation step over the resulting value sums. If the user's preferences are such that we can group assignments to each set of variables $\mathcal{X}_{i,l}$ into groups of assignments $|C_{i,l}|$, then deciding on an optimal allocation for this user will entail looking up the subutilities of assignments to k_i sets of variables, for each of, at maximum, $\prod_{l=1}^{k_i} |C_{i,l}|$ cases, with an upper bound of

$$k_i \cdot \prod_{l=1}^{k_i} |C_{i,l}| \tag{3.2}$$

lookups.

3.2.4 Summary on inference

Table 3.1 summarizes results regarding the number of lookups needed to compute an optimal allocation, with and without explicitly representing categories of outcomes, and for different utility function decompositions. This number of lookups is an indication of how computationally involved inference will be. The rightmost columns

give the upper bounds on the number of outcomes and categories, given the variable domains. These become equalities when each possibility is present in the available outcome or category space, respectively. We note that computational gains can be significant, depending on how the user groups outcomes into categories.

Utility function	***Non*-coarse**	**Coarse**	$\lvert O\rvert \leq$	$\lvert C\rvert \leq$
Non-decomposed	$\lvert O\rvert$	$\lvert C\rvert$	-	-
AI-decomposed	$\lvert O\rvert \cdot m$	$\lvert C\rvert \cdot m$	$\prod_{j=1}^{m} \lvert \mathbb{X}_j\rvert$	$\prod_{j=1}^{m} \lvert C_{i,j}\rvert$
GAI-decomposed	$\lvert O\rvert \cdot k_i$	$\lvert C\rvert \cdot k_i$	$\prod_{j=1}^{m} \lvert \mathbb{X}_j\rvert$	$\prod_{l=1}^{k_i} \lvert C_{i,l}\rvert$

Table 3.1: Summary of number of utility value lookups; with and without explicitly representing groups of outcomes, or partial outcomes, in preferences, and for different utility function decompositions.

In grouping outcomes into categories of equal utility, and demonstrating the computational speed-up of inference when this is explicitly represented, we have made an argument for modelling user preferences over a latent space of categories. However, we have no guarantees that such a representation would be compatible with the principles of rational behaviour, as presented in von Neumann and Morgenstern (1953). The next section derives a decision theoretic model for the representation of utility functions over such a latent space of categories.

3.3 Coarse Preferences

Assume two outcomes $o, o' \in O$ and an arbitrary user. Recall from Section 2.5.1, that a weak preference $o \succeq o'$ of outcome o over o' is a binary relation indicating that the user *weakly* prefers outcome o to o', or, in other words, o is at least as good as o', for

that user. The *weak preference relation* is typically expected to satisfy the properties of *comparability* and *transitivity*, if it is to be considered *rational* (von Neumann and Morgenstern, 1953). The binary relations of *indifference* and *strict preference* are defined in terms of the weak preference relation as:

$$o \sim o' \Leftrightarrow o \succeq o' \wedge o' \succeq o \tag{3.3}$$

$$o \succ o' \Leftrightarrow o' \not\succeq o. \tag{3.4}$$

We are now ready to give the definition of *coarse preferences*:

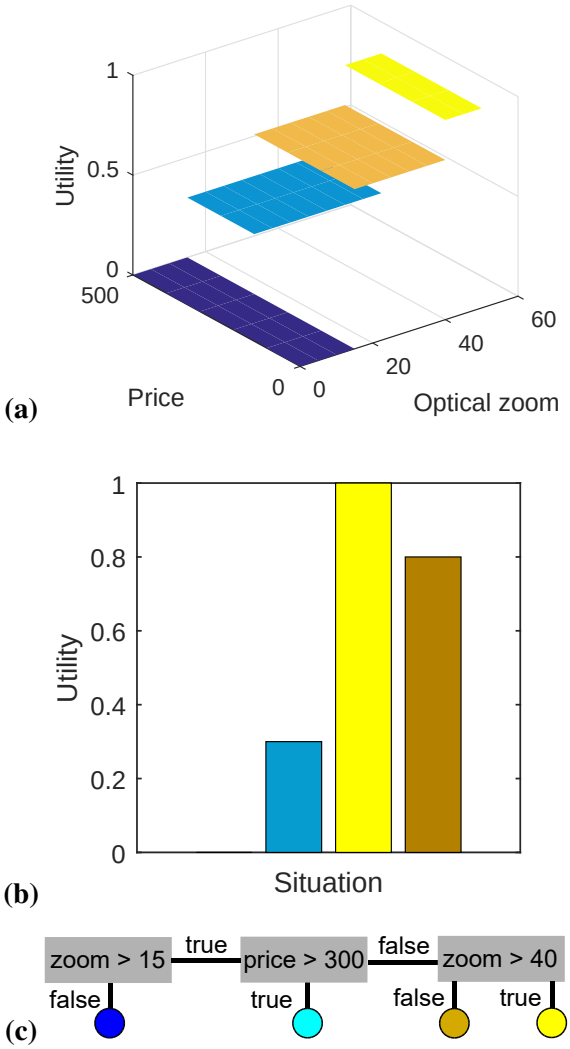
Definition 3.3.1 (Coarse preferences)**.** We say that a Decision Maker (DM) exhibits *coarse preferences* ϕ over outcomes $o \in O$ (or is a ϕ-coarse DM) if given a many-to-one mapping $\phi : O \rightarrow C$ from a space of outcomes to a space of *situations* or *categories*, with $\phi(o) = c, \phi(o') = c'$, we have:

$$o \succeq o' \Leftrightarrow \phi(o) \succeq \phi(o') \Leftrightarrow c \succeq c' \tag{3.5}$$

Intuitively, a user identifies in each outcome a corresponding category, according to a mapping ϕ, and maintains a preference ranking over these categories rather than directly over the space of outcomes. An alternative view of these categories is that of *preferential equivalence classes* over the space of outcomes; every category defines a subset of the space of outcomes, all members of which share the same utility value.

Key to our model of *coarse preferences* is the idea that a user's preferences are *coarser* than what the item vector space might imply. Specifically, it is often the case that his/her evaluation over an item is not as fine-grained as the combination of variable domains describing it but, rather, that they perceive alternatives in terms of the *category* they belong to. Essentially, a user identifies in each item a corresponding category, according to a mapping ϕ, and maintains a preference ranking over these categories

Figure 3.2: Representation of a user's coarse mapping for the camera example. (a) depicts the user's utility over different combinations of camera price and optical zoom factor. (b) depicts the equivalent utility function over the space of categories as defined by the partitioning represented by the decision tree in (c).



rather than directly over the space of items. For example, consider the user aiming to buy a new camera with their preferences being represented by the utility function in Figure 3.2a. We can partition the combined space of the variables "price" and

"resolution" according to the decision tree in Figure 3.2c and represent the user's utility function in the space of categories in Figure 3.2b.

We revisit Jon's situation from Chapter 2. Jon had been presented with a *black*, *unisex scarf* of *brand AB*, for the retail *price* of 26 *pounds*, which included *no discount*, and that was *out of stock*. He is asked to evaluate the item's utility before comparing it to his threshold of 0.6, in order to decide whether to accept or reject the item.

Jon takes a look at whether the item is *in stock* and, since that isn't the case, he checks the item's brand to see whether he would consider waiting for it, but does not find *AB* to his liking. He then checks the *price* of the item in case it might be a bargain, but finds it expensive. He never checks for any other details on the item and assigns it a utility of 0.10. He proceeds to swipe left on the item, rejecting it.

Corollary 3.3.2. ***Indifference and strict preference****: If a DM exhibits coarse preferences then it follows from Def. 3.3.1 and Eq. 3.3 and 3.4 that:*

$$o \sim o' \Leftrightarrow \phi(o) \sim \phi(o') \Leftrightarrow c \sim c' \tag{3.6}$$

$$o \succ o' \Leftrightarrow \phi(o) \succ \phi(o') \Leftrightarrow c \succ c' \tag{3.7}$$

Below, we detail the proof of the existence of a coarse utility function, over categories, representing the same preferences as a given utility function over outcomes. We can therefore, given a utility function $u : O \to \mathbb{R}$ defining a coarseness $\phi : O \to C$, write

$$u_c^{\phi} : C \to \mathbb{R}. \tag{3.8}$$

Definition 3.3.1 allows for a variety of coarse domain representations. In order to make use of such an approach for preference elicitation, however, we will need to make additional assumptions. We describe the problem of coarse preference elicitation in

Chapter 4. First, however, we describe the theoretical problem of representing coarse preferences with a utility function.

This section details the proof of existence of a coarse utility function $u_c^\phi : C \to \mathbb{R}$, over categories $c \in C$, representing the same preferences as a given utility function $u : O \to \mathbb{R}$ over outcomes $o \in O$.

We will constructively prove that, if a coarse user's preferences over outcomes can be represented by a utility function u, there exists a *coarse utility function* u_c^ϕ that can represent preferences over the space of categories. Equivalently, for a specific user or user type, assuming the existence of a utility function over the space of outcomes and a coarse mapping necessitates the existence of a coarse utility function. To start off, we denote lotteries over categories as $l_c = \langle p_1^c, c^1; p_2^c, c^2; ...; p_z^c, c^z \rangle$, where category c^j is realised with probability p_j^c.

Proposition 3.3.3 (Equivalence of simple lotteries over outcomes and categories)**.** *Assume a set of outcomes O and categories C, and a many-to-one mapping $\phi : O \to C$. Every lottery over outcomes $l = \langle p_1, o^1; ...; p_n, o^n \rangle$ with $o^i \in O,\ \forall i = \{1, ..., n\}$ defines a lottery over categories $l_c = \langle p_1^c, c^1; ...; p_z^c, c^z \rangle$ with $c^j \in C,\ \forall j = \{1, ..., z\}$, such that $p_j^c = \sum_{o^i \in O_{c^j}} p_i$, where $O_{c^j} \subseteq O : \forall o \in O_{c^j}, \phi(o) = c^j$. We write $\phi : L \to L_c$, where L, L_c are the sets of simple lotteries over outcomes and categories, respectively.*

Corollary 3.3.4. *Given Prop. 3.3.3, and since O_{c_j} are disjoint sets with union O, we have*

$$\sum_{i=1}^{n} p_i = \sum_{j=1}^{z} p_j^c \tag{3.9}$$

We will now use the above to prove that there indeed exists a coarse utility function u_c^ϕ, that represents the same preference ordering as u, given the mapping $\phi : O \to C$:

Proposition 3.3.5 (Utility functions representing coarse preferences)**.** *Assume a coarse DM, $\phi : O \to C$, with a utility function $u : L \to \mathbb{R}$ representing $\succeq$, and with sets of simple lotteries over outcomes and categories L, L_c, respectively. There exists a utility function $u_c^\phi : L_c \to \mathbb{R}$ representing $\succeq$, such that $u(l) = u_c^\phi\big(\phi(l)\big)$.*

Proof: Define a function $u_c^\phi : C \to \mathbb{R}$ with $u_c^\phi\big(\phi(o)\big) = u(o)$, and denote $u_c^\phi(l_c) = \sum_{j=1}^{z} p_j^c\, u_c^\phi(c^j)$, where $l_c = \langle p_1^c, c^1; ...; p_z^c, c^z\rangle$ with $c^j \in C,\ \forall j = \{1, ..., z\}$. Also assume the lottery $l = \langle p_1, o^1; ...; p_n, o^n\rangle$ with $o^i \in O,\ \forall i = \{1, ..., n\}$ with $l_c = \phi(l)$. We know from the Eq. 2.28 that $u(l) = \sum_{i=1}^{n} p_i\, u(o^i)$, but from Cor. 3.3.4 and since we defined $u_c^\phi(\phi(o)) = u(o)$, we have:

$$\sum_{i=1}^{n} p_i\, u(o^i) = \sum_{j=1}^{z} \sum_{o^i \in O_{c^j}} p_i\, u_c^\phi(c^j) = \sum_{j=1}^{z} p_j^c\, u_c^\phi(c^j) \Leftrightarrow u(l) = u_c^\phi(l_c). \tag{3.10}$$

From the expected utility definition, given lotteries l and l' we have $l \succeq l' \Leftrightarrow u(l) \geq u(l')$, but because $u(l) = u_c^\phi(l_c)$, and denoting $l_c = \phi(l)$ and $l'_c = \phi(l')$, we finally get:

$$u(l) \geq u(l') \Leftrightarrow l \succeq l' \Leftrightarrow l_c \succeq l'_c \Leftrightarrow u_c^\phi(l_c) \geq u_c^\phi(l'_c). \tag{3.11}$$

□

An important side-effect of proving the equivalence of u and u_c^ϕ is that preference elicitation (which we will investigate in the context of coarse preferences in Chapter 4) can be done with queries over the original outcome space, that therefore elicit u, even as we assume that u_c^ϕ was used in answering the query. Related to this is that the expected

utility of a recommendation requires estimating the expectation over u. Essentially u is a less structured representation of u_c^{ϕ}, which we eventually wish to elicit along with ϕ.

3.3.1 Form of Outcome-Space Utility Function given Coarsenesss

So far we have assumed the existence of of a utility function over the original space of outcomes O, without defining it further. The assumtion of coarse preferences restricts the form of this utility function u. We need $u(o) = u_c\big(\phi(o)\big)$, or:

$$u(o) = \sum_{c \in C} \mathbb{I}_{\phi(o)=c} \cdot u_c(c), \tag{3.12}$$

where exactly one indicator function $\mathbb{I}$ is active for any x. We can read the above as an additive decomposition of the utility function, and rewrite it as:

$$u(o) = \sum_{c \in C} b_c \cdot u_c(c), \tag{3.13}$$

where $b_c = \mathbb{I}_{\phi(o)=c} \in \{0,1\}, \forall c \in C$ and the constraint $\sum_{c \in C} b_c = 1$, and since $u_c(c), c \in C$ is a constant:

$$u(o) = \sum_{c \in C} v(b_c), \tag{3.14}$$

with $v(b_c) = b_c \cdot u_c(c)$.

3.3.2 Partial categories

Coarse preferences assume a many-to-one mapping from a space of outcomes to a space of categories. Though there are no constraints other than the user's preferences, it would be useful for us to make use of any structure present in the respective outcome

space utility function. In particular, we are interested in maintaining the decomposition of outcomes into assignments to a set of variables, even when working in the space of categories. Failing to do so would mean that we would have to map each individual outcome separately to a category, thereby relinquishing any benefits resulting from the original decomposition.

In order to achieve this goal, we will examine mapping assignments to subsets of variables $\mathcal{X}_t \subseteq \{x_1,...,x_m\}$, to partial categories. Intuitively, this reflects the fact that the user is able to independently evaluate assignments to these subsets, which is also the assumption behind the original outcome space decomposition, as we saw in Section 3.2.2.

For this purpose, let us assume that categories can be represented by a vector of *partial categories*, such that the mapping ϕ can be analysed to a set of functions ϕ_t, with $t = \{1,...,h\}$. ϕ_t are many-to-one mappings from a space $\mathbb{X}_t$ of potential assignments to a set of variables $\mathcal{X}_t \subseteq \{x_1,...,x_m\}$, to a space of *partial categories* C_t:

$$\phi_t : \mathbb{X}_t \rightarrow C_t, \tag{3.15}$$

with corresponding coarse subutility functions

$$u_{c,t} : C_t \rightarrow \mathbb{R}, \tag{3.16}$$

such that

$$u_c^{\phi}(o) = \sum_{t=1}^{h} u_{c,t}(\mathcal{X}_t). \tag{3.17}$$

We term $\mathcal{X}_t, \forall t \in \{1,...,h\}$ *Partial Situation Variable Sets* (PSVS) and require that their union fully covers (though is not necessarily a partition of) the set of variables

$\{x_1,...,x_m\}$:

$$\cup_{t=1}^{h} \mathcal{X}_t = \{x_1,...,x_m\}. \tag{3.18}$$

The variable subsets over which original subutility functions are defined do not need to be the same over which coarse subutility functions are defined. However, the equivalence of u and u_c^{ϕ} forces some constraints. Essentially, a user can not have an outcome space utility function that is decomposed in such a way, that utilities cannot be computed for partial situations. All variable subsets over which u's subutility functions are defined, must be equal to the union of one or more PSVS. These need not necessarily be disjoint. Formally:

Proposition 3.3.6. *If given a preference ordering $\succeq$, there exists any preferentially independent variable set A, such that $\nexists \tau \subseteq \{1,...,h\} : A = \cup_{t\in\tau}\mathcal{X}_t$, then the situation decomposition ϕ_t, with $t = \{1,...,h\}$, is not compatible with that ordering.*

3.4 Coarse Mappings for Multiple Users and User Types

We will examine decisions affecting a user of unknown type, drawn from a known set of coarse user types, with some known prior. A key issue is that, even as coarseness produces a preferential discretisation of the outcome space, this discretisation is, generally, unique to each user type. Examining the problem of recommendation, we can see how this becomes problematic: it prohibits us from making decisions over any specific situation space C_τ, of a particular user type τ.

The above becomes obvious if we consider evaluation of two outcomes o, o' such that both belong to the same preferential equivalence class c_1^1 (i.e. $\phi(o) = \phi(o') = c_1^1$) for

user 1, but belong to different situations c_1^2, c_2^2 for user 2 (i.e. $\phi_2(o) = c_1^2, \phi_2(o') = c_2^2$). We should not expect from a general evaluation of recommendations to be indifferent between outcomes in c_1^1. We could, of course, optimise over the original outcome space. However, this would undermine our goal of computational cost reduction. The question arises: *Is there a way of maintaining some of the benefits from the outcome space partitioning, while still making consistent decisions given a set of viable user types*? The above example points towards such an approach, where categories are split into sub-categories such that, for each user, all outcomes mapping to each sub-category belong to the same category, for that user.

3.4.1 Considering decisions given a set of coarse user types

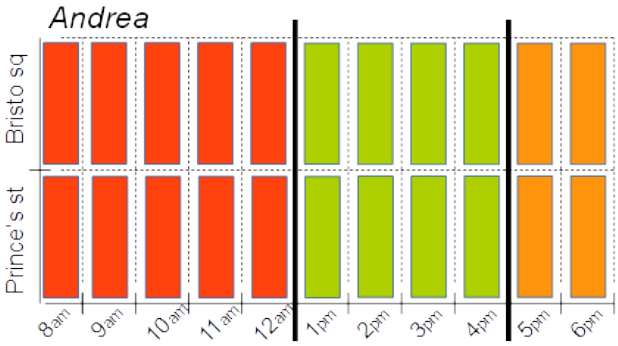
Let us now assume that users belong to one of a known set of types T, with each $\tau \in T$ defined by a coarse representation, $\phi_\tau : O \to C_\tau$. Each such mapping, represents a partitioning of the space of outcomes O into a set of preferential equivalence classes $\mathcal{C}_\tau = \{O_c^\tau, \forall c \in C_\tau\} : O_c^\tau \subseteq O$ with $\forall o \in O_c^\tau, \phi(o) = c$. We can define the point-wise intersection of all user types' equivalence class partitionings as:

$$\mathcal{C}_T = \{\cap_{\forall \tau} O_c^\tau, \forall O_c^\tau \in \mathcal{C}_\tau \forall \tau\}. \tag{3.19}$$

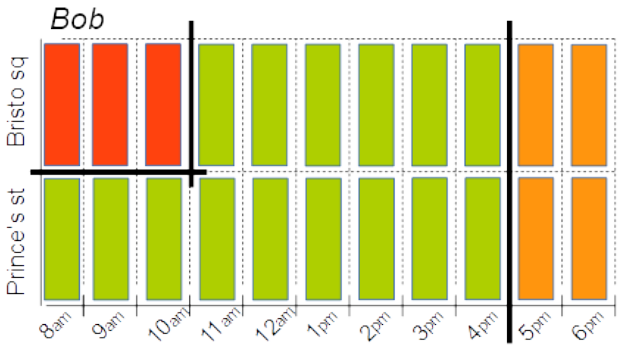
For example, consider a problem where users need to coordinate to pick a ride from Edinburgh to Glasgow, departing either from *Bristo Sq.* or *Prince's St.*, at any full hour between 8am to 6pm, and where each user has preferences over different categories of potential rides, as defined over this space. Specifically, consider the users Andrea and Bob in Figure 3.3 (a) and (b). These users will belong to the user type *Andrea & Bob* (as well as the user types *Andrea* and *Bob*, respectively).

It is not necessary for a user to exhibit the exact same coarseness as a type in order

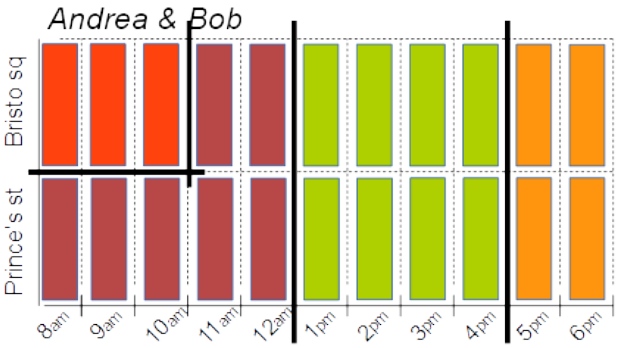
Figure 3.3: Andrea (a) is indifferent towards the departure location for the car ride, but splits possible rides into 3 different categories based on the time of departure. Bob (b), on the other hand, singles out any ride departing from *Bristo Sq.* before 10am, and any ride past 5pm, while being indifferent between any other option. If we were to pick a ride for a user that we know behaves either as Andrea or Bob, even if we do not know like whom precisely, then we could make such decisions over the space of categories produced by their intersection (c), without loss of information.



(a)



(b)



(c)

to belong to it. In fact, it is enough that any decisions made with the use of a type's coarse representation are consistent with decisions one would have made if he had direct access to the user's coarse representation. To verify this, consider that if a user is indifferent between outcomes in a set, then they will also be indifferent between outcomes in a subset of that set.

