Constructive Preference Elicitation for Multiple Users with Setwise Max-margin

Stefano Teso[1], Andrea Passerini[2], and Paolo Viappiani[3]

[1] KU Leuven, Leuven, Belgium
[⋆] stefano.teso@gmail.com
[2] University of Trento, Trento, Italy
andrea.passerini@unitn.it
[3] Sorbonne Universités, UPMC Univ Paris 06
CNRS, LIP6 UMR 7606, Paris, France
paolo.viappiani@lip6.fr

Abstract. In this paper we consider the problem of simultaneously eliciting the preferences of a group of users in an interactive way. We focus on *constructive* recommendation tasks, where the instance to be recommended should be synthesized by searching in a constrained configuration space rather than choosing among a set of pre-determined options. We adopt a setwise max-margin optimization method, that can be viewed as a generalization of max-margin learning to sets, supporting the identification of informative questions and encouraging sparsity in the parameter space. We extend setwise max-margin to multiple users and we provide strategies for choosing the user to be queried next and identifying an informative query to ask. At each stage of the interaction, each user is associated with a set of parameter weights (a sort of alternative options for the unknown user utility) that can be used to identify "similar" users and to propagate preference information between them. We present simulation results evaluating the effectiveness of our procedure, showing that our approach compares favorably with respect to straightforward adaptations in a multi-user setting of elicitation methods conceived for single users.

1 Introduction

Preferences are a widely studied concept in artificial intelligence [17]; the design of effective methods for preference elicitation is a particularly important topic in order to support the development of personalized systems such as recommender systems, electronic commerce applications and personal agents.

Recently, a number of techniques have been proposed allowing to incrementally elicit the preferences of a user by asking specifically chosen questions. These methods include Bayesian elicitation techniques [4,12,23] and regret-based methods [5,24]. The advantage of the Bayesian approaches is that they can identify

[⋆] Part of this research was done while ST was at University of Trento, partially supported by CARITRO Foundation grant 2014.0372.

informative queries in a principled manner and as well handle inconsistencies in preference feedback, but they require computationally intensive Bayesian updates; on the other hand regret-based methods can efficiently deal with larger configuration spaces but assume that all preference information is "noiseless".

Recently, setwise max-margin optimization has been proposed [22] as a paradigm for elicitation that has the following distinctive characteristics: 1) it allows to determine informative queries, 2) it can efficiently deal with large configuration spaces, 3) it is robust to user inconsistencies in preference feedback, and 4) it can be coupled with regularization terms if sparsity is required.

The focus of works in preference elicitation has been so far on acquiring the preferences of a single user. However, we claim that real systems - such as electronic commerce websites - do not usually interact with a user in isolation, but may be accessed by several users at the same time. Moreover, typical users of a web application may only provide very little information to the system. This means that it is crucial to exploit as much as possible the available preference information and to leverage the knowledge about the preferences of similar users.

In this paper we consider the problem of preference elicitation in the case that a number of users are simultaneously present. We focus on *constructive* recommendation problems, where the task is that of arranging novel configurations subject to feasibility constraints and user preferences, rather than selecting an item among a set of candidates. This setting rules out standard collaborative filtering techniques [21], where recommendations are propagated between users based on shared ratings over the same or similar objects. We instead rely on a notion of similarity in model space, i.e. similar users have similar utility functions, and propagate information while simultaneously learning user utilities. We extend setwise max-margin preference elicitation to the multi-user setting, by: i) defining a user similarity as a kernel over user utility models; ii) measuring the reliability of the learned model for each user; iii) defining for each user an aggregate utility function combining her utility model with those of the other users, weighted by their respective reliability and similarity to the user being recommended. We show how to incorporate these aspects in the setwise max-margin optimization problem, while retaining the formulation as mixed integer-linear problem (MILP) which allows for efficient computation.

Our experimental evaluation on both syntetic and real datasets shows how a simple procedure iteratively querying the least elicited user succeeds in improving recommendation quality with respect to independent elicitation of users.

2 Related Work

The basic idea of max-margin is as follows. The utility (often assumed to be linear) is determined by a set of parameter weights (unknown to the system). The currently known preferences of the user are encoded by a set of inequalities (typically stating the preference for an alternative over another one) on the feasible parameters; a shared non-negative margin is introduced as a decision variable that is maximized in the objective function. Noisy feedback can be

addressed by relaxing the constraints using slack variables and adding a penalty term in the objective for violated constraints.

This intuition has been adopted for preference learning by different authors in essentially the same way. In particular, Gajos and Weld [10] proposed the use of maxmargin optimization for learning preferences in the context of personalized user interfaces using a volumetric heuristic to choose the next question to ask. More recently, maxmargin methods have been used to assess preferences expressed in terms of Choquet capacities [1]. These works are limited by the lack of a principled way to determine informative queries and offer low scalability.

In our work we follow the ideas of [22] that extends maxmargin to produce a set of solutions (instead of a single one); in this way we can use such a set to devise a query to ask to user. *Setwise maxmargin* can then provide an efficient method for interactive preference elicitation handling user inconsistencies (the preference reported by the user may not be always true) and particularly suited to large configuration spaces. One main advantage is that determination of the next question (that is conflated into the problem of generating a set of diverse recommendations as in [24,23] in different paradigms) is the output of an optimization problem and therefore much more scalable that ad-hoc heurstics that need to iterate over available items.

While preference elicitation is a well studied topic in the community of artificial intelligence and algorithmic decision theory, the elicitation of preferences in a multi user setting is still underexplored, with the exception of [13]. However some recent works in the computational social choice community [8,25,15,3] have considered the problem of eliciting interactively the preferences of several users (agents) in order to determine a choice for the group. The difference with our work is that these approaches aim at establishing a best choice for the whole community (according to a voting rule that is fixed in advance); instead we wish to make recommendations that are personalized to each user while exploiting similarity between users' preferences. Some authors have instead considered how to combine interactive