Consider now a problem where we have to make a decision over a set of users belonging to different coarse user types. We can infer a coarse representation for the set of coarse user types as below:

Theorem 3.4.1 (Coarse Representation for Sets of User Types)**.** Assume a set of user types T, representing coarse preferences $\phi_\tau : O \to C_\tau$, with some ranking $\succ$ over C_τ, $\forall \tau \in T$, given a space of potential outcomes O. The point-wise intersection of the user types' preferential equivalence class partitionings, results in the maximally coarse representation of the outcome space, $\phi_T : O \to C_T$, such that each user of type $\tau \in T$ is indifferent among outcomes in each non-empty preferential equivalence class in the resulting partition.

Proof: Proving that each user of a specific type is indifferent among outcomes in any of the resulting preferential equivalence classes is trivial. Each such equivalence class is a subspace of each user equivalence class from which it was constructed. The preferential indifference among all outcomes of that class still holds for any of its non-empty subspaces. Proving that the above construction is maximal, requires proving that adding any outcome to any resulting equivalence class would make it lose the above property. Assume that it did not. Then, for each user, it would have belonged to the respective equivalence class which constructed the resulting equivalence class. It would therefore have been included in it. Consequently, it is not a new addition to it. □

Continuing the above example in Figure 3.3, consider another user Carlton, who would also be of the user type *Carlton*. We can make decisions for the users Andrea, Bob, and

Carlton, as well as for any user for which we could have made decisions for with the models of these 3 users, using the point-wise intersection of the partitionings for the user types *Andrea & Bob* and *Carlton*. This is the user type *(Andrea & Bob) & Carlton* in Figure 3.4

We will refer to the set of user types T defining $\phi_T : O \to C_T$ as the *seed* group of the partitioning $\mathcal{C}_T$. If the provided rankings over situations allow indifference, and we are provided with rankings of the type $\succeq$ rather than $\succ$, then we can obtain the maximally coarse representation by, for each user type, re-mapping any outcomes belonging to preferential equivalence classes between which that user is indifferent, to the same situation. The above theorem will then apply.

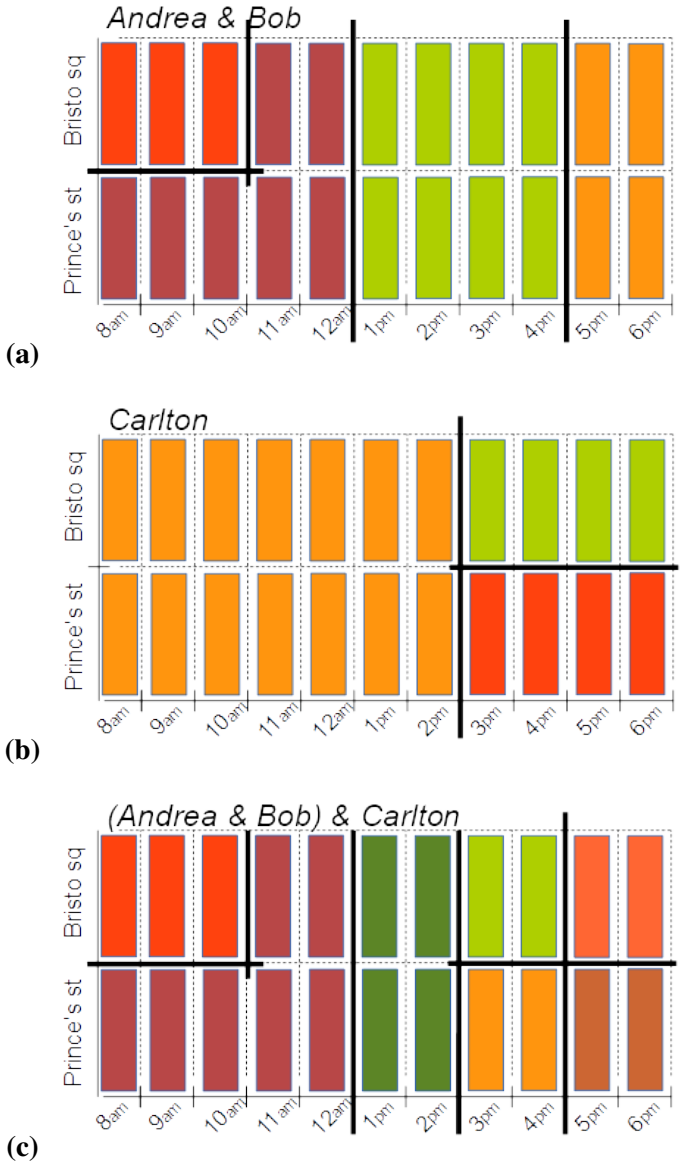
One overlooked problem in our analysis, so far, is that the computation of point-wise intersections can be costly. Moreover, these will, generally, vary with the seed group. One naturally wonders whether this computation can be transferred offline. An interesting observation is that the point-wise intersection of the resulting group outcome space partitioning with any of those of the respective group members will result in that group partitioning itself. In fact this can be true, even for users who do not belong to the *seed* group. Crucially, the resulting situation space can be used to make decisions over any set of users for which this is true. This leads to a very natural definition of user typing in the context of coarseness:

Definition 3.4.1 (coarse preference User Types)**.** Assume a partitioning $\mathrm{A} = \{A_1, ..., A_k\}$ of the space of outcomes O, and the coarse preferences partitioning of the outcome space $\mathcal{C}_i$, for some user $i \in I$. We will say that user i is of *coarse preference User Type A*, iff the pairwise point intersection of $\mathcal{C}_i$ with A results in A.

We can relax the definition above by allowing approximate user types with bounded loss of accuracy in terms of utility values:

Definition 3.4.2 (ε-close coarse preference User Types)**.** Assume a partitioning $\mathrm{A} =$

Figure 3.4: Users of the type *Andrea & Bob* (a) partition the space of potential rides into those past 5pm, those between 1pm and 4pm, those leaving from *Bristo Sq.* before 10am, and the rest. Carlton, and those of the same type (b), split their options into those departing before 2pm, those departing past 3pm from *Bristo Sq.*, and those departing past 3pm from *Prince's St.*. If we were to pick a ride for a user that we know behaves either as one of the type *Andrea & Bob*, or *Carlton*, then we could make such decisions over the space of categories produced by their intersection (c), without loss of information.



$\{A_1,...,A_k\}$ of the space of outcomes O, and the coarse preferences partitioning of the outcome space $\mathcal{C}_i$, for some user $i \in I$, with $\phi_i : O \to C_i$ and $u_i^c : C_i \to \mathbb{R}$. We will say that user i is of ε-*close coarse preference User Type A*, if there exists a partitioning of the outcome space $\mathcal{C}_i^\varepsilon$, for a mapping $\phi_i^\varepsilon : O \to C_i^\varepsilon$, such that the pair-wise point intersection of $\mathcal{C}_i^\varepsilon$ with A results in A. Where $\forall c^\varepsilon \in C_i^\varepsilon$, c^ε is the result of the union of equivalence classes defined by some $c \in C_i$ such that $\forall o, o' \in O : \phi_i(o) = c_1, \phi_i(o') = c_2$, for some $c_1, c_2 \in C_i$, with $u_i^c(c_1) - u_i^c(c_2) \leq \varepsilon$, we have $\phi_i^\varepsilon(o) = \phi_i^\varepsilon(o') = s^\varepsilon$.

3.5 A Generative Model of Coarse Preferences

Though not a primary concern of this thesis, we consider it important to, at this point, develop a generative model of coarse preferences. This will help substantiate the implications behind the model as we move forward, as well as provide grounds for future extensions, as it pre-empties certain concerns on our procedures in later chapters.

In Chapter 4 we will develop a methodology for the elicitation of coarse preferences. With the experiments of Chapters 4 and 5 we will successfully apply that procedure to a typical online recommender system, and on a multi-user coordination through set recomendation scenario. In both cases, we will demonstrate significant gains in recommendation quality and computational time. However, this procedure hinges on the existence of a substantial dataset in order to learn the coarse mapping, the basis for the procedure. Where this is not available, and where new users might vary significantly from those already encountered, this procedure will not suffice, even with periodic retraining, since the mapping learned will not apply to new users. The model proposed in this section could potentially allow for learning *personalised* coarse mappings *online*.

The preference model represents a probabilistic regression tree, with a prediction

available at any depth. The overall structure of the graphical model maps to that of a binary tree, with variables indicating the feature over which the tree splits at the respective node. The domain of these variables is that of all possible splits. Using this convention, we can accurately represent trees over any domain composed of binary variables, and any discrete domain by transforming it to the latter. Continuous domains can be represented this way approximately, by discretising the domain of each variable.

Regression trees are an effective way of partitioning an item space according to a target regression variable, in this case the utility, and will be used for the discriminatory coarse model in later chapters.

Consistency between the predictions at each depth of the tree is maintained by summing a predicted difference after each transition to a deeper node. This can be understood as an increase in the accuracy of our evaluation as we go down the tree. Users' coarseness is expressed by this difference going to 0. We impose this correlation to mimick the behaviour of a regression tree during learning, which maintains an empirical average over the samples mapping to the respective subspace at each node.

The tree structure and split features are learnt as representative of the whole population of users and underlying problem, whereas the evaluations at each node, and, therefore, differences between evaluations depend on either the user or the user's type. This way, we achieve the representation of the *user-set coarse mapping*, which is discussed in detail in Section 4.4.2, while still allowing for a personalised representation in the form, not only of the utility values, but also the personalised partition. The latter will be a subtree of the whole model, as defined by differences of 0 utility as we go further down the tree.

3.5.1 Regression tree learning

Regression trees are a type of decision tree (Breiman *et al.*, 1984; Loh, 2011), where the target variable, the rating in our case, is a continuous or ordinal variable. Algorithm 1 gives the general pseudocode for generating a discriminatory tree model given a set of item-rating pairs. The subsets S_1 and S_2 must be separable by an orthogonal line to the axis of the split variable x. For regression trees, the standard metric used for measuring the impurity at each *node* is the variance of samples in S_{node}.

Data: a set of item-rating pairs S

Result: a decision tree model mapping the space $\mathbb{X}$, defined by a set of variables $\mathcal{X}$, to the space of ratings $\mathbb{R}$

1. Start at the root node with $S_{node} = S$.
2. For each $x \in \mathcal{X}$, find a split of S_{node} into two subsets S_1 and S_2, such that it minimises the sum of the node impurities in the two child nodes and choose the split that minimise this criterion over all $\mathcal{X}$ and *splits*.
3. If a stopping criterion is reached, exit. Otherwise, apply step 2 to each child node, *node 1* and *node 2* in turn, with respective item-rating pairs S_1 and S_2 .

Algorithm 1: Pseudocode for tree construction by exhaustive search.

3.5.2 Generative model

Consider a space of items/solutions $\mathbb{X}$ and a population of users I, with each user $i \in I$ belonging to a user type $\tau \in T$, forming a sub-population $I_\tau \subset I$: $\cup_{\tau \in T} I_\tau = I$ and $I_\tau \cap I_{\tau'} = \emptyset$, $\forall \tau, \tau' \in T$ with $\tau \neq \tau'$. Further assume that $\forall i \in I_\tau$, $\forall \tau \in T$, that user is characterised by a utility function $u_i = u_\tau : \mathbb{X} \to \mathbb{R}$, with $u_i(x) \geq u_i(x') \Leftrightarrow x \succeq x'$, $\forall x, x' \in \mathbb{X}$, indicating his/her preference over different solutions. We assume that the

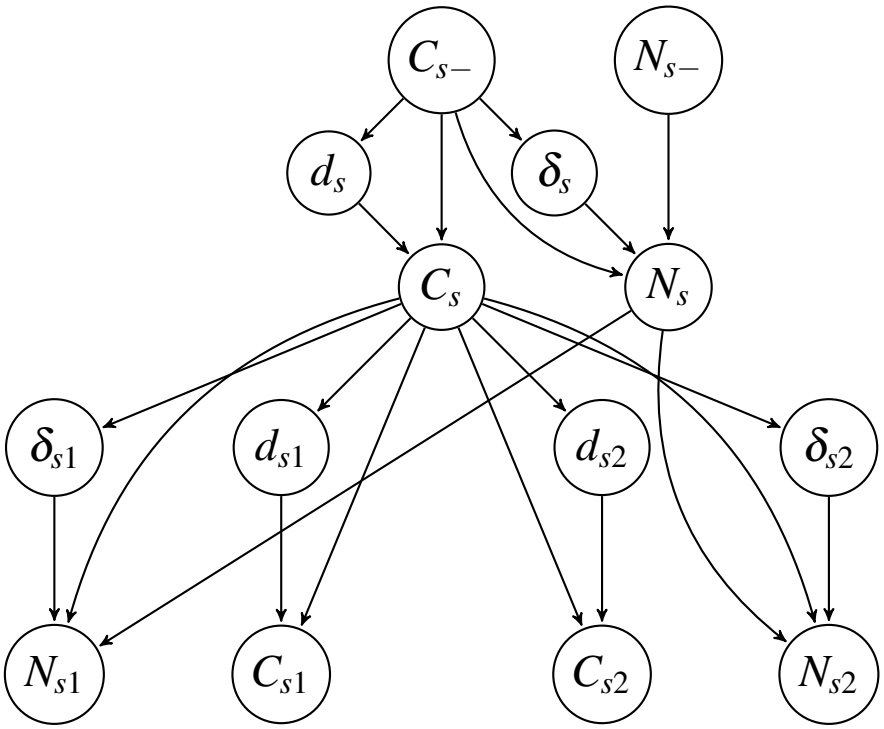


Figure 3.5: A directed graphical network for the generative model of a regression tree.

form of u_i is such that the user is ϕ-coarse ($\phi : \mathbb{X} \to \mathbb{R}$), with coarse utility function $u_i^c : C_i \to \mathbb{R}$.

If we constraint partitionings of the space $\mathbb{X}$ to those that can be produced by using hyper-lines that are drawn parallel to the axes of variables defining $\mathbb{X}$, then the space of utility functions U, with $u_i \in U, \forall i \in I$, is that of all possible regression trees over the space X. Below, we present a generative model for such a Regression Tree when the possible splits, referred to here as "features", at each node are finite.

In Figure 3.5, we give a first approach to the directed graphical network representing the generative model of coarse preferences. We take advantage of the fact that going down the tree is expected to increase the accuracy of a prediction in order to force a correlation between the evaluations of parent-children nodes.

In all occasions, subscripts to variables indicate the sequence of left-right branches

followed so far. We use s to indicate a generic trajectory, $s-$ to indicate the same trajectory up to the second last split, and $s1$ and $s2$ to refer to the extension of s with either a left, 1, or right, 2, split. All variables C are *conditional Categorical* variables indicating which feature we last split on dependent on which features we split on previously. These act as *selectors* for the variables they condition, which are either *conditional Gaussian* or *conditional Dirichlet*. We should note that a more accurate model would condition these variables over *all* past branchings in s, but we expect that conditioning on the previous split, along with the structure of the tree, will provide accurate enough results when fit to users' behaviour.

All variables δ and N are conditional Gaussian with N_s being the valuation of the regression tree at branching s, and $N_s = N_{s-} + \delta_s$ for a given C_{s-}. δ then indicates the difference of the evaluation as we refine the space of solutions. What values the evaluations of N and δ take will depend on the specifications of the problem at hand. Variables d_s are conditional Dirichlet, representing a Dirichlet distribution for each assignment to its parent C_{s-}, and act as the probability vector for C_s. The Dirichlet distribution is the conjugate prior for the Categorical.

In Figure 3.6 we further refine the model by considering the relation between the change to the prediction of the evaluation as we go down the tree, and the probability that a solution would be found in the subspace defined by each branch. For the evaluation at each child node to be consistent, the parent node's evaluation must equal the weighted sum of expectations at each child node, where each weight is the conditional probability of a point in the subspace defined at the parent node belonging to the subspace defined by the corresponding child node. We define P_s as the vector of probabilities $[p_s^1, p_s^2]$ of branching to either side of the sub-tree. If we simplify our problem by further considering that items are distributed uniformly in the space X, then there is no need to maintain a distribution over P_s, though its value will still depend on the split feature C_s (for binary variable domains and points distributed uniformly in X,

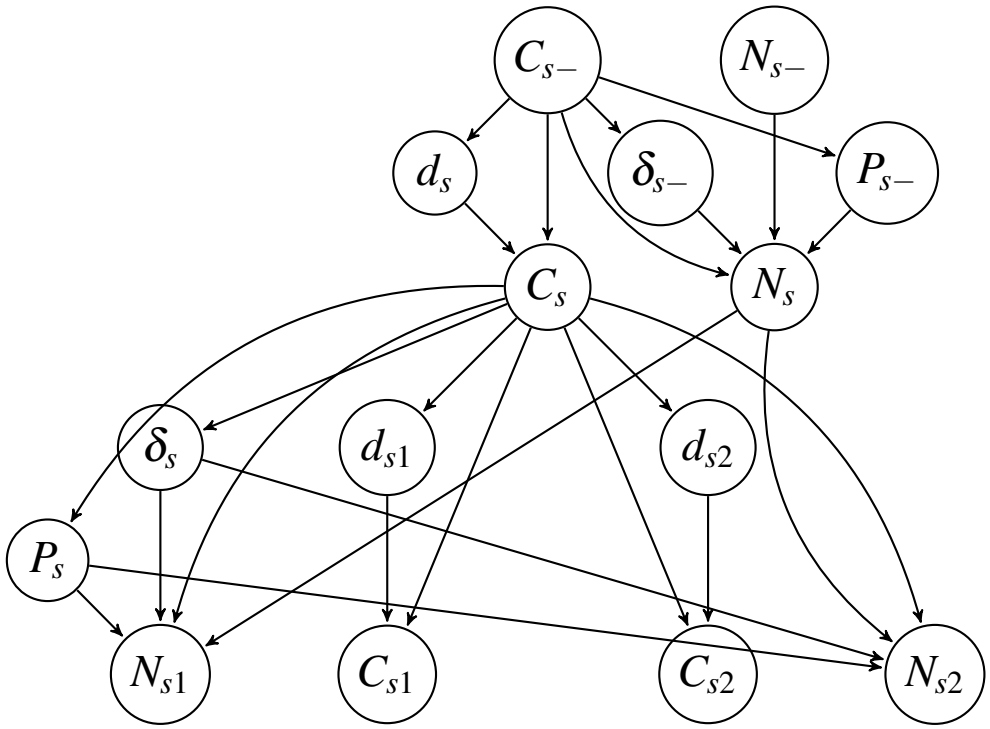


Figure 3.6: Directed graphical network for the generative model which takes into account the probability of going down each split of the regression tree.

we have $p_s^1 = p_s^2 = 1/2$). Figure 3.7 presents the factor graph corresponding to the second model.

In every case, the model presented here can be viewed as our belief over the representation of a single user's or a user type's preferences. If considering a set of possible user types to which a specific user might belong, then the model could be viewed as that of a component to a mixture model. In either case, inferring an optimal decision for the user can be done as an expectation over the model or by identifying the maximum likelihood category of each solution and taking the expectation over that category's evaluation.

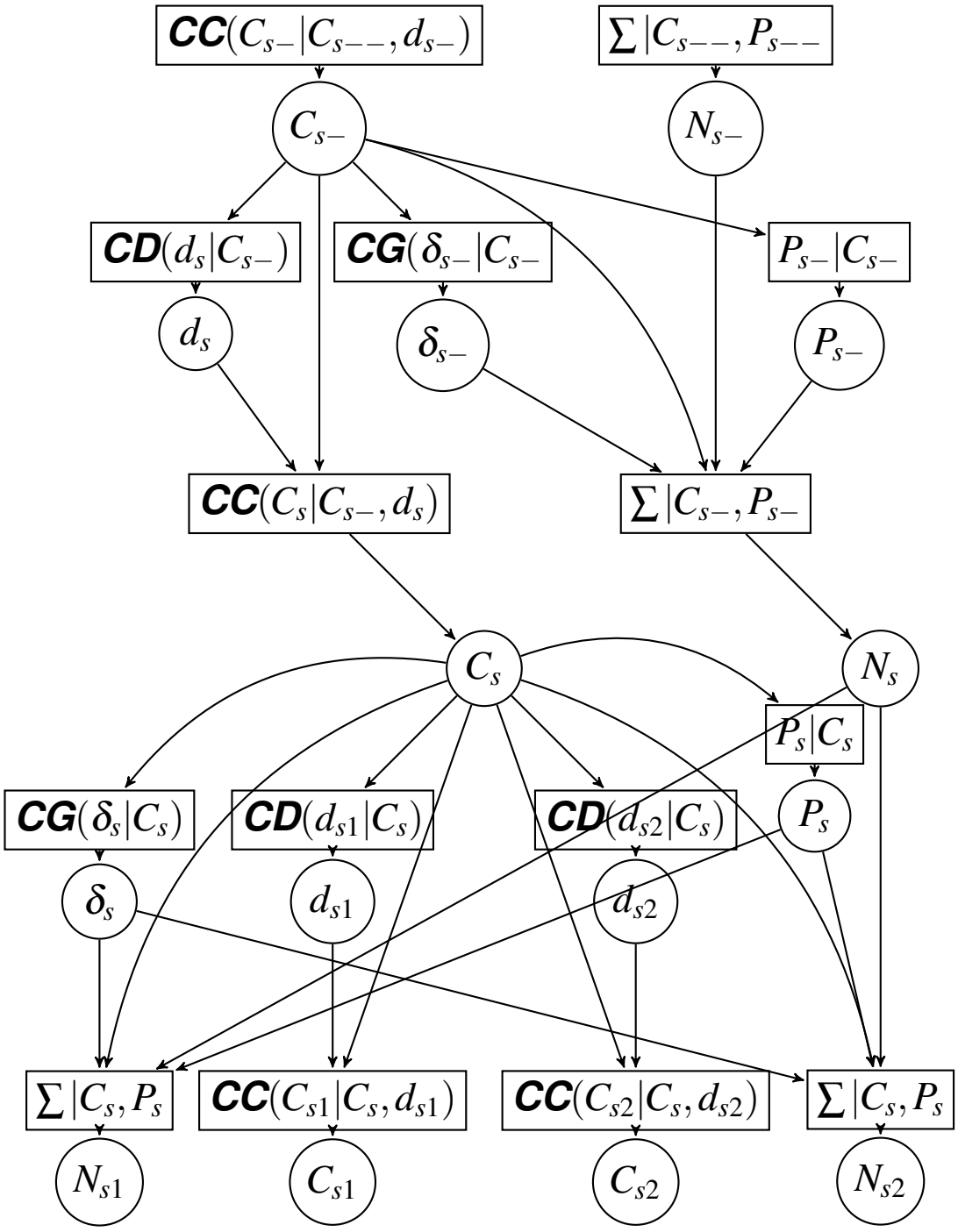


Figure 3.7: Factor graph for the generative model which takes into account the probability of going down each split of the regression tree.

3.6 Conclusions

This chapter presented a decision theoretic model of coarse preferences, representing users which make decisions by recognising the situation or *category* represented by each alternative. By being consistent with the (von Neumann and Morgenstern, 1953) Expected Utility Representation Theorem 2.5.1, we allow for the function defined over the space of categories to be a *utility* function, consistent with the user's ranking over alternatives. This will guarantee that the model learnt in Chapter 4 represents rational users and that measures such as social welfare, the sum of individual users' utility, are meaningful, provisioned on the normalisation of values across users.

Having explained the intuition behind the interest in formalising a model derived from *categorical thinking*, Section 3.2.4 detailed the computational benefits that can be achieved, given different assumptions of user behaviour and known utility functions, by representing users' preferences through a coarse utiltiy function. These benefits extend to groups of users and user typing.

The intuition, definitions, and analysis presented in this chapter serves as the basis on which the procedures in the following two chapters are developed. The next chapter details a methodology for performing preference elicitation over users drawn from a population with coarse preferences. It explains how one can learn the coarse mapping offline, by using the readily available technique of regression tree learning, and an approach for eliciting an individual user's preferences over the resulting space. The experiments in this chapter show both computational and qualititative benefits when compared to a state of the art preference elicitation procedure over an online retailer's dataset. Chapter 5 presents the multi-attribute, multi-agent problem of coordination through set recommendation, where the benefits of modelling users as coarse decision makers in decision problems involving groups of users are made evident.

Chapter 4

Efficient Preference Elicitation with Coarse Preferences

Chapter 3 presented a model of *coarse preferences*, and studied inference with coarse utility functions. We showed that, provided our model assumptions hold, we can achieve significant computational benefits under single user inference. We then proceeded to show how these benefits can be retained under decision problems with multiple users and user types. In doing so, we introduced the concept of a *user-set coarse mapping*, a coarse representation for a set of users or user types, as a sufficient and optimal way of representing all users' coarseness with a single latent space of categories.

In this chapter, we will propose a methodology for the online, active, learning of coarse preferences. A crucial part of achieving this is developing a procedure for learning the user's coarse mapping from an offline data set. We first propose Regression Tree Learning over each user's part of the corpus, but discover that this quickly converges to trivial solutions, exhibiting the worst of both cold-start problems. We then demonstrate that running Regression Tree Learning over our whole corpus can provide us directly

with the user-set coarse mapping, giving an efficient way of combatting both cold-start problems. The resulting procedure is applied to preference elicitation over a real-world online retailer's data set, significantly outperforming the state of the art, both in terms of runtime, and in terms of quality of recommendations. Furthermore, the hierarchical structure of our model allows us to optimise our recommendations for a secondary criterion (revenue), without reducing the quality of recommendations.

4.1 Introduction

The computational benefits stemming from performing inference with a coarse preferences model, by definition extend to the decisions of which query to present during *active learning*. Generalising across members of a *preferential equivalence class* will provide benefits in terms of computation time for query selection and the number of queries for converging to a good solution; the latter provided our assumptions over user behaviour hold. Further, as we will see in this chapter, mapping the solution space into a univariate latent space also significantly reduces the computation time for making an update over the preference model; in turn further accelerating query selection as a result of reduced time for query evaluation. Lastly, since decisions are optimised at the level of coarse classes, optimal queries and recommendations are defined as subsets of the original item space. This allows for a secondary round of optimisation prior to the final presentation of a query or recommendation, without loss of expected value of information or recommendation quality.

For any of the above benefits to materialise, a system first requires access to the user's latent space of categories. We will demonstrate that learning such a partition for each user independently is not robust, and presents a significant cold-start, item and user, problem. We will then propose a procedure for learning a partition of the solution space

from a corpus of user interactions offline, and prove that this partitioning represents the *user-set* coarse mapping.

The methods detailed below are relevant to any online learning of a coarse utility function, regardless of the querying policy, if any, such as active learning or bandit algorithms. However, we choose to study a typical preference elicitation scenario with a one-step look-ahead *Expected Value Of Information* (EVOI) computation in order to illustrate the increased computational benefits from coarse preferences during query selection. To ground our approach, we revisit the Mallzee clothing recommender system scenario, first introduced in Section 2.1.1.

4.2 Problem Description

As we saw in Chapter 2, the Mallzee scenario involves sequentially presenting a user with different items of clothing in an attempt to quickly gauge his or her preferred items from whether they accept or reject the recommendation, such that future recommendations have a greater chance of being accepted. We model this as a problem of preference elicitation with *value queries*. However, we assume noisy responses to compromise for the fact that user responses are constrained to fall on a predefined set of values.

The system represents items in a multivariate vector space X. Variables in this space could, e.g., refer to the type of clothing, its brand, price, or colour. At each time-step t the user is presented with an item $x_t \in Q_t \subseteq X$, where Q_t is a set of available items for recommendation at time t. Each recommendation x_t elicits a response $r_t \in R$, where R is the space of possible responses. The user evaluates an item $x \in X$ according to a utility function $u : X \to \mathbb{R}$ with $u \in U$ drawn from the space of possible utility functions U. Values $u(x)$, $\forall x \in X$, define how much the user prefers each item. In

other words, $u(x) > u(x') \Leftrightarrow x \succ x', \forall x, x' \in X$ (von Neumann and Morgenstern, 1953). We assume that the user's responses are truthful but noisy evaluations of the presented items, drawn from a normal distribution $P_r = P(r_t|x_t, u) = \mathcal{N}(r_t; u(x_t), \sigma)$ with mean equal to the utility of the queried item. The actual procedure by which users choose their response when constrained to a set of options is more convoluted, and this point could be handled more formally in future work.

We consider two alternative *optimality criteria* for the system: maximising the *user's utility* over a recommended item, and maximising the *revenue* of the system from that recommendation. We are particularly interested in the tradeoff between these two criteria, since they represent a conflict of interest between user and system. We define revenue as a constant percentage α of a recommended item's value multiplied by its rating by the user: $v(x, r) = \alpha \cdot r \cdot p(x)$, where $p(x)$ is the price of item x. The assumption is that accepted recommendations have a constant chance of translating into sales, and that the system receives a set commission on each sale. Maximising the user's utility is equivalent to maximising the user's rating of the recommended item, which is the focus of the majority of recommender system applications. Additional criteria are also often considered, such as *serendipity* and *diversity* (Andreadis *et al.*, 2016). While we could incorporate these criteria into the user's utility function, this goes beyond the scope of this thesis.

When our aim is to maximise the user's utility, the optimal decision w.r.t. u is $x^* = \text{argmax}_{x \in X}\ u(x)$, giving utility $u(x^*)$. When optimising for revenue, the optimal decision w.r.t. u is $x^* = \text{argmax}_{x \in X} \int v(x, r)\, P_r dr$, with expected revenue $\int v(x^*, r)\, P_r dr$. The next section assumes the former, so as to simplify presentation.

4.2.1 Preference elicitation

Given a finite and *manageable*, set of available items, we could assign one parameter for the utility of each item. [1] When this is not the case, then more structured representations of the space U are used, such as those making use of additive (Koller and Friedman, 2009) or generalised additive independence (Gonzales and Perny, 2004) of variables. The system does not, in general, have complete knowledge of u, but maintains a density b^t over the space U, indicating its current belief over the user's utility function at time step t, with b^0 representing its prior knowledge. If we denote the expected utility of an outcome x given density b^t over U as $EU(x,b^t)$ then the optimal decision is $x^* = \text{argmax}_{x\in X} EU(x,b^t)$. We denote by $MEU(b^t)$ the value of being in state b^t, assuming one is forced to make a decision: $MEU(b^t) = EU(x^*,b^t)$.

At each time step t the system can present the user with an item $x_t \in Q_t$, eliciting a response $r_t \in R$. The user's response can be used to update our belief over their utility function, in accordance with Bayes' rule. The (myopic) *expected value of information (EVOI)* of a query can be defined by considering the difference between $MEU(b^t)$ and the expectation (w.r.t. r_t) of $MEU(b^{t+1})$. A myopically optimal elicitation strategy involves asking queries with maximal EVOI at each time step Braziunas (2006). By definition of the EVOI, presenting the user with the query that maximises the EVOI guarantees, in expectation, that a recommendation immediately after the user's response would provide them with the maximum possible utility.

[1] We use *manageable* to refer to both memory-related performance and to the dynamics of the item population. Since this approach does not allow for generalising across items the set of items needs to be small and constant.

4.3 Coarse Preferences

As we saw in Chapter 3, given a utility function $u : X \to \mathbb{R}$ defining a coarse mapping $\phi : X \to C$, we can write the utility function

$$u^c : C \to \mathbb{R}, \tag{4.1}$$

with $u^c\big(\phi(x)\big) = u(x), \forall x \in X$.

4.3.1 Eliciting coarse preferences

If correct, the mapping ϕ from the space of items to that of categories allows us to select queries from the space of categories without loss of information. This reduces the cardinality of the query space from $|X|$ to $|C|$ and should speed up the convergence of the learning algorithm significantly when compared to approaches defined over the original outcome space. A further result is that, as we select for optimal queries over the space C, we are allowed a subset of X as optimal queries, in terms of expected value of information. The system can then freely select what to present from this set, according to external criteria.

We do not assume any correlation between the utility a user assigns to different categories (though a model of *neighbouring* categories could be an interesting future extension). this allows us to encode our belief over the user's utility for a category c with a uni-variate density $p\Big(u^c(c)\Big)$. Since we are working on a discrete domain C, we can represent our belief over the utility function as

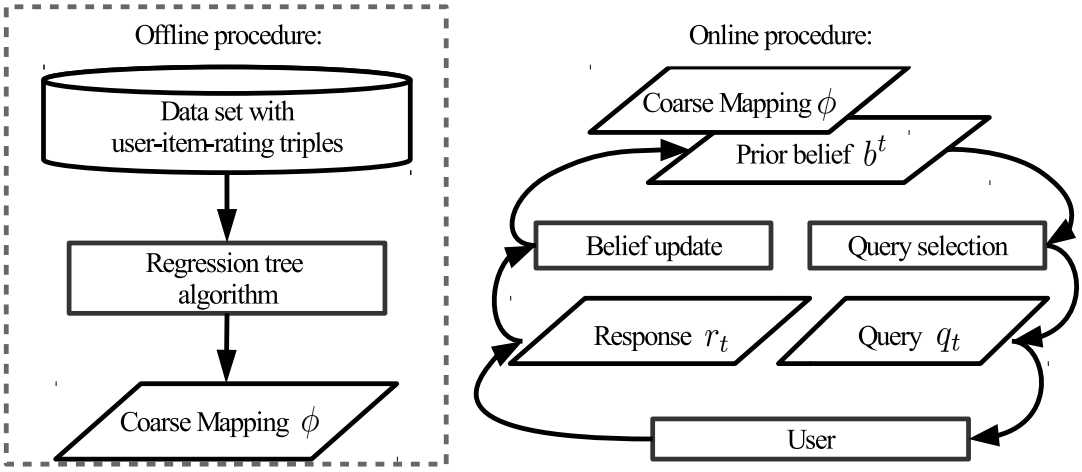


Figure 4.1: Overall methodology for the elicitation of coarse preferences. The offline procedure involves learning the coarse mapping ϕ from a corpus of data. The online preference elicitation procedure follows the normal procedure of query selection and belief updating to the user's response, but does so over the space of categories, as defined by ϕ.

$$b^t(u^c) = \prod_{c=1}^{|C|} p\Big(u^c(c)\Big). \tag{4.2}$$

Given a response r_t to a query $q_t = x_t$ at time step t, we can compute the new belief b^{t+1} according to Bayes' Rule. We write the posterior in closed form after each response as follows:

$$b^{t+1}(u^c|r_t) = \frac{P(r_t;q_t|u^c)\cdot b^t(u^c)}{\int P(r_t;q_t|u^c)\cdot b^t(u^c)\,\mathrm{d}u^c}. \tag{4.3}$$

To allow for Eq. 4.2 to be a closed form expression, we can assume that p is a Gaussian density with $u^c(c) \sim \mathcal{N}\Big(u^c(c); \mu(c), \sigma^2(c)\Big)$.

So far, we have discussed how to elicit coarse preferences given a coarse mapping ϕ. We next develop an offline methodology for learning this mapping from a corpus of user interactions in the form of a set of ratings of products from past users of the system.

4.4 Learning the Coarse Mapping

In order to utilise the procedure detailed in the previous section, we need to first identify the coarse categories across our population of users. Though every user i will have their own mapping $\phi_i : X \to C_i$, the mapping ϕ resulting from their intersection can be used to make decisions for every individual user with no loss of utility since, as follows from Definition 3.3.1, a user i will exhibit both coarse preferences ϕ and ϕ_i.

We assume access to a history of user interactions in the form of user-outcome-rating triples, and that this history is representative of future users. We need to partition the solution space such that all outcomes in each class can be mapped to the same utility with little or no loss in accuracy.

4.4.1 Learning single-user coarse mappings

Assuming user preferences are represented by a utility function, then learning a user's coarse mapping equates to producing a partitioning of the solution space into classes, such that membership to a class singularly predicts the value of a hidden regression variable, the utility. Assuming that there is some correlation between the proximity of

points in this space and their membership in the same class, this can be understood as a clustering problem. However, this fails to capture the dependency of class membership on the utility. Specifically, if we were given knowledge of the utility of different solutions, then the proximity of solutions becomes a non factor in determining class membership. That is, conditioned on the utility value of each item, each item's class membership is independent of their relative position in the solution space.

Despite the above, one could concede to use utility as another variable describing each item. However, it is unclear how one would weight this dimension against those of the item descriptors.

The above indicates that our partitioning problem is one of *supervised learning*, where the utility of each item, for each user, is the target regression variable, and the vector descriptions of items, in the original solution space, are the samples. A partitioning procedure which takes a target regression variable into account is Regression Tree Learning (RTL) (Breiman *et al.*, 1984).

The RTL algorithm receives a set of outcome-rating pairs as input, and outputs a decision tree with a continuous target variable; in this case, the utility. This regression tree defines a partitioning of the original outcome space by use of axis-parallel linear constraints, with each partition corresponding to a leaf node in the tree.

We can produce the space of categories by assigning a class label to each leaf node in the resulting regression tree, such that each preferential equivalence class is determined by following a sequence of tree-splits. However, applying RTL over an individual user's samples will only provide a guess over that user's coarse mapping, since there is an infinite number of trees that can represent any finite number of samples equally well, as determined by the split metric (Breiman *et al.*, 1984). The problem with this becomes apparent if we consider predictions for items that are outside of the user's experiences, and therefore not in our samples; predictions will change dramatically

based on where the user defines a boundary. Not only will this lead to a significant item cold start problem, but the model would remain an inaccurate representation of newly experienced items, until retraining occurs. Another problem with this approach, is that we have no way of learning mappings for new users, until a significant amount of behavioural data has been gathered. This leads to a significant user cold-start problem.

4.4.2 Learning the user-set coarse mapping

As we saw in 3.4.1, we can use the pairwise intersection of all users' coarse mappings for inference over any user, user type, or group of users. This user-set coarse mapping would solve our cold start problems by defining a valid latent space for all users and items, to the extent that these are represented by some users and items from the *entire* corpus of data. However, the intersection of mappings from a large number of users will quickly converge to a trivial mapping of 1 sample per class, especially since the users' mappings, as learnt from RTL, can vary significantly from their *true* mappings. We would prefer a procedure that is more robust to noise and outliers, producing coarse mappings with more overlay between users and better ability to generalise to new items.

One way of solving this problem would be to learn the user-set coarse mapping directly from all users' interaction data. This would allow us to utilise all of our samples together, achieving generalisation across users which, provided they are representative of future users, will effectively the user cold start problem. Moreover, by learning the mapping over all items in our data set, we effectively tackle the item cold start problem, to the extent that those items are representative of future items in our catalogue.

However, aiming to learn the user-set coarse mapping means we are no longer partitioning the space of solutions according to a user's evaluations. We could try to define a new target variable over the evaluations of all users. However, it is not clear

what the sample space for the RTL algorithm is. One natural consideration would be to use the original solution space, defining a function over the evaluation of all users for each item rated. For example, the utility mean of each item estimated across all users. However, the sparsity of our data set will not allow for this. Even so, it is not clear what metric would be able to represent the fact that, for each user, all items in a subspace of items are rated the same way, as the user-set coarse mapping defines.

Luckily, it can be shown that, if we run the RTL algorithm over all the corpus defined over the original space of solutions, with user evaluations as the target variable, and discarding any information on the user giving each rating, then the partitioning implied by the resulting tree can be an effective definition of the user-set coarse mapping (i.e. the coarse mapping that is consistent with all users). To understand why this is the case, we need to examine how the RTL algorithm works.

4.4.3 Regression tree learning for the user-set coarse mapping

A Regression Tree, essentially a Decision Tree where the target variable is a regression variable, sequentially splits a provided set of samples, in our case vector descriptions of clothing items, into subsets, such that membership in a specific subset is increasingly a better predictor of the target variable, in our case a rating (Breiman *et al.*, 1984). Regression Tree Learning (RTL) is an algorithmic procedure for learning such a tree. RTL chooses how to split at every node, according to a one-step look-ahead metric estimation. Since we have a target regression variable, we will use variance reduction, with the procedure aiming at the reduction of the sum of variance of the utility across subsets after each split Breiman *et al.* (1984).

Consider a set of users I, the space of user-set coarse categories C, and a belief over each user's $i \in I$ utility function b_i. The probability that the expectation over the

sum (and average) of the utilities of randomly, independently, sampled users for two randomly sampled items mapping to different categories $c_1, c_2 \in C$, conditioned on that mapping are equal is zero. This holds for continuous utiliy values and, in the limit of users, for discrete utility values. This follows from the fact that the probability of two samples drawn from continuous distributions being equal is 0, and the application of the Central Limit Theorem (Billingsley, 2008) for the discrete case.

The implication is that we can distinguish coarse categories by comparing the expected value of the utility of their members across users. What remains to be shown is that RTL will identify splits that separate samples in this manner.

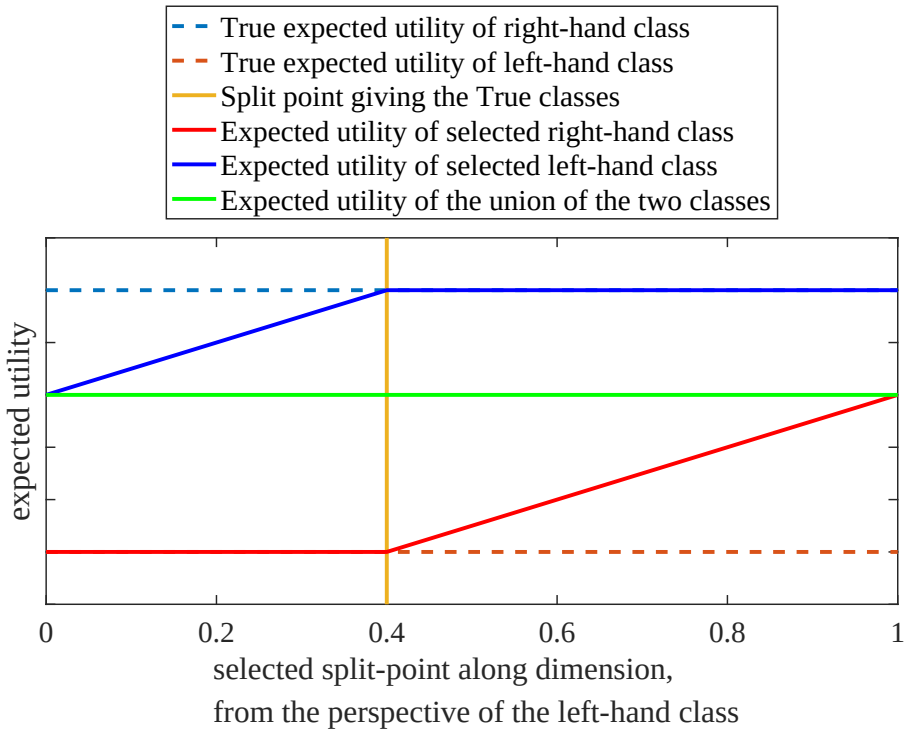


Figure 4.2: Example of how the expected utility of sampled points in two neighbouring classes changes, as we shift the split-point between the two.

Consider an arbitrary node down the regression tree, as it is still being constructed during RTL, and a variable x_j with values in X_j, such that there exists a point in X_j that would split the subspace into two sets of preferential equivalence classes, according to

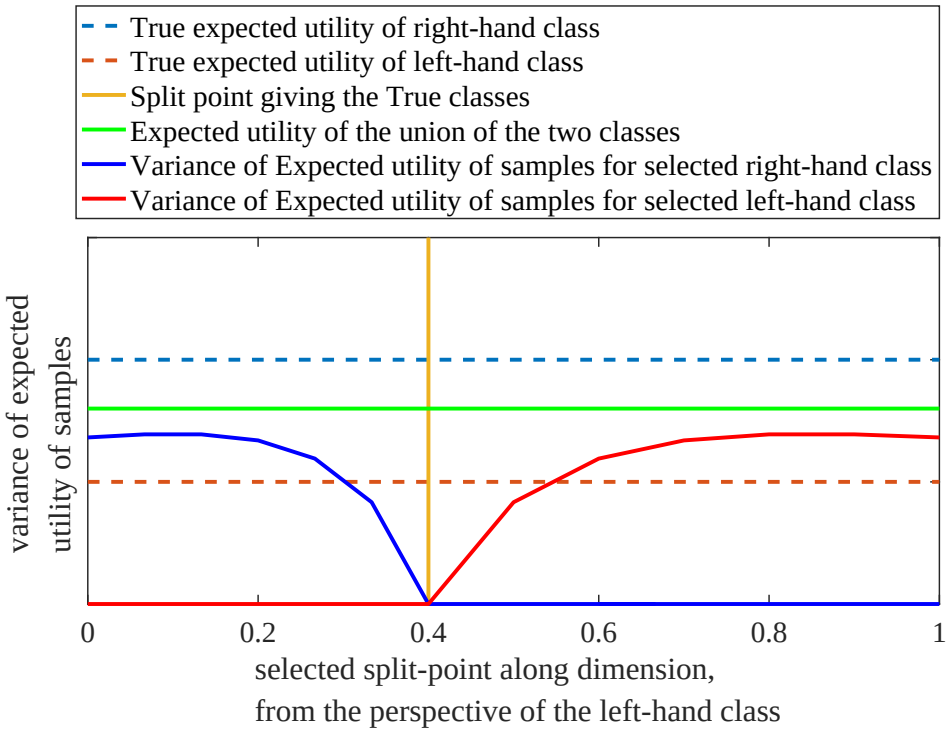


Figure 4.3: Example of how the variance of the expected utility of sampled points in two neighbouring classes changes, as we shift the split-point between the two.

the user-set coarse mapping. Will RTL be able to correctly identify this split point? Figure 4.2 illustrates this decision problem. The x-axis represents the space of possible splits, whereas the y-axis indicates the expected utility of a sample drawn from a random user, and at a uniformly random point within the space on the left of the split point; and is therefore the expected value of a sample drawn from the category that would have been defined by that split. As long as the split point is chosen to be on the right of the actual border between the two equivalence classes, the expectation of the utility of a sample remains the same. Beyond that point, it linearly increases until it reaches the area-weighted average of the expected utilities of the two categories. In order to prove that RTL will identify this split point, we need to show that this is also a minimum for the expected sum of variance of samples in the two categories. It is important to note that the graphs in Figures 4.2 and 4.3 plot the mean and variance,

respectively, of the expected utility. This is equivalent to assuming all items in a class have been evaluated to have a utility equal to the mean utility across items in the class.

Figure 4.3 plots the change in variance of the expectations of samples drawn from a set of categories defined to be on the left of a split-point as defined by the x-axis, and in relation to the *true* split point. As this is the variance of the expected utility, rather than the expectation of the variance, it remains at 0 until the split-point surpasses the true split point. After this, it increases up to a value, before decreasing asymptotically towards 0, when the size of the right category is significantly larger than that of the left. Since the other category expresses a similar behaviour, the sum of variances of the expected utility will indeed minimise at the true split-point.

Unfortunately, this result does not guarantee that adding points from a neighbouring category will not decrease overall variance, but it is an indication that, as long as user's evaluations don't deviate significantly from the average evaluation within the category, RTL will find a good representation for the user-set coarse mapping.

If users were to rate similar items, then we could instead run RTL over the mean evaluation of each item across users. Provided enough samples, this would guarantee the behaviour we require. However, such a data set is not always available, as is the case with our experiments in the next section. Identifying a metric that will guarantee that RTL convergences to the true classes remains an open question.

4.5 Experiments

We now describe our experimental procedure for evaluating our proposed model of *coarse preferences*, including a comparison against a state-of-the-art utility-based model by Guo and Sanner (2010).

4.5.1 Data set

We evaluate on a real-world data set generated from the interactions of users on Mallzee, a smart-phone clothing retail application. Users are identified solely by their *id*. Interactions are represented as binary responses (swiping left or right) to product recommendations. For each recommended item, we are provided with assignments to a set of 8 parameters (each variable's domain cardinality is given in parentheses), specifically: current price, discount from original price, currency (4), type of clothing (22), intended gender (5), whether it is in stock (2), brand (139), and colour (16). Out of these, *current price* and *discount from original price* are continuous, while all others are categorical variables. The data set consists of 200 users in total, each one rating a different set of 500 products, for a total of 100'000 data-points. We split the data into a *training* and *test* set, each comprising of the responses of 100 users (Andreadis, 2016).

4.5.2 Procedure

We set an uninformative prior for each method, with an expectation of 0.5 utility spread equally across variables. The experiments are executed in two phases. First we learn the coarse mapping ϕ from our history of user interactions as stored in the *training* set. Then we run a series of 100 experiments with the *test* set acting as the query and recommendation space. We run one preference elicitation experiment instance for each of the users in the test set, for each algorithm and optimality criterion: expected user utility and system profit. For all methods, and at each time step t, we uniformly select 20 from the available queries for evaluation to comprise Q_t, and motivated by computational reasons. Such a constraint does reduce the informativeness of queries but represents real-world constraints, in terms of computational cost and availability of items, though the specific number has been selected arbitrarily. We compute the EVOI (see Section 2.5.11) for each query and select the maximising one to present to the user

for evaluation. After receiving the user's response (as stored in the data set) we update our belief over the user's preferences. We evaluate the quality of our current belief state by hypothesizing an optimal recommendation from our available set of items, and noting the user's response from the data set. Under the user utility criterion, and when there are multiple items with the same expected evaluation, we select the one that would generate the most profit.

We compare our coarse preference elicitation procedure to the state of the art of AI-decomposed utility function elicitation procedure, as adapted from Guo and Sanner (2010) . To account for ratings instead of pairwise comparisons the benchmark is changed to make use of value queries. The query selection procedure makes use of the same Bayesian updating scheme as in the original work, and as presented in this chapter. In order to do so, we have to discretise the continuous product descriptors: *current price* and *discount from original price*. In this step, the belief takes the form:

$$b^t = \prod_{d=1}^{D} \prod_{i=1}^{|x_d|} \mathcal{N}\Big(u^d(d_i); \mu(d_i), \sigma^2(d_i)\Big), \tag{4.4}$$

where D is the dimensionality of the item representations, $|x_d|$ is the size of the corresponding discrete assignment space, and d_i is an assignment to variable d.

In order to learn the coarse mapping from the space of item descriptors to that of categories, we first transform all categorical variables into sets of binary variables. In order to determine the hyperparameters for the regression tree algorithm we randomly select a history of user interactions and run the offline clustering procedure for different values for the *maximum tree height* and *minimum number of nodes per leaf-node* hyperparameters, in increments of 1 and 50 respectively. The Regression Tree algorithm outputs an expected utility value at each leaf node. We round each prediction to its nearest integer value in order to get an understanding of how well the Decision Tree predicts the *training* data. We then chose the configuration which

resulted in maximum precision when considering the products to which the users responded positively. During the experiments we then run a *regression tree* procedure as explained in Section 4.4, using these learned hyperparameters. To put these results into context, we perform least-squares linear regression on the same training set but without discretising the continuous variables, and round it as described above. We expect that this will give indication of how well the benchmark will perform.

Each leaf-node in this tree represents a category. In order to be able to generalise to the test data, we wanted to keep the cardinality of the latent category space relatively small compared to the available product space, focusing more on achieving higher precision, rather than recall, when considering the products to which users' responded positively. This is because making a good recommendation relies on our ability to identify a preferred item, rather than locating as many good items as possible.

To evaluate our performance for the criterion of user utility we plot the average normalised loss in user utility. This measures, at each time step, the normalised distance of the value assigned by the user to their recommended product, from the maximum evaluation that could have been achieved, given the available products for recommendation. We also examine our performance in terms of system profit. For this purpose, we plot the normalised loss in profit, computed considering the value of the most expensive item available as the best possible profit achievable. This metric is agnostic to whether the user had swiped right or not on that item in the data set.

4.5.3 Results

Optimising for the regression tree hyperparameters resulted in setting a maximum height of 10, and a minimum number of 500 training samples per node. This in turn resulted in a space owith cardinality $|C|$ between 23 to 45 categories across experiment

runs, with a mean of 33.2 and a standard deviation of 5.35 categories. Considering the products to which the user responded positively, we achieve an average precision of 0.97 and average recall of 0.70 on the training data. The least-squares linear regression procedure results in an average precision of 0.81 and average recall of 0.21. Figure 4.4 shows part of a learned decision tree, with two examples of categories. The first one shows users grouping all products not displayed in British pounds into a single category. The second one refers to all non female-specific shirts, with a defined colour, and that are displayed in British pounds. These descriptions are easy to understand and put into context, and can act as an additional tool for system designers.

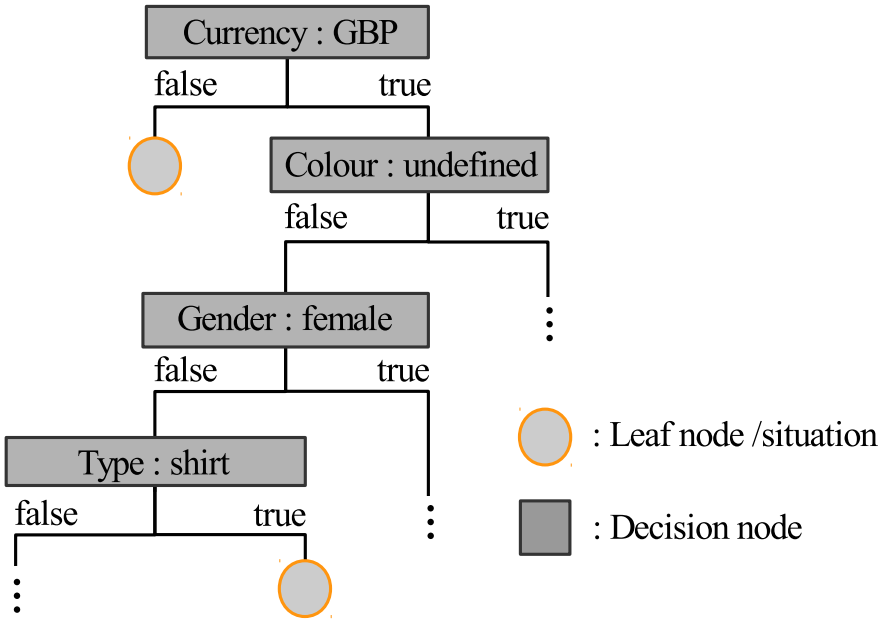


Figure 4.4: Part of a decision tree learned from the history of user interactions. Two leaf nodes are shown, each one identifying a category in the space C. This example tree had a height of 10.

Figure 4.5 plots the normalised loss of utility for the coarse preferences algorithm and the sum of conditional Gaussians benchmark, as adapted from Guo and Sanner (2010). 'Time-step' refers to the number of queries presented so far. The shaded areas cover ± 2 standard deviations, showing 95.45% confidence intervals. The lines

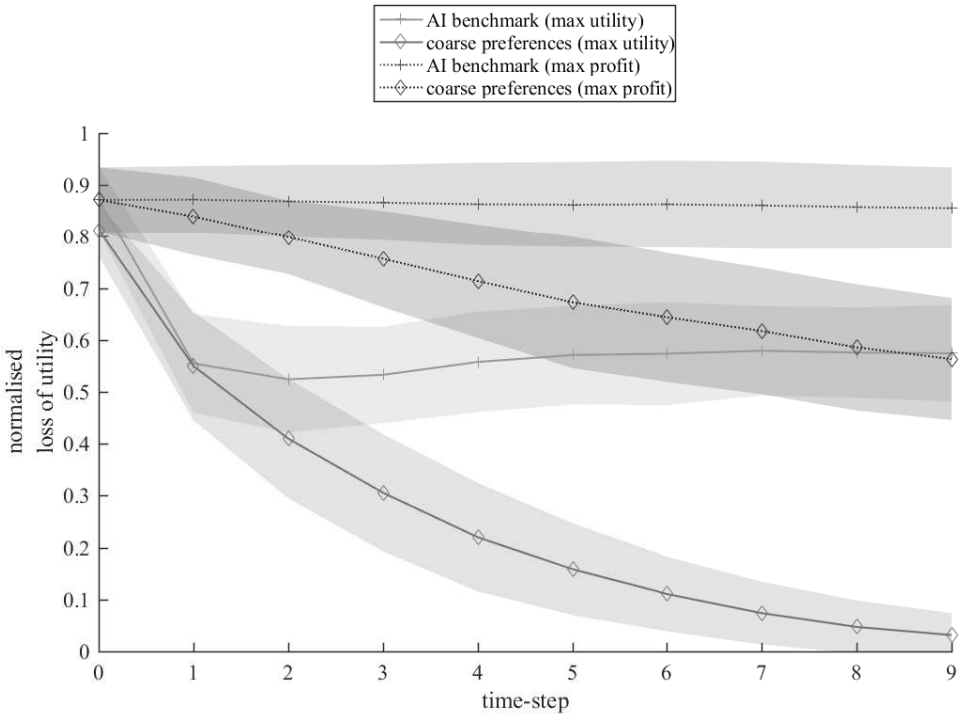


Figure 4.5: Average normalised loss of utility across 100 experiment instances from the user's response to our recommendation, averaged across 100 experiment runs.

labelled 'AI benchmark' present the results for the benchmark as run with each optimality criterion. Correspondingly, lines labelled "coarse preferences" present the results for our algorithm. The performance of our procedure is indicative of its ability to better trade-off between generality and accuracy for this data set. Querying any point in an equivalence class informs us of the user's preference for all its members. To the extent that this mapping is accurate, we achieve a significant reduction in the dimensionality of the solution space, by going from a space of 8 parameters with 500 samples (the space is combinatorially large but constrained by the number of available samples) to a space of 1 variable with a maximum size of $|C|$. This in turn translates to faster convergence to an accurate representation of the user's preferences. As expected, procedures optimising for user utility achieve

better performance in this metric. However, even when optimising for user utility, the AI benchmark quickly converges to a suboptimal representation of user preferences. Crucially, when optimising for system profit, the users experience no improvement in their recommendations as time progresses. This results from the additive independent assumption being a bad fit to the specific problem combined with restrictions in its ability to query the user. These restrictions stem from the fact that not all of the solution space is available for querying with or, since we are using the same space, recommending to the user. In fact only 500 points out of a combinatorially large set are available. This problem is enhanced by further restricting the space to a randomly selected set of 20 available items. The coarse preferences approach is much more effective at generalising from a small number of samples and therefore does not suffer from this effect.

Figure 4.6 displays the gradual improvement in the effective monetary value of recommendations, in terms of average normalised loss. The gains in profits in comparison to the benchmark are significant, which we attribute to both our model being a better fit to the problem and our ability to select from a space of, from the user's perspective, equivalent solutions. One might expect that this graph would give a complementary image to that of Figure 4.5. However, both coarse preferences approaches significantly outperformed the AI benchmark. Moreover, running our procedure for optimal user utility outperformed the same procedure over system profit. We believe this to be the result of an overoptimistic prior over the utility of each category, leading the algorithm to take risks for profit that don't always pay off. Focusing on the benchmark, we note that the improvement for the first two time-steps when run for user utility does not translate to increased profits. Further, when optimising for profit the inability to accurately represent users' preferences translates to poor improvement in profit.

Figure 4.7 presents the time taken during a single step of preference elicitation,

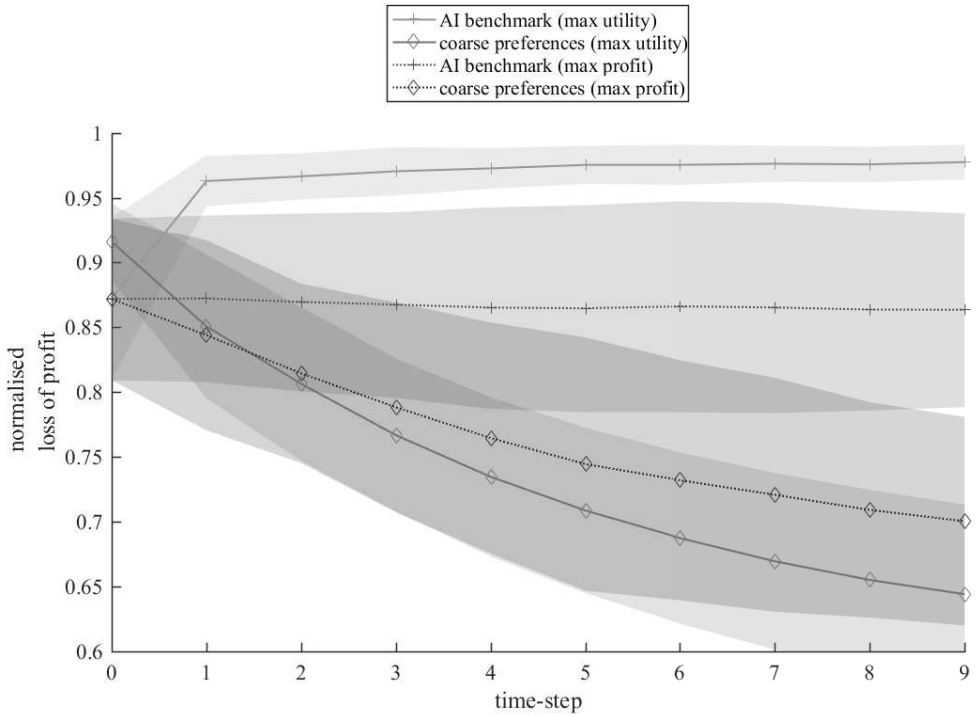


Figure 4.6: Average normalised loss of profit across 100 experiment instances from the user's response to our recommendation, averaged across 100 experiment runs.

averaged across all runs, experiments, and time steps. Error bars represent ± 2 standard deviations. The presented values include time taken for query evaluation and selection, as well as the belief update after the user's response. Regardless of the optimality criterion, our approach requires about half as much time as the benchmark, which can be a significant advantage in online applications.

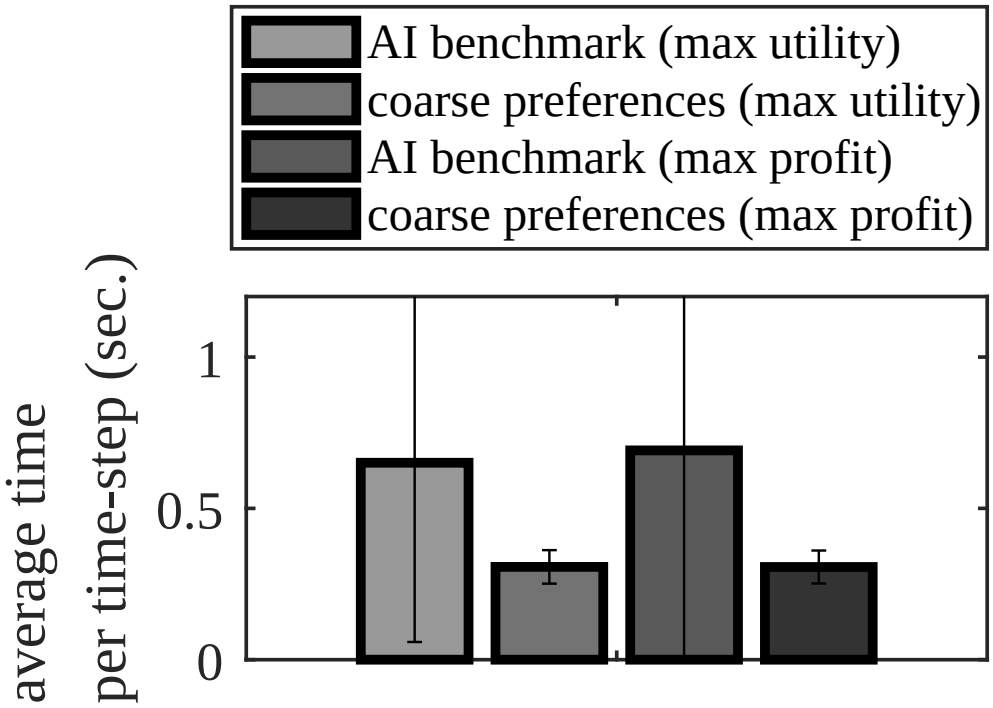


Figure 4.7: Average computational time in seconds for each preference elicitation model and optimality criterion.

4.6 Discussion

The main incentive for adopting our approach is that it allows for using a different decomposition of utility functions, conditioned on different assumptions from those of variable-based decomposition models. As such, it expands the set of problems that can be sufficiently modelled by such approaches. Preference elicitation with coarse preferences is much faster online while their structure allows for optimising for secondary criteria, such as profit or environmental impact, without having to sacrifice user utility.

However, these benefits are hard to justify if user behaviour does not conform to the assumption of coarseness, that is that there are significant subspaces of the solution

space in which users are indifferent between different solutions. Before deploying any utility function decomposition the system designer needs to verify whether its underlying assumptions approximate real user behaviour.

Lastly, in its current iteration our model is dependent on having access to a history of user interactions, with the assumption that those will be representative of future user behaviour. This could potentially be circumvented by learning the coarseness online.

There exist a number of approaches for handling the exploration-exploitation trade-off, e.g. bandit algorithms (Busa-Fekete and Hüllermeier, 2014) and partially observable Markov decision processes (Boutilier, 2002). The issue of computing these policies is orthogonal to that of exploiting the structure of the utility function representation.

4.7 Conclusion

We proposed a new approach to utility function decomposition termed *coarse preferences* which models user behaviour that is consistent with them evaluating alternatives based on which category each one falls into. Our approach is orthogonal to that of variable-based decompositions such as additive independence in that it can be used in combination with them while also being based on different assumptions. We demonstrate that there exist real-world problems where the coarseness assumption is a better fit than additive independence to user behaviour. This allows for a significant increase in recommendation quality while also taking advantage of reduced computational time; a benefit that scales with the number of variables. The magnitude of the effect our procedure had suggests that it is worthy of consideration in recommender system applications, particularly those that involve users rating sequentially presented items. Furthermore, our model is the first, to our knowledge, approach that allows for

optimising for secondary criteria, such as profit, while guaranteeing to optimise for the user's utility. In fact, we do better at both criteria against a state-of-the-art benchmark.

Having developed a coarse preferences online learning procedure, and provided evidence for its applicability to typical recommender systems, the next chapter will examine how this methodology enables otherwise computationally prohibitive inference and online learning computations to take place. For this, we will revisit the MovieLens coordination through set recommendation scenario, first described in Section 2.1.1.

Chapter 5

Coordination through Set Recommendation: a Case Study for Coarse Preferences

The previous two chapters introduced a theory of coarse preferences, proposed a preference elicitation methodology, and applied it to a real-world recommendation problem. Though the experiments of Chapter 4 showed significant benefits in terms of computational time and recommendation quality, the decision scenario was not such that the full potential of the methodology could be explored. This potential manifests in problems of increased computational complexity, such as the one that is analysed in this chapter. We examine the problem of iteratively recommending a set of items to each in a set of users, where the outcome experienced by each is contingent on the choices of all other users, but where these choices are made without any knowledge or expectation over other users' choices, and without the ability to communicate with each other. The procedure for coordinating their choices is computationally expensive, and a number of techniques are introduced for managing this issue. Crucially, we demonstrate how

the reduced complexity of the coarse preferences model allows for exploring a larger set of options, while also generalising user feedback for more efficient learning of user preferences. We demonstrate benefits at scale, in terms of item description length during learning, and in terms of solution domain size during inference. This results in a significant increase in recommendation quality, providing evidence for the enabling of otherwise intractable computational procedures.

5.1 Introduction

In Andreadis *et al.* (2016), we first examined the problem of *Coordination through Set Recommendation*. In this, users are each to be recommended a personalised set of alternatives, from which each is expected to select an item, in accordance with their models of preference and selection behaviour. This was motivated as a *sharing economy application*, which is a domain of multi-agent resource allocation and coalition formation. In these applications, users act as producers and consumers of resources, aiming to find peers to share the resources with, while a platform supports them during peer discovery and resource sharing (Andreadis *et al.*, 2016).

A crucial aspect of these systems, is that the outcome for the system and each individual user is dependent on the choices of all users. In the case of coordination through set recommendation, users do not have a chance to coordinate their actions and do not think strategically, selecting only according to their evaluations of items in their recommended set. The problem we focused on was that of selecting the sets of recommended alternatives to present to the users, with limited knowledge of their collective behaviour, such that, after their selections, one or more functions are optimised.

Where does our interest in this application arise from? Typically, such problems are

solved by the system allocating solutions to users; approaching it as a problem of matching between users, or between users and resources (Gusfield and Irving, 1989; Mourad Baïou, 2002; Dickerson *et al.*, 2012); users have no control over how the allocation is performed. Some, more flexible, approaches sequentially present users with solutions they can choose to reject or accept (Gale and Shapley, 1962; Aziz *et al.*, 2015), while others assume a known preference profile for the users (Chung, 2000). All of these approaches however treat the problem as one of a centrally-controlled allocation of solutions to users, where the possibility of the users preferring not to be allocated at all, as opposed to accepting a substandard allocation, is not taken into consideration.

The freedom of the user to abandon the platform, and our uncertainty over their preferences, are good motivation for approaching sharing economy applications as recommendation problems. By providing users with a choice, we both learn more of their preferences and improve our chances of them identifying an acceptable solution. Inferring the correct sets of recommendations however, especially given a large number of alternatives and users, is computationally hard (Andreadis *et al.*, 2016).

In Andreadis *et al.* (2016), a methodology was presented for the coordination of user collectives, in the absence of communication among agents. The proposed approach outperformed that of directly allocating to users the solution with the highest expected system utility, demonstrating that we can allow users to have a choice in their alternatives, at no loss to the system. The method also allowed for the adaptive trade-off between system-level utility and fairness of final allocation. This chapter expands on that paper by also addressing the problem of learning users' preferences online, which largely motivated the original work. As discussed in Chapter 3, though utility models allow for this, the computational cost associated with the online model update can be prohibitive. In this chapter, we will demonstrate how coarse preferences can be utilised in computationally complex sequential recommendation problems, achieving benefits

at scale during the online preference model update and the optimisation for selecting which recommendations to make. For this purpose, we will make use of the MovieLens 20M Dataset (Harper and Konstan, 2015), and propose a movie streaming service scenario, where users join *rooms* in order to watch a movie together. Users might be motivated by the social interaction or the provision of some limited service, such as live director commentary and discussion. Modelling restrictions over the provision of such services, such as the cost of inviting the movie director, will mean that the optimisation is constrained, in this case, by a minimum number of participants.

The remainder of this chapter is organised as follows. In Section 5.2, we provide a formal description of the allocation problem that characterises sharing applications, and introduce models of user selection, which we then expand on to consider the sequential nature of our problem and online learning. Section 5.2.2 focusses on the problem of learning the users' preference functions as they interact with the system. Section 5.3 presents a detailed description and formulation of the mixed integer linear programs used in our optimisation framework, as adapted to the case study. The experimental evaluation and obtained results are described in Section 5.4. Section 5.5 concludes, summarising our results and indicating possible future directions.

5.2 Problem Description

We first formalise the resource allocation problem, and then introduce set recommendation and the corresponding learning procedure.

Consider a set of items $J = \{1, \ldots, |J|\}$. [1] We assume a vector space $\mathbf{X}$ such that each item $j \in J$ maps to a vector representation $x_j \in \mathbf{X}$. Let $I = \{1, \ldots, |I|\}$ be the set of

[1] In the original work, a task $j \in J$ was associated with one and only one user who *owned* the task, for example the owner of the resource that will be shared, or whoever initiated the sharing task (Andreadis *et al.*, 2016).

users [2] In (Andreadis *et al.*, 2016) items were restricted to a subset of users though we will not consider such constraints in this work.

Let $\mathbf{a} = \{a_{1,1}, \ldots, a_{1,|J|}, \ldots, a_{|I|,1}, \ldots a_{|I|,|J|}\} \in \mathbf{A}$ define which user is allocaetd to an item, i.e., $\mathbf{a}$ is an allocation of users to items, where $\mathbf{A}$ is the set of allocations. In particular, if i is allocated to item j then $a_{i,j} = 1$, otherwise $a_{i,j} = 0$.

The preferences of each user $i \in I$ over possible allocations are represented by a utility function $\upsilon_i : \mathbf{A} \to \mathbb{R}$ which provides a complete ranking over potential allocations $\mathbf{a} \in \mathbf{A}$. We will assume that υ_i, $\forall i \in I$, is consistent with users that maintain a utility function $u_i : \mathbf{X} \to \mathbb{R}$ over successfully allocated items, and that experience a utility of 0 from not successfully being allocated to an item. We further consider a system-level criterion, *social welfare maximisation*, $V_s : \mathbf{A} \to \mathbb{R}$, specifically $V_s(\mathbf{a}) = \sum_I \upsilon_i(\mathbf{a})$, which will be what the system will aim to maximise. In Andreadis *et al.* (2016), where the problem of coordination through set recommendation was first proposed, we considered further criteria related to *fairness*. Such an analysis goes beyond the focus of this chapter.

5.2.1 Set recommendation

Viewed as an allocation problem, it would be enough to find $\text{argmax}_{\mathbf{a}\in\mathbf{A}} V_s(\mathbf{a}) = \sum_I \upsilon_i(\mathbf{a})$. However, we are interested in providing users with sets of recommended items, therefore enabling choice and learning from user interactions with the recommended set. Further, we will consider the sequential nature of such interactions, not necessarily with the same exact users, but in the form of on-line sessions. Specifically, assume that sessions occur at distinct time-steps $t > 0$, and that at each such session,

[2] Users who do not own a task, in (Andreadis *et al.*, 2016).

the available set of items is $J_t \subseteq J$ and that the participating users are $I_t \in I$. For simplicity of representation, we will still write the space of allocations as $\mathbf{A}$, though we will demand $a_{i,j} = 0, \forall i \notin I_t$, and $a_{i,j} = 0, \forall j \notin J_t$.

The aim of the application is, for each session, to present users with sets of recommendations such that their combined behaviour in expectation maximises V_s. We write R_i^t to refer to the recommended set of items to user i in session t; though we will omit the session index whenever it is not required by the context. All recommendation sets are drawn from the set of users J, such that $R_i^t \in \mathbf{X}^H$, where $H = |R_i^t|, \forall t > 0, i \in I_t$. Each user i then independently selects an item r_i^t in R_i^t, according to their utility function u_i and the user response model. Each user $i \in I_t$ selects an allocation from the set R_i^t of recommended solutions independently and without direct coordination with other users, according to a *user response model*. Viappiani and Boutilier (2010) consider three user response models:

- *noiseless* response model: each user i acts deterministically and selects the solution $r_i^{t*} = \text{argmax}_{r_i^t \in R_i^t} u(r_i^t)$.

- *constant noise* response model: each user selects solution $r_i^{t*} = \text{argmax}_{r_i^t \in R_i^t} u(r_i^t)$ with probability α and any other solution $r_i^t \in R_i^t - \{r_i^{t*}\}$ with probability $\beta = (1-\alpha)/(H-1)$.

- *logit* response model: each user selects an allocation from the set R_i^t proportionally to its utility value: $u_i(r_i^t) \,/\, \sum_{r_i^{t\prime} \in R_i^t} u_i(r_i^{t\prime}), \forall i \in I_t$.

We examine the differences in optimisation behaviour for different assumed user response models in Andreadis *et al.* (2016).

At each session, post-selection, user choices are compared and the selected items are

allocated to their respective users only if certain criteria are met. For the scenario we are considering, users are successfully allocated an item they have selected if the number of users that have selected that item in this session is above a threshold of minimum participation. It is important to note that it is users that actually make the final selection, and then constraints are checked. Whether that selection is the sponsored one is not relevant to the final allocation; it is simply used for producing the recommendation set.

5.2.2 Learning from user interactions

After having been presented with a recommendation set R_i^t, user i selects item $r_i^t \in R_i^t$ in accordance with their preferences and selection behaviour. By considering the presentation of this recommendation set as a *choice query* (Viappiani and Boutilier, 2010), where a user is asked to indicate their preferred option in a set of options, we can interpret their selection as evidence that $r_i^t \succeq r_i^{t\prime}, \forall r_i^{t\prime} \in R_i^t$.

Consider that the system only has partial knowledge of user i's preferences at time t (before the beginning of a session $t+1$ and after session t), as represented by their utility function u_i. Further, consider that this knowledge is represented by a belief function over the space of utility functions U, in the form of a probability density function $b_t(u_i)$. After a user's selection, we can write the Bayes update rule as:

$$b_{t+1}(u_i|r_i^t) = \frac{b_t(u_i) \cdot P(r_i^t; R_i^t|u_i)}{\int b_t(u_i) \cdot P(r_i^t; R_i^t|u_i)\, \mathrm{d}u_i}, \tag{5.1}$$

where we assume access to a prior distribution $b_0(u_i)$, and a user response model $P(r_i^t; R_i^t|u_i)$. The belief models used in our experiments for the baseline and our model

were described in Sections 4.3.1, and 5.4. We have assumed a constant noise response model, which we can more explicitly write as:

$$P(r_i^t; R_i^t | u_i) = \begin{cases} \alpha, & \text{if } u_i(r_i^t) > u_i(r_i^t), \forall r_i^{t\prime} \neq r_i^t \in R_i^t \\ \beta = (1-\alpha)/(H-1), & \text{otherwise.} \end{cases} \tag{5.2}$$

Though this formula does not account for ties between the utility of different items, this will not be of consideration in our experiments, since the taxation scheme we use does not allow for ties.

5.3 Optimisation Problem Formulation

The optimisation problem described in the previous section can be approached as a resource allocation problem, in which users are implicitly assumed to be compliant with any solution proposed to them, and are therefore not afforded any alternatives. The results of such an approach are constrained to that of a matching between users and items/resources. Consequently, there is no consideration of the inherent uncertainty in user behaviour, or the fact that users could simply refuse to participate in systems that do not satisfy their needs (i.e. are individually rational). A system that realistically addresses human diversity and the uncertainty in human behaviour requires an explicit representation of user preferences and their responses to different decision scenarios. Furthermore, the decision scenario needs to be formulated so that it allows for the recommendation of solutions to users, while accounting for their possible deviations from expected behaviour.

In order to help the system in the process of coordinating users' selections, we introduce a *taxation* mechanism, so as to influence user selection behaviour by artificially modifying the utility they have for the recommended solutions, i.e. by modifying their preferences. Effectively, taxation allows the system to impose a penalty on allocations users are better off not selecting. Generally, the tax imposed is different for each user and for each allocation, and must guarantee that users still have multiple options (e.g. the system cannot impose an infinite tax). The taxation mechanism is developed in Section 5.2. The taxation scheme affects users' evaluations of items in the recommendation set, and, therefore, their selection behaviour. The aim is to have some control over what items are selected by users, without prohibiting deviation from the optimal solution for the system (which would be the same as a direct allocation).

A problem here is that taxation changes the value of a solution, since it is part of the users', and therefore also the system's, utility function. We are not trying to guarantee a specific solution. Quite the contrary, we are trying to allow the users freedom in their choice, acknowledging that our knowledge of their preferences is incomplete. Since the tax is part of the utility function, it acts as an additional parameter to the recommended item specifications.

In order to handle this problem of computing an optimal set recommendation, we construct the recommendation set by executing two consecutive Mixed Integer Linear Programs (MILPs) Klotz and Newman (2013). The first of these identifies the allocation that maximises Social Welfare, given certain constraints. This allocation will be referred to as the *sponsored allocation*, while assignments to users specified by it will be referred to as *sponsored solutions*. The second MILP generates the rest $H-1$ recommendations for all users, taxed, where appropriate, so as to promote the sponsored allocation. The presented methodology is both an adaptation of and an extension to the work in (Andreadis *et al.*, 2016). Specifically, available items are

not conditioned on the participating users, which simplifies the system. The non-consideration of criteria beyond that of Social Welfare also simplifies the problem, allowing us to merge the first two MILP as presented in that work. Lastly, this chapter abandons the sequential computation of non-sponsored solutions for a more effective batch computation, after having discovered that the previous sequential approach over-taxes all but the first two solutions computed.

When computing the recommendation sets, we will only consider constraints related to user behaviour, such as the probability of a user selecting a sponsored solution, and on the minimum number of users that are required for a viewing to occur. Note that in practise users may have other requirements that the system should satisfy. For example, they may have constraints regarding the characteristics of users they are willing to share an item with. Such constraints were presented in the original work, but will not be considered in the MovieLens scenario.

The assumption is that users are boundedly rational (Simon, 1972; Tversky and Kahneman, 1981; Kahneman and Tversky, 1984; Gigerenzer and Selten, 2003), but they do not have access to any information on other users' preferences or recommendation sets, and do not retain knowledge of previous interactions with the system, and can therefore not select items strategically (Chevaleyre *et al.*, 2006). Instead, items are selected based on only their own preferences and the restrictions on their rationality imposed by the response model. Therefore, and since users can only receive the provided items through our system, envy-freenes (Chevaleyre *et al.*, 2006) is not a useful consideration for our problem. The original work in Andreadis *et al.* (2016) does use envy-freeness as a metric of 'fairness' however, and future work could consider users with access to some information on other users' behaviour, such as some measure of popularity of the items presented, or that adapt their selections based on the observed outcomes from previous interactions with recommendation sets. If we represent the lack of information over the probability of a selected item actually being allocated as

the user believing that each selection has the same probability p of actually being allocated post selection, then the users' optimal choice is to select truthfully, according to their utility function. However, this selection will be noisy (as represented by the response model) representing boundedly rational users.

It is worth noting that our allocation mechanism is individually rational (Chevaleyre *et al.*, 2006), in the sense that a user can only benefit from deciding to participate and select an item from the recommended set. Either the user will be rewarded with that item, which we have assumed have a non-negative utility, or the user will not be allocated, which is the equivalent of not participating in the negotiation.

Although we have designed our problem in a way that typical mechanism design criteria are not relevant, we do recognise that the problem is a special case of multi-agent resource allocation. However, it is unique in that the allocation mechanism is a desirable constraint, around which the optimisation procedure has been built.

The next subsections present the details of the Mixed Integer Linear Programs (MILPs) that compose the framework described above.

5.3.1 Maximising social welfare

The MILP presented in this section is used in the first step of our framework and aims to compute the maximum system utility achievable without violating any requirements. The resulting allocation will be the *sponsored allocation*, defining the sponsored solution for each user. We refer to this system as $MILP^{first}$.

The input parameters to the model are:

- An arbitrary very large positive number (enough to out-scale other values in the

program) M - a typical tool used in Mixed Integer Linear Programming Klotz and Newman (2013);

- An artificial lower bound on the number of users that have to be recommended the same movie in an allocation, for that allocation to be accepted by the program m - for the purposes of our experiments, we set this to be $m = c + c * (1 - P_c)$ rounded up. This was estimated, by Matlab simulations, to have around 50% users allocated to an item, when recommendation sets were uniformly randomly generated;

- The expected utility of each user for each movie, written with slight abuse of notation as: $u_i(x_j), \forall i \in I_t$ and $j \in J_t$.

The variables for $MILP^{first}$ are:

- An indicator for whether item j is allocated to user i: $a(i,j) \in \{0,1\}, \forall i \in I_t$ and $j \in J_t$;

- An artificial variable for whether an item j is being allocated: $k(j) \in \{0,1\}, \forall j \in J_t$.

The objective function is the maximisation of *utilitarian* Social Welfare (Chevaleyre *et al.*, 2006), i.e. the sum of all individual user's (expected) utilities:

$$\max_{a(i,j), \forall i \in I_t and j \in J_t} \sum_{i \in I_t} \sum_{j \in J_t} u_i(x_j) \cdot a(i,j) \tag{5.3}$$

We require two additional sets of constraints, beyond the variable space definitions. The first one is a constraint on the minimum number of users that need to be allocated

a movie for it to be allocated at all, along with the necessary constraints for setting $k(j)$ correctly:

$$\sum_{i \in I_t} a(i,j) - m \cdot k(j) \geq 0, \forall j \in J_t \tag{5.4}$$

$$M \cdot k(j) - \sum_{i \in I_t} a(i,j) \geq 0, \forall j \in J_t \tag{5.5}$$

The second set of constraints forces exactly one item to be allocated to each user:

$$\sum_{j \in J_t} a(i,j) = 1, \forall i \in I_t \tag{5.6}$$

5.3.2 Maximising user coordination

The last step of our framework aims to compute the remaining $H-1$ solutions (one has already been identified by $MILP^{first}$). We will do this by constructing and solving the MILP we will refer to as $MILP^{allOthers}$. $MILP^{allOthers}$ outputs all recommendation sets R_i^t , $\forall i \in I_t$, with the exception of the first element in each set which has been computed by $MILP^{first}$. It guarantees that each user is never recommended the same movie twice, and estimates the taxation for each user - movie pair outputted. The taxation is computed so as to guarantee that each user prefers their sponsored solution, under the assumption that the expectation over user's utility for each item is their true evaluation. The program outputs the minimum taxation that achieves this, as an infinite tax would still accomplish the same goal, but be equivalent to an allocation of the sponsored solution. A small error ε is allowed in this estimation. As previously stated, it is assumed that user's behave according to the constant response model, with a given probability for selecting their most preferred item α.

The input parameters to that model are m, M, and $u_i(x_j)$ $\forall i \in I_t$ and $j \in J_t$, as in the previous subsection, as well as:

- The sponsored allocation as outputted from $MILP^{first}$: $\mathbf{a}^*$; and
- The expected utility of each user for their sponsored solution x^*: $u_i(x_i^*)\ \forall i \in I_t$.

The variables for $MILP^{allOthers}$ are:

- An indicator for whether item i is allocated to user j through allocation h (out of $H-1$ allocations): $a(i,j,h) \in \{0,1\}$, $\forall i \in I_t$, $j \in J_t$ and $h \in \{1,...,H-1\}$;
- An artificial variable for whether an item j is being allocated through allocation h: $k(j,h) \in \{0,1\}$, $\forall j \in J_t$ and $h \in \{1,...,H-1\}$;
- The taxation imposed to the allocation of item j to user i in allocation h: $tax(i,j,h) \in \Re^+$, $\forall i \in I_t$, $j \in J_t$ and $h \in \{1,...,H-1\}$.

The objective function is the maximisation of the sum of Social Welfare across all allocations, minus the total imposed tax:

$$\max_{a(i,j,h),\, tax(i,j,h), i\in I_t,\ j\in J_t, h\in\{1,...,H-1\}} \sum_{i\in I_t}\sum_{j\in J_t}\sum_{h=1}^{H-1} u_i(x_j)\cdot a(i,j,h) - \sum_{i\in I_t, j\in J_t, h\in\{1,...,H-1\}} tax(i,j,h). \tag{5.7}$$

We require five additional sets of constraints, beyond the variable space definitions. The first one is a constraint on the minimum number of users that need to be allocated a movie for it to be allocated at all, along with the necessary constraints for setting $k(j)$ correctly:

$$\sum_{i\in I_t} a(i,j,h) - m\cdot k(j,h) \geq 0, \forall j \in J_t \text{ and } h \in \{1,...,H-1\} \tag{5.8}$$

$$M\cdot k(j,h) - \sum_{i\in I_t} a(i,j,h) \geq 0, \forall j \in J_t \text{ and } h \in \{1,...,H-1\} \tag{5.9}$$

The second set of constraints forces exactly one item to be allocated to each user for each allocation:

$$\sum_{j \in J_t} a(i,j,h) = 1, \forall i \in I_t \text{ and } h \in \{1,...,H-1\} \tag{5.10}$$

The third set of constraints forces all solutions for each user to be different across allocations:

$$\sum_{j \in J_t} |a(i,j,h) - a^*(i,j)| \geq 1, \forall i \in I_t \text{ and } h \in \{1,...,H-1\} \tag{5.11}$$

$$\sum_{j \in J_t} |a(i,j,h) - a(i,j,h')| \geq 1, \forall i \in I_t, \text{ and } h,h' \in \{1,...,H-1\}, with\, h \neq h'. \tag{5.12}$$

The fourth set of constraints forces taxes to take the minimum acceptable value:

$$\sum_{j \in J_t} \big(u_i(x_j) \cdot a(i,j,h) - tax(i,j,h)\big) \leq u_i(x_i^*) - \varepsilon, \forall i \in I_t, \text{ and } h \in \{1,...,H-1\} \tag{5.13}$$

Finally the fifth set of constraints guarantees that only allocation - item combinations that have been assigned to a user are taxed:

$$M \cdot x(i,j,h) - tax(i,j,h) \geq 0, \forall i \in I_t, j \in J_t \text{ and } h \in \{1,...,H-1\}. \tag{5.14}$$

Initial solution heuristic

Preliminary experiments indicated that $MILP^{allOthers}$ would occasionally (approximately 1/1000 experiments) fail to produce a solution within a reasonable time-frame (30 minutes). In order to avoid this during our experiments, as well as to allow for a solution to always be found, even when the optimisation procedure reaches a time-out, we developed a simple heuristic for feeding an initial viable solution to the $MILP^{allOthers}$ optimisation procedure:

For each allocation pick an item $j \in J_t$ and assign j to users until it has been assigned m times; Select a new item and repeat until all users have been allocated an item; Compute taxation according to Eq. 5.13, and 5.14.

Generalisation for multiple assignments of the same item

When using a coarse model of preferences, we can replace the item space I with the space of categories C, and postpone the selection of specific items to after the optimisation procedure. In fact, this would allow us to make no decision about which or how many items to consider during optimisation. Each category could be assigned any number of times to each user and within an allocation. Interestingly, this would also allow for other criteria to be used, besides user or system utility, when choosing representatives for each time a category was recommended.

Unfortunately, the sparsity of the data set used in our experiments does not allow for this, since not all users have rated an item from each category. In fact, the generalisation presented below would require a number of items in each category such that each recommended item can be drawn from the same class. This would not necessarily be a problem during real world deployment since we would not be constrained by a data set. Generally, if we have at least n items belonging to each class, then we can add a constraint on the maximum number of times each category can be allocated.

To achieve this generalisation in $MILP^{first}$ and $MILP^{allOthers}$, we need to treat items as not having a specific utility per user assigned, but rather add variables allowing the allocation of a category to each of these items. The objective function would then have to be adjusted to consider the utility per user of categories assigned to each item.

5.4 Experimental Evaluation

The aim of these experiments is not to justify the use of our optimisation procedure, a topic which was successfully handled in Andreadis *et al.* (2016), but rather to compare learning and optimisation performance between the use of *coarse preferences*, and *additive independence*. Three experiments are run, with the goal of providing evidence for the following statements (corresponding experiment in parentheses):

1. The proposed optimisation procedure does not scale in terms of the number of available items $|J_t|$, when users' preferences are modelled with additive independence (experiment 3);
2. The computational time for the optimisation procedure is less when coarse preferences are used, then when additive independence is used, given the same number of available items $|J_t|$ (experiment 1);
3. The computational time taken for the on-line update of users' preference models after their interaction with the recommended set is significantly less when coarse preferences are used, then when additive independence is used (experiments 1&2);
4. Using a coarse preferences model results in better system performance, in terms of Social Welfare, than using a model of additive independence (experiment 1); even if the latter procedure is given access to more available items (experiment 2).

Two complementary statements are true by definition but will be explained in this section in terms of how they appear in practise, namely:

- The proposed optimisation procedure scales in terms of the number of available items, $|J_t|$, when users' preferences are modelled with coarse preferences;

- Learning from user interaction with the recommended sets scales in terms of the length of vector representations of items $x_j \in \mathbf{X}$, when users' preferences are modelled with coarse preferences.

Since only the expectation over a user's evaluation of items is used in the optimisation procedure, the latter will scale in terms of the length of vector representations of items $x_j \in \mathbf{X}$, regardless of preference model.

5.4.1 General set-up

Each experiment instance will consist of a sequence of 20 sessions, in each of which, t, we will uniformly randomly select a set of users I_t and a set of items J_t, out of sets I and J, respectively. The latter have been extracted from the *MovieLens 20M Dataset* (Harper and Konstan, 2015), which contains a sparse matrix of user ratings of different movies. Each *set* of experiments will be composed of a number of experiment instances, and will be characterised by a specific *full* ground truth matrix, which was generated from the MovieLens dataset by the following procedure:

- Begin with an empty set of users *Users*, and the full sparse matrix of ratings $Matrix_0$;
- For $user = 1...50$
  - While no new user has been added to *Users*:
    1. Identify user with maximum number of rated items in I;
    2. Reject user with probability $p = 0.05$ else add this user to *Users* and remove him from I
  - Generate full ratings matrix $Matrix_{user}$ for all users in *Users*

- Output $Matrix_{50}$ and *Users*

It follows that each experiment set will have access to 50 users, prior to the random selection of users for each instance. Each *experiment* will in turn be composed by a given number of sets of experiments, each with a different ground truth matrix, for different users and movies.

Movies in the MovieLens 20M Dataset are rated on a discrete scale of 0.5 to 5 stars, with a step of 0.5 stars. Each movie has a vector description of length 1128, with each parameter taking continuous values in $[0, 1]$. These values have been learnt as part of the research described in Harper and Konstan (2015), by considering how users *tagged* movies, and is an attempt to summarise those tags in a more comprehensive form. In order for the benchmark to function over this dataset, we have discretised the assignments to each of these parameters by splitting the space $[0, 1]$ into 4 intervals of length 0.25.

All ground truth matrices were based on the MovieLens 20M Dataset, which has 138493 users each rating some of 27278 movies, for a total of 20000263 ratings, with 465564 tag applications. For each experiment, any data points that did not end up in the ground truth, were used as the training data. Each user has rated at least 20 items.

We learn our prior for both models over the training data set. For the benchmark, this meant running *Linear Regression* Murphy (2012) separately over each user's data points, and redistributing the bias, 1, into each parameter by adding a constant $1/1128$ into each partial utility (each assignment's evaluation for each parameter). For the coarse preferences case, we executed the procedure outlined in Section 4.4.2, in order to generate the space of categories, and then used the Regression Tree prediction, and the variance of data points belonging to each node, as the mean and variance for the respective coarse prior. We did not test for using the Tree to partition every user's

individual data and then get the expectation over that user's points; mostly because we did not expect all user - class pairs to be populated in the data set.

At the beginning of each session t, the optimisation procedure receives as input a set of available movies, which may be any subset of J_t, and the expectation over each user's $i \in I_t$ utility evaluation of each of those movies, as computed from the current belief over each user's utility function $b_{t-1}(u_i)$. The optimisation procedure is executed, and each user i is recommended a set of movies R_i^t of size H, from which they select a movie r_i^t according to their utility function u_i and their a constant user response model with probability α. Since we have no knowledge of users' true utility functions, we simulate this by assuming that their ratings as given in the ground truth matrix are accurate representations of their preferences.

After each user's selection, we sequentially execute one update procedure for each implied pairwise comparison (see Section 5.2.2), by passing the current belief as a prior to a *TrueSkill* Herbrich *et al.* (2007) optimisation procedure along with the vector descriptions of the compared movies, and an indication of which movie was selected. The resulting potentials are then used in updating our belief to b_i^t, by replacing them in the corresponding parameter-assignment pairs. In the coarse preferences case, the vector description of a movie consists solely of the index of the category to which it maps.

In order to accommodate for the taxation of solutions, we extended the model in Guo and Sanner (2010) by adding a taxation factor for each taxed item. Consistent with the original model, taxation factors are chosen as Gaussians with mean equal to the taxation and a 'small' standard deviation of 0.1, since we cannot use a variance of 0. In real-world applications we will also have to learn the utility associated by each user to the particular incentive used for taxing, so this variance will not have to be arbitrarily set.

All 3 experiments described below follow this motif; the differences lying in the size of the recommended sets H and the sets of users I_t and movies J_t, the number of experiment sets and instances, which algorithms we run, whether we use the initialisation heuristic in Section 5.3.2, and which movies are forwarded to the optimisation procedure from the available set J_t. We assume a constant user response model with probability $\alpha = 0.7$, and set the utility error to $\varepsilon = 0.05$.

We compare the two procedures based on:

- The *normalised loss of Social Welfare*, computed as the percentage of loss of Social Welfare from the maximum Social Welfare possible;

- The computational time taken to run the optimisation procedure, including the time to set it up (in msecs); and

- The computational time taken to update each respective preference model after user selection (in msecs).

When evaluating the Social Welfare at the end of each session, we subtract from every user's utility any respective taxation applied to their selected movie. We assume that the maximum social welfare achievable is $5 \times |I_t|$.

All mixed integer linear programs were solved using the CPLEX (IBM, 2017) optimiser. Experiments were run in Java, on a laptop PC with an Intel Core i5-6300U CPU 2.40GHz, having 2 cores.

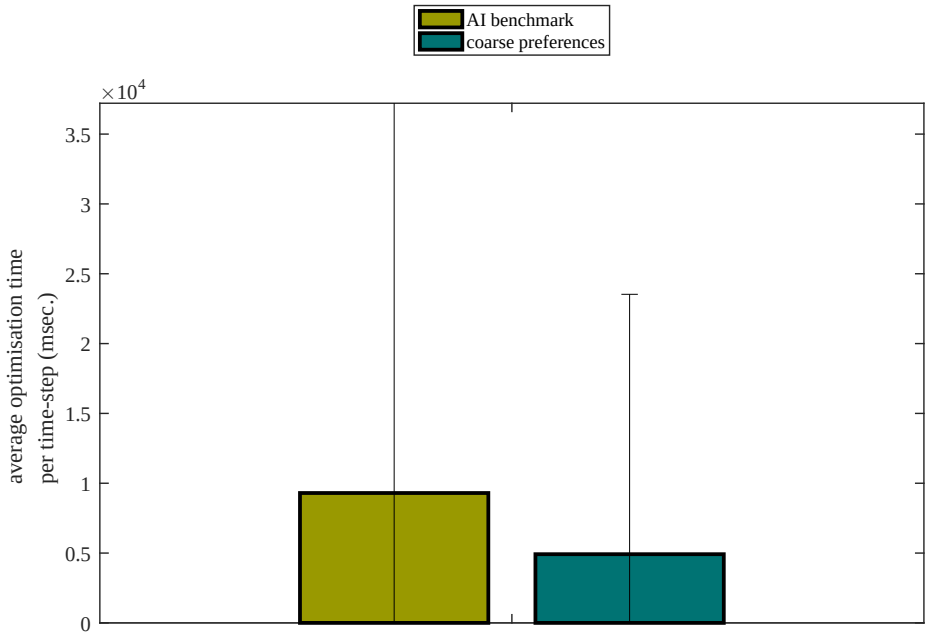


Figure 5.1: Experiment 1: Average computational time (in *msecs*) for setting up and executing the proposed optimisation procedure using additive independence and coarse preferences user models.

5.4.2 Experiment 1 - basic performance

The first experiment sets an even stage for the two learning procedures. We compute set recommendations with a size $H = 3$, for sets of $|I_t| = 20$ users and providing the optimisation procedure with sets of $|J_t| = 10$ items.

We run 10 experiment sets of 100 instances each. These experiments did not make use of the initialisation heuristic from Section 5.3.2.

Figures 5.3, 5.1, and 5.2 show the results of the optimisation in terms of the average normalised loss of Social Welfare, the average computational time taken to set up band perform the optimisation, and the computational time taken to perform the updates

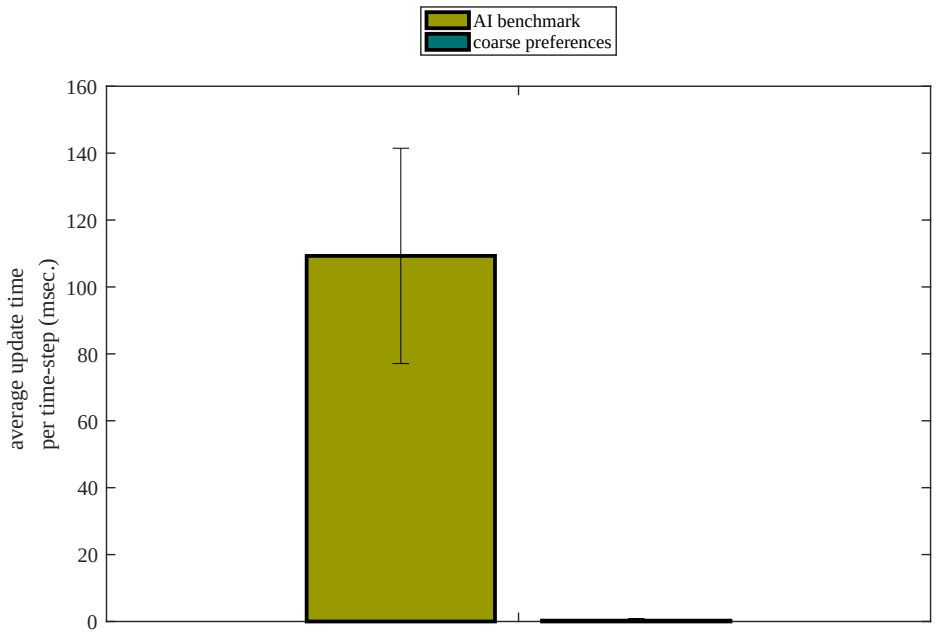


Figure 5.2: Experiment 1: Average computational time (in *msecs*) for updating all users' preference models using additive independence and coarse preferences user models.

of users' utility functions after their selections, respectively. All error bars are of ± 2 standard deviations around the mean, indicating that the value is within that interval with 95.45% confidence.

We notice that the procedure utilising the coarse preferences model converges rapidly to a significantly better solution than the model utilising the additive independence model. Our model initially achieves an improvement in the allocated item to each user of an expected Âij of a star rating. This is computed after items that don't meet the criterion of minimum number of allocated users have been filtered out (users that are allocated to such items are treated as receiving 0 utility). This is a noticeable improvement in recommendation, especially since queries are not selected to increase informativeness.

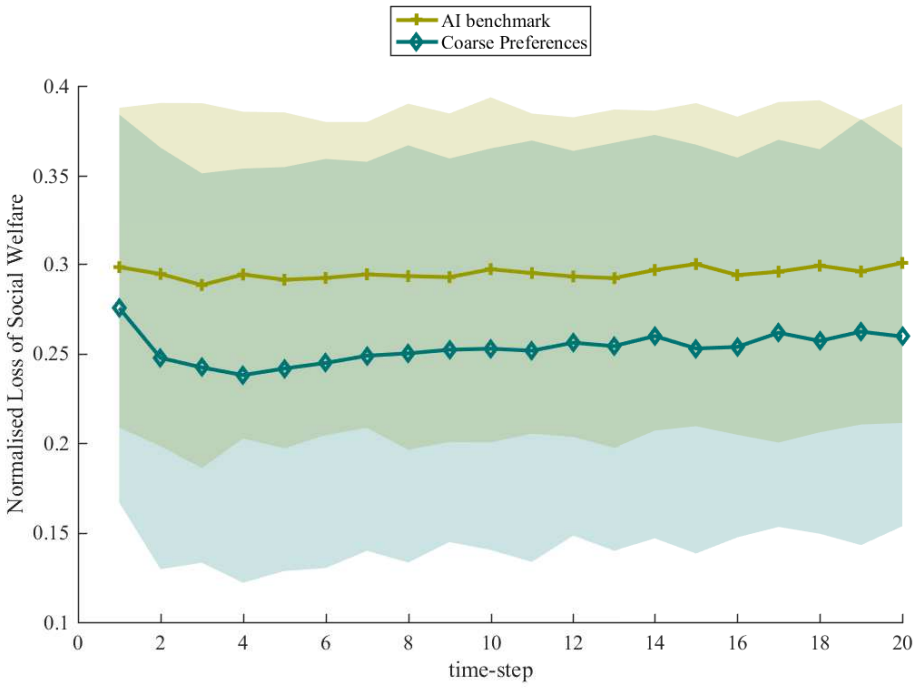


Figure 5.3: Experiment 1: Average Normalised Loss of Social Welfare using additive independence and coarse preferences user models.

However, our model's performance stops improving starting from the 5th time-step, and in fact slightly decreases. It is not immediately clear why this would occur. Provided that items are uniformly randomly present in queries, the model's accuracy should improve until convergence. Either the performance of our optimisation procedure does not monotonically increase with model accuracy, or there is bias in favour of the representation of certain items in our queries. We investigate this further with the second set of experiments.

The benchmark's performance is similar to our results in the previous chapter and are indicative of how user behaviour is not well represented by a linear function. However, there is some improvement, and given the small available set of items that constraints our experiments, the linear model could eventually learn the identity function; provided that each item in the ground truth has at least one assignment to a parameter that is

different from every other item's respective assignment. This is of course not possible for real world deployment.

We ran an unpaired two-sample t-test (Dell *et al.*, 2002) for all data points' normalised losses at the 4th time-step. Specifically, a left-tailed test for the benchmark having a larger normalised loss than our model. The null hypothesis of this not being the case was rejected at 5% significance. This is evidence that even though there is a large variance on the performance, this is more indicative of differences across experiments, with both models being affected. We can be confident that our model is outperforming the benchmark at the 4th time-step.

The computational time taken to update the coarse model is significantly less than that of the benchmark, as expected. however, most of the computational time is devoted to setting up and running the optimisation procedure. Though the difference is much less pronounced, coarse preferences still achieves better results on this metric as well.

Even though the optimisation time is significantly larger than the update time, there are case where gains in the latter can be of importance:

- Expanding the procedure in this chapter with some form of active learning would mean that the total gains in update time (as occurs during query evaluation) would increase linearly with the number of query - response pairs examined.
- Certain applications might decentralise user model updates, only communicating the expectation over each item or category to the system. The impact of reducing computations could be of greater importance when performed on a mobile platform.

The results from this experiment provide some evidence for statements 1 and 2 above, indicating better performance with significant gains in computational time. It is

worth noting that the computational time for learning in the coarse preferences case is representative of all scenarios utilising these procedures, since movie viewings are always represented by a single parameter: the coarse class.

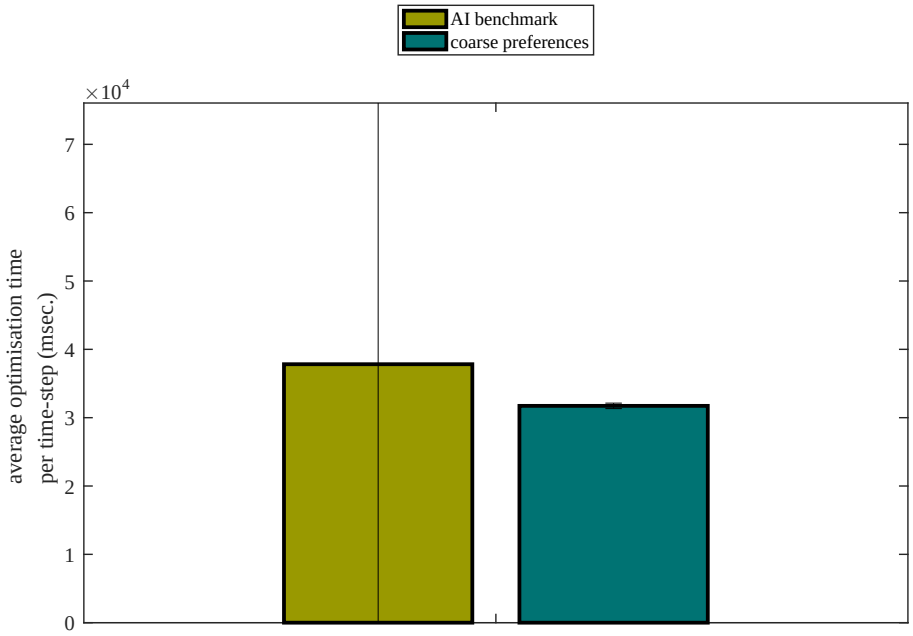


Figure 5.4: Experiment 2: Average computational time (in *msecs*) for setting up and executing the proposed optimisation procedure using additive independence and coarse preferences user models.

5.4.3 Experiment 2 - increased benchmark item set

In the second set of experiments, we increase the difficulty of the optimisation problem by requiring recommendation sets of $H = 5$ movies for each of $|I_t| = 21$ users. The optimisation procedure is given access to a set of movies J_t which varies for each experiment instance, in the additive independence case, or for each session, for the coarse preferences case. Specifically, in the case of additive independence we always feed all of the items in the ground truth to the optimisation procedure. In the coarse

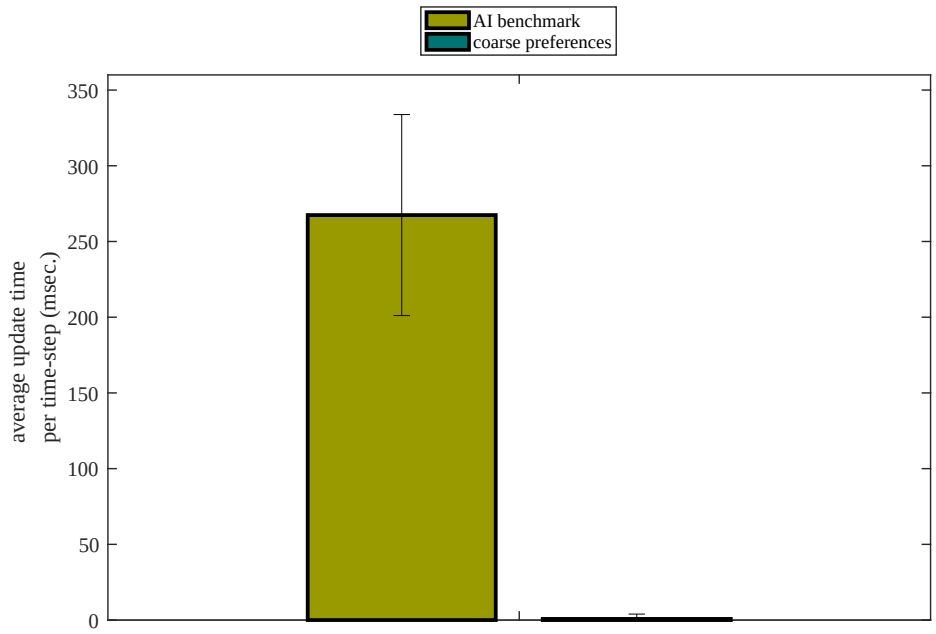


Figure 5.5: Experiment 2: Average computational time (in *msecs*) for updating all users' preference models using additive independence and coarse preferences user models.

preferences case, however, we provide the procedure with one representative from each coarse class that is present in the ground truth for this instance. If these are not enough to guarantee that a solution will be found, we provide an additional representative from the items that have still to be added, and so on until enough items are forwarded to the optimiser. The number of movies that are required in order to guarantee a solution are defined as the ones necessary for the initial solution heuristic presented in Section 5.3.2 (in our case 15). Lastly, the second stage of the optimisation procedure (see Section 5.3.2) has been given an upper time limit of 30 seconds. Though this was done to speed up the experiments, it is well justified by the assumed application.

Figures 5.6, 5.4, and 5.5 show the results of the optimisation in terms of the average normalised loss of Social Welfare, the average computational time taken to set up band

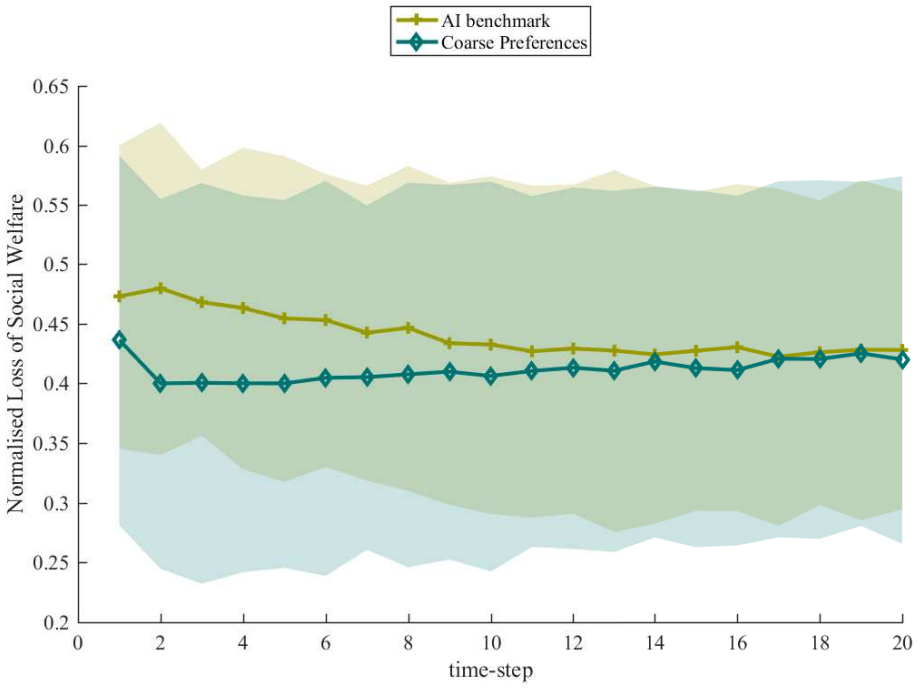


Figure 5.6: Experiment 2: Average Normalised Loss of Social Welfare using additive independence and coarse preferences user models.

perform the optimisation, and the computational time taken to perform the updates of users' utility functions after their selections, respectively. All error bars are of ± 2 standard deviations. We expect that the increased number of pairwise comparison query - response pairs per time-step will speed up convergence for both models.

Validating our expectations stated at the end of the previous section, the learning time of coarse preferences remains very small. However, the gap in optimisation time has decreased, meaning that both models attempt to make full use of the 30 *sec* time limit. The remaining difference is then best attributed to the set up of the optimisation procedure, which we expect scales up linearly for the benchmark in terms of the number of items considered. This latter point will be tested for in the third set of experiments.

We note here, that for the coarse case, the optimisation time taken is invariant to the size of the available set of items, since we can represent the entire data set by selecting one representative from each class. The system designers can still elect to add multiple representatives from each class however, especially if they are considering some additional criterion or constraint, such as recommending each movie a set number of times. Use of the modification proposed in Section 5.3.2 would make even this consideration redundant, since the number of items from each class would be selected by the optimisation procedure.

Figure 5.6 illustrates the problem we first identified in Section 5.4.2, though now at a significant detriment to performance. The coarse preferences model quickly converges to a better result than the AI benchmark, but subsequently de-converges until its performance matches that of the benchmark. On the other hand, the benchmark matches our expectations from the previous section, since the increased number of pairwise comparisons per time-step increases its convergence rate. As in experiment 1, we ran an unpaired two-sample t-test (Dell *et al.*, 2002) for all data points' normalised losses at the 4th time-step. Specifically, a left-tailed test for the benchmark having a larger normalised loss than our model. The null hypothesis of this not being the case was rejected at 5% significance.

Considering the fundamental differences between the first and second experiments, what has changed is the number of updates per time-step, the optimisation problem's complexity, and the pre-optimisation selection procedure for the coarse model. The latter is not a part of the model but, rather, a concession to our small ground truth matrix. When analysing the results from the first set of experiments we hypothesised that one of the following must be the case: *Either the performance of our optimisation procedure does not monotonically increase with model accuracy, or there is bias in favour of the representation of certain items in our queries*.

After examining our implementation of the item pre-selection procedure, we discovered that the representatives of each category were not uniformly randomly selected from the set of items mapping to the category, but was favouring earlier item indices. We elected to present these experimental results despite that, since they provide some insight into why the de-convergence occurred in the earlier experiments as well. Provided that correcting the item pre-selection procedure will improve our results, so that they more closely resemble those of Section 5.4.2, we can with some confidence attribute this phenomenon to *bias in favour of the representation of certain items in our queries*. Such bias will easily appear in our experiments, since the small number of representative items for each category means that there is high likelihood that a number of categories will be disproportionately represented in the recommendation sets by a subset of these items. This would be a positive result, since such a phenomenon would be improbable in a setting with a dynamic set of items, and practically impossible with a compositional item space.

At this point, it is important to clarify *how* this phenomenon of imbalance in representation between the items mapped to a category can adversely affect system performance, in terms of average normalised loss of social welfare. Though the items that are over-represented will have their ground truth more accurately predicted, the loss of accuracy on the other items in the category can negatively impact taxation by:

- overtaxing solutions that compete with a sponsored solution that is not a priori preferable by some user;

- by not taxing non-sponsored solutions whose utility has been underestimated.

5.4.4 Experiment 2b - Adjusting query randomisation

The experiments in the previous section revealed a concern relating to the drop in performance for the coarse preferences model due to the non-uniform sampling of items by the recommendation procedure. We will investigate this concern by correcting a flaw in the item pre-selection procedure, and observing whether this brings the performance of our model given the setup in the second set of experiments back to that of the first set of experiments.

Figure 5.7 presents the performance of our model with corrected pre-selection in comparison to the results of the AI benchmark from Section 5.4.3. Besides this correction, no other change was applied, though constraints in available experiment time have restricted the number of experiment instances per experiment to 10, which accounts for the noisier curve.

We observe, with some added uncertainty, that the performance of our model has increased significantly, approaching the results from Section 5.4.2. From this we infer that our model is sensitive to bad sampling from item spaces with small cardinality, which cannot be entirely corrected for without balancing model exploitation with some form of active learning. Fortunately, the problems we are concerned with in this thesis do not have the characteristic of discrete, small, constant item spaces, and this is not a behaviour we expect to see during real world deployment. Again, we ran an unpaired two-sample t-test (Dell *et al.*, 2002) for all data points' normalised losses at the 4th time-step. Specifically, a left-tailed test for the benchmark having a larger normalised loss than our model. The null hypothesis of this not being the case was rejected at 5% significance.

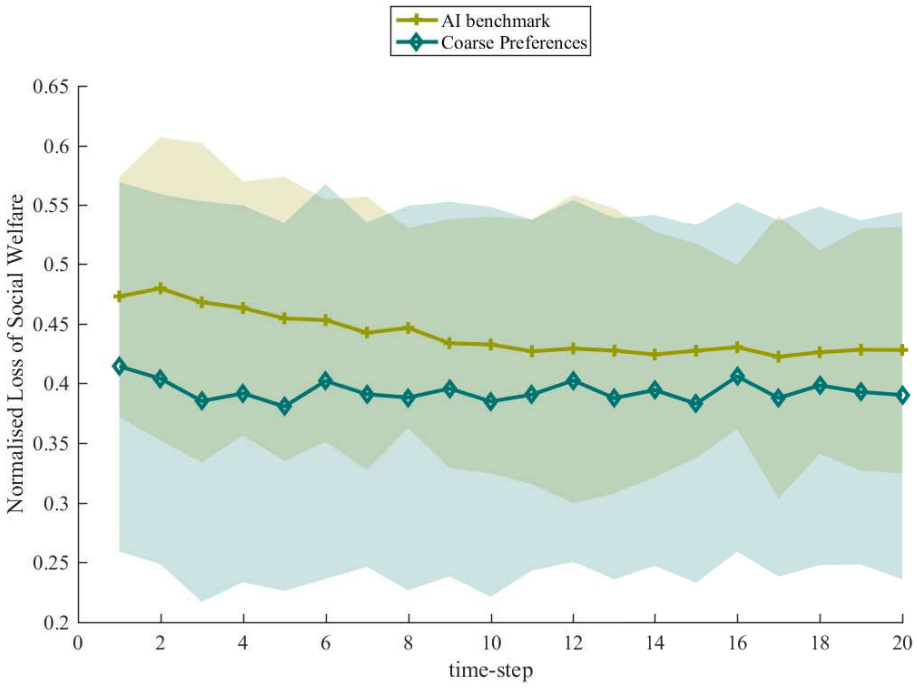


Figure 5.7: Experiment 2b: Average Normalised Loss of Social Welfare using additive independence and coarse preferences user models, after correcting the pre-selection procedure.

5.4.5 Experiment 3 - benchmark optimisation scaling

The purpose of this experiment is to demonstrate that the proposed optimisation procedure does not scale in terms of number of potential items for recommendation, when the additive independence model of preferences is used. For that reason, we only run this set of experiments with the benchmark learning procedure. We run 5 sets of 15 experiments, with each experiment giving the optimisation procedure access to all of J_t, whose size is incremented by 1 after each experiment, in the range $\{15, ..., 29\}$. Each session involves 21 users who are each recommended a set of 5 movies.

We plot the time it takes the system to perform an optimisation procedure in Figure 5.8, along with a linear fit. The latter has a gradient of $0.241 \cdot 10^3 msecs$ and an intercept of

$27.623 \cdot 10^3 msecs$. The linear increase in expected time, coupled with a relatively small 2x standard deviations area, indicates that running this optimisation procedure over a large set of available options, such as the MovieLens dataset, would be practically infeasible.

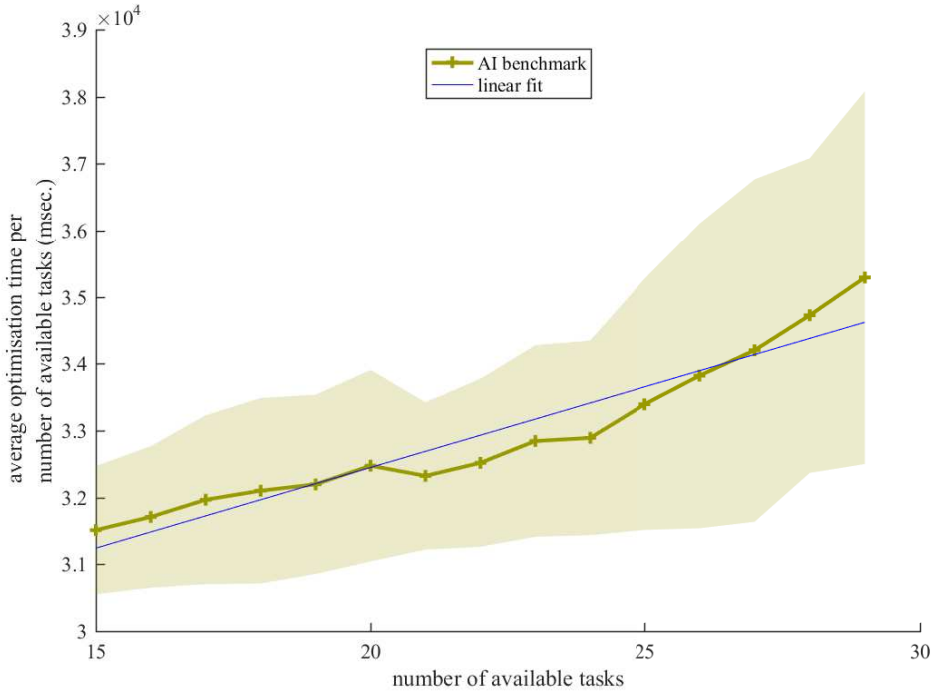


Figure 5.8: Experiment 3: Average computational time (in *msecs*) for setting up and executing the proposed optimisation procedure using an additive independence model of user preferences, for different sizes of the set of available movies. The blue line is the linear fit to the data.

5.5 Conclusion

This chapter concludes this thesis' contributions to the areas of decision theory and preference elicitation. Having presented the use of coarse preferences in inference (Chapter 3) and learning (Chapter 4), this chapter brought those results together to

demonstrate the potential of the approach in complex, time-critical, on-line optimisation problems. In order to do so, we introduced the problem of *coordination through set recommendation* and an approximate solution procedure. We showed that representing items with their coarse classes enables significant computational and performance gains, when compared to an additive representation. Furthermore, updating coarse user preference models was significantly faster than updating a model using additive independence.

The results in this chapter indicate how coarse preferences can be used for enabling otherwise computationally prohibitive operations involving users preferences; specifically as relates to the learning of, and the optimisation with, user preference models.

Chapter 6

Conclusions, Discussion, and Future Work

The advent of on-line services provides ample opportunity for the learning of users' preferences, as they interact with the system at hand. Though typical recommender systems applications would make use of such interaction data offline, in order to better learn how to accommodate for these users in the future, there have been approaches developed for updating user models as they interact, online. However, there is little work in making such procedures manageable in time-critical, computationally complex problems, such as those that emerge from attempting to coordinate the decisions of multiple users logged into a service.

As demonstrated in our case study of *coordination through set recommendation* in Chapter 5, optimising for the recommendations to be made to a set of users, such that meaningful choice is permitted, is computationally costly, and does not scale with the number of considered options. Moreover, the time to update our belief of a user's preferences can be significantly reduced, if the proper model is used.

In this thesis, we proposed such a model, titled 'Coarse Preferences', which represents users as distinguishing between options in terms of the category in which they fall. Having proven compatibility with the von Neumann and Morgenstern (1953) Expected Utility Theorem, we proceeded to show how to make decisions over sets of users characterised by such models; something necessitated by how each user's preferences are defined over a different latent space. Given this, we developed a procedure for learning such models, with emphasis on how to learn a latent space of categories that is viable for all the user population. Armed with this, we demonstrated that our model performs significantly faster than the state of the art model based on additive independence, and that it can in fact be a better match for users' behaviour, in domains such as the MovieLens 20M Dataset (Chapter 5), and the Mallzee dataset (Chapter 4), as provided for this study.

For the cost of some offline computation at setup, our approach allows for significant speedups of online learning and inference. Conclusively, it is our expectation that our methods will enable otherwise computationally prohibitive recommender systems to be designed. The increased outputs demonstrated in terms of user utility and system revenue, indicates that our methods can also improve system performance, even in the cases where computational time is of little concern.

Still, since our offline learning procedure is based on regression tree learning, our model can be said to be discriminatory instead of generative; at least as concerns the latent space of categories. We provided some versions of a generative model of coarse preferences in Chapter 3, and will consider extending that work in the near future. Even so, we are interested in considering other approaches towards generating a discriminative model of the coarse latent space, including the invention, or validation of, a regression tree metric that would guarantee discovering the coarse choices, in the limit of samples used. This metric would likely be the sum of sample variances across nodes, conditioned on the user or user type, but more work needs to done

to produce efficient algorithms that would work with this metric. Beyond that, we have already begun experiments using Deep Learning (Goodfellow *et al.*, 2016) for producing the coarse partitioning offline. Producing a partitioning for all users even as the target utility variable is conditioned on specific users, has created interesting problems; particularly when considering error representation and backpropagation.

Regardless of such potential advancements however, our prime interest rests in discovering what other computationally complex recommendations or other procedures can be enabled by use of the technology presented in this work. In terms of applications, we expect that coordination through set recommendation can be put to use in the on-line service or other industries.

6.1 The Effect of Coarseness on the Robustness of Solutions

Chapters 4 and 5 demonstrated how the coarse preferences model allows for factorised decision making. In Chapter 4 we were able to identify a user's most preferred category and then select the most expensive item in that category for recommendation. In Chapter 5 we were able to reduce the effective cardinality of the solution space, and therefore reduce computational time, by only considering a limited number of items from each category.

Another significant benefit of this factorised representation was not addressed in the experiments: the interchangeability of solutions adds robustness to systems that cannot afford to recompute a solution when items allocated in that solution become unavailable. With the coarse preferences model, we can easily replace an item made unavailable during execution with some other of the same category and expect no

change in system performance. This increased robustness in terms of allocations is worth considering for future study.

6.2 Expanding on the Decision-Theoretic Aspect of the Study

As coarse preferences significantly reduces the dimensionality of the item space, to 1, and can significantly reduce its cardinality to a small set of categories, it would be worth investigating methods to take advantage of that in computationally complex preference elicitation tasks. For example, if we were to extend the work of Chapter 5 to consider the expected value of information from recommendations, we could develop efficient procedures for preference elicitation in multi-user set recommendation, similarly to what was done for set recommendation in Viappiani and Boutilier (2010). If the problem was simplified to ignore probabilistic aspects, then theoretical guarantees could be given on the number of queries needed to elicit the true utility function for all users.

From the point-of-view of multi-agent resource allocation research (Chevaleyre *et al.*, 2006), coarse preferences can be a good model for time-critical resource allocation problems. The problem specification in Chapter 5 was such that the issues of envy-freeness and incentive-compatibility (truthfulness) were not relevant. However, the same chapter also discusses how allowing users access to, or assuming they have access to, more information about the preferences of other users would change that. It could be worth investigating whether coarse preferences can be effectively used in producing recommendation procedures that would guarantee some properties typicall addressed in the Mechanism Design literature (Nisan and Ronen, 2001). Are there benefits in using coarse preferences when considering how to design the recommendation

procedure such that, for example, users are *truthful* in their selections, i.e. they actually select, or tend to select, their most preferred item? Another interesting extension would be to produce algorithms that explicitly handle further recommendations to unallocated users, considering, for example, what happens to users that were not allocated after the first round of item selections.

Bibliography

Abbas, A. (2004). Entropy methods for adaptive utility elicitation. **34**.

Abbasnejad, M.E., Bonilla, E.V. and Sanner, S. (2013). Decision-theoretic sparsification for gaussian process preference learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 515–530, Springer, Prague, Czech Republic.

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. **17**.

Agatz, N., Erera, A., Savelsbergh, M. and Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, **45**, 1450–1464.

Agatz, N., Erera, A., Savelsbergh, M. and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, **223**, 295–303.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann.

Allan, J., Carbonell, J., Doddington, G., Yamron, J. and Yang, Y. (1998). Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*.

Anand, S.S. and Mobasher, B. (2005). Intelligent techniques for web personalization. In *Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization*, Springer.

Andreadis, P. (2016). Mallzee Dataset. `https://datahub.ckan.io/dataset/mallzee-dataset`.

Andreadis, P., Ceppi, S., Rovatsos, M. and Ramamoorthy, S. (2016). Diversity-aware recommendation for human collectives. In *European Conference on Artificial Intelligence Workshop on Diversity-aware Artificial Intelligence (DIVERSITY @ ECAI 2016)*, 23–32, the Hague, Netherlands.

Angluin, A. (1988). Queries and concept learning. *Machine Learning*, **2**, 319–342.

Aziz, H., Brill, M., Fischer, F.A., Harrenstein, P., Lang, J. and Seedig, H.G. (2015). Possible and necessary winners of partial tournaments. *J. Artif. Intell. Res. (JAIR)*, **54**, 493–534.

Bacchus, F. and Grove, A. (1995). Graphical models for preference and utility. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 3–10, Morgan Kaufmann Publishers Inc.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley.

Bell, R., Bennett, J., Koren, Y. and Volinsky, C. (2009). The million dollar programming prize. *Spectrum, IEEE*, **46**, 28–33.

Bell, R.M., Koren, Y. and Volinsky, C. (2007). The bellkor solution to the netflix prize.

Billingsley, P. (2008). *Probability and measure*. John Wiley & Sons.

Billsus, D., Pazzani, M.J. and Chen, J. (2000). A learning agent for wireless news access. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, ACM.

Bjorndahl, A., Halpern, J.Y. and Pass, R. (2013). Language-based games. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2967–2971, AAAI Press.

Bohnert, F., Schmidt, D.F. and Zukerman, I. (2009). Spatial processes for recommender systems. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*.

Boutilier, C. (2002). A POMDP formulation of preference elicitation problems. In *Proceedings of the 18th National Conference on Artificial Intelligence*, 239–246, AAAI.

Boutilier, C. (2003). On the foundations of *expected* expected utility. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 285–290, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Boutilier, C., Brafman, R.I., Geib, C. and Poole, D. (1997). A constraint-based approach to preference elicitation and decision making.

Boutilier, C., Bacchus, F. and Brafman, R.I. (2001). UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, UAI'01, 55–64, Seattle, WA.

Boutilier, C., Das, R., Kephart, J.O., Tesauro, G. and Walsh, W.E. (2003a). Cooperative negotiation in autonomic systems using incremental utility elicitation. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 89–97, Morgan Kaufmann Publishers Inc.

Boutilier, C., Patrascu, R., Poupart, P. and Schuurmans, D. (2003b). Constraint-based optimization with the minimax decision criterion. In *Principles and Practice of Constraint Programming*, 168–182, Springer.

Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H. and Poole, D. (2004a). CP-nets: A tool for representing and reasoning with conditional *Ceteris Paribus* preference statements. **21**.

Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H. and Poole, D. (2004b). Preference-based constrained optimization with CP-nets. **20**.

Boutilier, C., Sandholm, T. and Shields, R. (2004c). Eliciting bid taker non-price preferences in (combinatorial) auctions. In *Proceedings of the 19th National Conference on Artificial Intelligence*.

Boutilier, C., Patrascu, R., Poupart, P. and Schuurmans, D. (2005). Regret-based utility elicitation in constraint-based decision problems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*.

Boutilier, C., Patrascu, R., Poupart, P. and Schuurmans, D. (2006). Constrained-based optimization and utility elicitation using the minimax decision criterion. **170**.

Boutilier, C., Regan, K. and Viappiani, P. (2009a). Online feature elicitation in interactive optimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML-09, 73–80, Montreal, Quebec, Canada.

Boutilier, C., Regan, K. and Viappiani, P. (2009b). Preference elicitation with subjective features. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, 23–25, New York, New York, USA.

Boutilier, C., Regan, K. and Viappiani, P. (2010). Simultaneous elicitation of preference features and utility. In *Proceedings of the 24th National Conference on Artificial Intelligence*, AAAI-10, 1160–1197.

Boutilier, C., Zemel, R.S. and Marlin, B.M. (2012). Active collaborative filtering. *CoRR*, **abs/1212.2442**.

Bouveret, S., Chevaleyre, Y. and Maudet, N. (2016). Fair allocation of indivisible goods.

Bradley, R.A. and Terry, M.E. (1952). Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, **39**, 324–345.

Brafman, R. and Domshlak, C. (2002). Introducing variable importance tradeoffs into CP-nets. In *Proceedings of the 18th National Conference on Artificial Intelligence*.

Braziunas, D. (2006). Computational approaches to preference elicitation. Tech. rep., University of Toronto.

Braziunas, D. and Boutilier, C. (2005). Local utility elicitation in gai models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

Braziunas, D. and Boutilier, C. (2006). Preference elicitation and generalized additive utility. In *Proceedings of the National Conference on Artificial Intelligence*, vol. 21.

Braziunas, D. and Boutilier, C. (2007). Minimax regret based elicitation of generalized additive utilities.

Braziunas, D. and Boutilier, C. (2009). Elicitation of factored utilities. *AI Magazine*, **29**, 79.

Braziunas, D. and Boutilier, C. (2010). Assessing regret-based preference elicitation with the utpref recommendation system. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, ACM.

Breese, J.S., Heckerman, D. and Kadie, C.M. (1998). An empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann.

Breiman, L., Friedman, J., Stone, C.J. and Olshen, R.A. (1984). *Classification and regression trees*. CRC press.

Buckley, C., Salton, G. and Allan, J. (1994). The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Springer.

Burke, R. (2000). Knowledge-based recommender systems. **69**.

Burke, R. (2002). Interactive critiquing for catalog navigation in e-commerce. **18**.

Burke, R. (2007). Hybrid web recommender systems. In *The Adaptive Web*, 377–408, Springer.

Burke, R., Hammond, K. and Young, B. (1997). The findme approach to assisted browsing. **4**.

Busa-Fekete, R. and Hüllermeier, E. (2014). A survey of preference-based online learning with bandit algorithms. In *International Conference on Algorithmic Learning Theory*, 18–39, Springer.

Cafaro, M., Mirto, M. and Aloisio, G. (2013). Preference–based matchmaking of grid resources with cp–nets. *Journal of grid computing*, **11**, 211–237.

Canny, J. (2002). Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM.

Chajewska, U. and Koller, D. (2000). Utilities as random variables: Density estimation and structure discovery. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc.

Chajewska, U., Getoor, L., Norman, J. and Shahar, Y. (1998). Utility elicitation as a classification problem. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 79–88, Morgan Kaufmann Publishers Inc.

Chajewska, U., Koller, D. and Parr, R. (2000). Making rational decisions using adaptive utility elicitation. In *Proceedings of the 17th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 363–369, AAAI Press.

Chajewska, U., Koller, D. and Ormoneit, D. (2001). Learning an agent's utility function by observing behaviour. In *Proceedings of the 18th International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc.

Chakrabarti, S. (2002). *Discovering Knowledge from Hypertext Data*. Science and Technology Books.

Chee, S.H.S., Han, J. and Wang, K. (2001). Rectree; an efficient collaborative filtering method. In *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*.

Chen, L. and Pu, P. (2004). Survey of preference elicitation methods. Tech. rep.

Chevaleyre, Y., Endriss, U., Estivie, S., Maudet, N. *et al.* (2004). Multiagent resource allocation with k-additive utility functions. In *Proceedings of the DIMACS-LAMSADE workshop on computer science and decision theory*, vol. 3, 83–100, ILLC.

Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J.A. and Sousa, P. (2006). Issues in multiagent resource allocation.

Chevaleyre, Y., Endriss, U., Lang, J. and Maudet, N. (2008). Preference handling in combinatorial domains: From ai to social choice. *AI magazine*, **29**, 37.

Chevaleyre, Y., Koriche, F., Lang, J., Mengin, J. and Zanuttini, B. (2011). Learning ordinal preferences on multiattribute domains: The case of CP-nets.

Chung, K.S. (2000). On the existence of stable roommate matchings. *Games and Economic Behavior*, **33**, 206 – 230.

Cohen, W.W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, Morgan Kaufmann.

Conitzer, V. (2007). Eliciting single-peaked preferences using comparison queries. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 65, ACM.

Crès, H. (2001). Aggregation of coarse preferences. *Social Choice and Welfare*, **18**, 507–525.

Darmann, A. and Schauer, J. (2015). Maximizing nash product social welfare in allocating indivisible goods. *European Journal of Operational Research*, **247**, 548–559.

Decock, L. and Douven, I. (2011). Similarity after Goodman. *Review of Philosophy and Psychology*, **2**, 61–75.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. (1990). Indexing by latent semantic analysis. **41**.

Dell, R.B., Holleran, S. and Ramakrishnan, R. (2002). Sample size determination. *ILAR journal*, **43**, 207–213.

Dickerson, J.P., Procaccia, A.D. and Sandholm, T. (2012). Optimizing kidney exchange with transplant chains: theory and reality. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 711–718.

Doshi, F. and Roy, N. (2008). The permutable POMDP: Fast solutions to POMDPs for preference elicitation. In *Proceedings of the 7th International Joint Conference Autonomous Agents and Multiagent Systems*, vol. 1, 493–500, International Foundation for Autonomous Agents and Multiagent Systems.

Doyle, J., Shoham, Y. and Wellman, M. (1991). A logic of relative desire (preliminary report). In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems, Lecture Notes in Computer Science*, Springer-Verlag.

Duan, L., Street, W.N. and Xu, E. (2011). Healthcare information systems: Data mining methods in the creation of a clinical recommender system. *Entrprise Information Systems*, **5**, 169–181.

Elahi, M., Ricci, F. and Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, **20**, 29 – 50.

Farfel, J. and Conitzer, V. (2011). Aggregating value ranges: Preference elicitation and truthfulness. *Autonomous Agents and Multi-Agent Systems*, **22**, 127–150.

Farquhar, P.H. (1984). State of the art - utility assesment methods. *Management Science*, **30**, 1283–1300.

Felfernig, A. and Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. In *Proceedings of the 10th International Conference on Electronic Commerce*, ACM.

Felfernig, A., Friedrich, G., Jannach, D. and Zanker, M. (2006). An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, **11**, 11–34.

Fishburn, P.C. (1968). Utility theory. **14**.

Fishburn, P.C. (1970). *Utility theory for Decision Making*. Wiley, New York.

Fishburn, P.C. (1999). Preference structures and their numerical representations. *Theoretical Computer Science*, **217**, 359–383.

French, S. (1986). *Decision Theory*. Halsted Press, New York.

Gale, D. and Shapley, L.S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, **69**, 9–15.

Geanakoplos, J., Pearce, D. and Stacchetti, E. (1989). Psychological games and sequential rationality. *Games and Economic Behavior*, **1**, 60–79.

Ghosh, C. and Kalagnanam, J. (2003). Polyhedral sampling for multiattribute preference elicitation. In *Proceedings of 5th ACM Conference on Electronic Commerce*.

Gigerenzer, G. and Selten, R. (2003). *Bounded Rationality: The Adaptive Toolbox*. The MIT Press.

Goldberg, K., Nichols, D., Oki, B.M. and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. **35**.

Goldberg, K., Roeder, T., Gupta, D. and Perkins, C. (2004). Eigentaste: A constant time collaborative filtering algorithm. **4**.

Golub, G. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. **2**.

Gonzales, C. and Perny, P. (2004). GAI networks for utility elicitation. 224–234.

Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2016). *Deep learning*, vol. 1. MIT press Cambridge.

Goodman, N. (1972). Seven strictures on similarity. *Goodman, N (Ed.), Problems and Projects*, 35–41.

Green, P.E. and Rao, V.R. (1971). Conjoint measurement for quantyfying judgemental data. **8**.

Green, P.E. and Srinivasan, V. (1978). Conjoint analysis in consumer research: Issues and outlook. **5**.

Guo, S. and Sanner, S. (2010). Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *International Conference on Artificial Intelligence and Statistics*, 289–296, AISTATS.

Guo, S., Sanner, S. and Bonilla, E.V. (2010). Gaussian process preference elicitation. In *Advances in Neural Information Processing Systems*, vol. 23, 262–270, Morgan Kaufmann Publishers Inc.

Gusfield, D. and Irving, R.W. (1989). *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA.

Harper, F.M. and Konstan, J.A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, **5**, 19:1–19:19.

Haussler, D. (1989). Learning conjuctive concepts in structural domains. *Machine Learning*, **4**, 7–40.

Herbrich, R., Minka, T. and Graepel, T. (2007). Trueskill: a bayesian skill rating system. In *Advances in neural information processing systems*, 569–576.

Herlocker, J.L., Kostan, J.A., Borchers, A. and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference*, ACM Press.

Herlocker, J.L., Kostan, J.A. and Riedl, J.T. (2004a). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. **5**.

Herlocker, J.L., Kostan, J.A., Terveen, L.G. and Riedl, J.T. (2004b). Evaluating collaborative filtering recommender systems. **22**.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. **22**.

Holloway, H.A. and White, C.C.I. (2003). Question selection for multiattribute decision-aiding. **148**.

Houlsby, N., Huszar, F., Ghahramani, Z. and Hernández-lobato, J.M. (2012). Collaborative Gaussian processes for preference learning. In *Advances in Neural Information Processing Systems*, 2096–2104, NIPS.

Huang, Z., Chen, H. and Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. **22**.

IBM (2017). CPLEX optimizer.

Iyengar, V.S., Lee, J. and Campbell, M. (2001). Q-eval: Evaluating multiple attribute items using queries. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, Tampa, FL.

Jannach, D. (2006). Finding preferred query relaxations in conent-based recommenders. In *Proceedings of IEEE Intelligent Systems Conference*, IEEE Press, Westminster, UK.

Jannach, D., Zanker, M., Felfernig, A. and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edn.

Jin, R., i, L. and Zhai, C. (2006). A study of mixture models for collaborative filtering. **9**.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, Springer-Verlag, Chemnitz, Germany.

Kahneman, D. and Tversky, A. (1984). Choices, values, and frames. *American psychologist*, **39**, 341.

Keeney, R.L. and Raiffa, H. (1993). *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press.

Keiningham, T.L., Cooil, B., Aksoy, L., Andreassen, T.W. and Weiner, J. (2007). The value of different customer satisfaction and loyalty metrics in predicting customer retention, recommendation, and share-of-wallet. *Managing service quality: An international Journal*, **17**, 361–384.

Klotz, E. and Newman, A.M. (2013). Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, **18**, 18 – 32.

Koenemann, J. and Belkin, N.J. (1996). A case for interaction: A study of interactive information retrieval behavior and effectiveness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 205–212, ACM, Vancouver, BC.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press.

Komatsu, L.K. (1992). Recent views of conceptual structure. *Psychological Bulletin*, **112**, 500–526.

Koprinska, I., Poon, J., Clark, J. and Chan, J. (2007). Learning to classify e-mail. **177**.

Koren, Y., Bell, R. and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, **42**, 30–37.

Koriche, F. and Zanuttini, B. (2010). Lerning conditional preference networks. **174**.

Kőszegi, B. and Rabin, M. (2006). A model of reference-dependent preferences. *Quarterly Journal of Economics*, 1133–1165.

Lahaie, S.M. and Parkes, D.C. (2004). Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, 180–188.

Lang, J. (2004). Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, **42**, 37–71.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 5th SIAM International Conference on Data Mining*, Newport Beach, CA.

Lin, W. (2002). Efficient adaptive-support association rule mining for recommender systems. **6**.

Linden, G., Smith, B. and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. **7**.

Lipton, R.J., Markakis, E., Mossel, E. and Saberi, A. (2004). On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic commerce*, 125–131, ACM.

Loh, W.Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **1**, 14–23.

Mahmood, T. and Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, 73–82, ACM.

Maimon, O. and Rokach, L. (2005). *The Data Mining and Knowledge Discovery Handbook*. Springer.

Maltz, D. and Ehrlich, K. (1995). Pointing the way: Active collaborative filtering. In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*.

Manning, C.D., Raghavan, P. and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambrdge University Press.

Mardani, A., Jusoh, A., Nor, K.M., Khalifah, Z., Zakwan, N. and Valipour, A. (2015). Multiple criteria decision-making techniques and their applications - a review of the

literature from 2000 to 2014. *Economic Research-Ekonomska Istraivanja*, **28**, 516–571.

Marinho, L.B., Hotho, A., Jäschke, R., Nanopoulos, A., Rendle, S., Schmidt-Thieme, L., Stumme, G. and Symeonidis, P. (2012). *Recommender systems for social tagging systems*. Springer Science & Business Media.

Mas-Colell, A., Whinston, M.D. and Green, J.R. (1995). *Microeconomic Theory*. Oxford University Press, New York.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI.

McCarthy, K., Reilly, J., Smyth, B. and McGinty, L. (2005). Generating diverse compound critiques. **24**.

Medin, D.L. (1989). Concepts and conceptual structure. *American Psychologist*, **44**, 1469–1481.

Mehryar, M., Rostamizadeh, A. and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

Miller, G.A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. **63**.

Mourad Baïou, M.B. (2002). The stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research*, 485–503.

Mullainathan, S. (2002). Thinking through categories. *Unpublished Manuscript*.

Mullainathan, S. (2008). Coarse thinking and persuasion. *Quarterly Journal of Economics*, **123**, 577–619.

Murphy, K.P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Ng, A.Y. and Russell, A.J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, 663–670, Morgan Kaufmann Publishers Inc., Stanford, CA, USA.

Nguyen, T.V., Karatzoglou, A. and Baltrunas, L. (2014). Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 63–72, ACM.

Nisan, N. and Ronen, A. (2001). Algorithmic mechanism design. *Games and Economic behavior*, **35**, 166–196.

Öztürk, M., Pirlot, M. and Tsoukias, A. (2011). Representing preferences using intervals. *Artificial Intelligence*, **175**, 1194–1222.

Pardalos, P.M., Migdalas, A. and Pitsoulis, L. (2008). *Pareto optimality, game theory and equilibria*, vol. 17. Springer Science & Business Media.

Patrascu, R., Boutilier, C., Das, R., Kephart, J.O., Tesauro, G. and Walsh, W.E. (2005). New approaches to optimization and utility elicitation in autonomic computing. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, vol. 1, Pitsburgh, Pensylvania.

Pazzani, M.J. (1999). A framework for collaborative, content-based and demograpchic filtering. **13**.

Pazzani, M.J. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. **27**.

Pazzani, M.J. and Billsus, D. (2007). Content-based recommendation systems.

Prestwich, S., Rossi, F., Venable, B. and Walsh, T. (2005). Constraint-based preferential optimization. In *Proceedings of the 20th National Conference on Artificial Intelligence*, vol. 5, AAAI, Pitsburgh.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco.

Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J. and Riedl, J. (2002). Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, ACM, San Francisco.

Rasmussen, C.E. (2006). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning, The MIT Press, Cambridge, MA, USA.

Regan, K. and Boutilier, C. (2009). Regret-based reward elicitation for markov decision processes. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 444–451, AUAI Press.

Reilly, J., Zhang, J., McGinty, L., Pu, P. and Smyth, B. (2007). Evaluating compound recommenders: A real-user study. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, 114–123, ACM.

Resnick, P. and Varian, H.R. (1997). Recommender systems. *Communications of the ACM*, **40**, 56–58.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, 175–186, ACM, New York, NY, USA.

Ricci, F., Rokach, L., Shapira, B. and Kantor, P.B. (2010). *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edn.

Rosch, E. and Lloyd, B. (1978). *Cognition and Categorization*. Lawrence Erlbaum Associates, Mahwah, NJ.

Rubens, N., Elahi, M., Sugiyama, M. and Kaplan, D. (2015). *Active Learning in Recommender Systems*, 809–846. Springer US, Boston, MA.

Salo, A. and Hämäläinen, R.P. (2001). Preference ratios in multiattribute evaluation (prime) - elicitation and decision procedures with incomplete information. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, **31**, 533–545.

Salo, A. and Hämäläinen, R.P. (2004). Preference programming.

Salton, G. (1971). *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall.

Salton, G. and Buckley, C. (1997). Improving retrieval performance by relevance feedback. **24**.

Salton, G., Wong, A. and Yang, C.S. (1975). A vector space model for information retrieval. **18**.

Sarwar, B., Karypis, G., Konstan, J.A. and Riedl, J. (2000a). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, ACM.

Sarwar, B., Karypis, G., Konstan, J.A. and Riedl, J. (2000b). Application of dimensionality reduction in recommender systems – a case study. In *Proceedings of the ACM WebKDD Workshop*.

Sarwar, B., Karypis, G., Konstan, J.A. and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, ACM.

Savage, L.J. (1951). The theory of statistical decision. *Journal of the American Statistical Association*, **46**, 55–67.

Schafer, J.B., Frankowski, D., Herlocker, J. and Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa and W. Nejdl, eds., *The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science*, vol. 4321, Springer-Verlag.

Shawe-Taylor, J. and Singer, Y. (2004). Towards a characterization of polynomial preference elicitation with value queries in combinatorial auctions. **3120**.

Simon, H.A. (1972). Theories of bounded rationality. In C.B. McQuire and R. Radner, eds., *Decision and Organization*, North-Holand Publishing Company.

Strugeon, E.G.L. (2014). Using value ranges to reduce user effort in preference elicitation. *Group Decision and Negotiation: A Process-Oriented View*, **180**, 211–218.

Toubia, O., Simester, D.I., Hauser, J.R. and Dahan, E. (2003). Fast polyhedral adaptive conjoint estimation. **22**.

Toubia, O., Hauser, J.R. and Simester, D. (2004). Polyhedral methods for adaptive choice-based conjoint analysis. **41**.

Tsang, E. (1993). *Foundations of Constraint Satisfaction*. Academic Press.

Tversky, A. and Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science*, **211**, 453–458.

Vanchinathan, H.P., Nikolic, I., De Bona, F. and Krause, A. (2014). Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems*, 225–232, ACM.

Viappiani, P. and Boutilier, C. (2009a). Optimal set recommendations based on regret. In *ITWP*.

Viappiani, P. and Boutilier, C. (2009b). Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, ACM.

Viappiani, P. and Boutilier, C. (2010). Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems*, 2352–2360.

von Neumann, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton.

von Wright, G.H. (1963). *The Logic of Preference: An Essay*. Edinburgh University Press.

Wald, A. (1950). Statistical decision functions.

Wang, J., de Vries, A.P. and Reinders, M.J.T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM.

Wang, T. and Boutilier, C. (2003). Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

Weiss, G. (2013). *Multiagent Systems*. MIT Press.

Widrow, B. and Stearns, S.D. (1985). *Adaptive signal processing*. Prentice-Hall.

Wilson, N. (2004). Extending CP-nets with stronger conditional preference statements. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, vol. 4, AAAI.

Wilson, R. and Keil, F. (1999). *The MIT Encyclopedia of the Cognitive Sciences*. The MIT Press, Cambridge, MA.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y. and Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM.

Zaíane, O.R. (2002). Building a recommender agent for e-learning systems. In *Proceedings of the International Conference on Computers in Education*, IEEE.

Zanker, M., Bricman, M., Gordea, S., Jannach, D. and Jessenitschnig, M. (2006). Persuasive online-selling in quality & taste domains. In *Proceedings of the 7th International Conference on Electronic Commerce and Web Technologies*, vol. 10, Springer.

Zanker, M., Jessenitschnig, M. and Schmid, W. (2010). Preference learning with soft constraints in constraint-based recommender systems. In *Proceedings of the 7th International Conference on Electronic Commerce and Web Technologies*, vol. 15, Springer.

Zinkevich, M.A., Blum, A. and Sandholm, T. (2003). On polynomial-time preference elicitation with value queries. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, vol. 10, ACM.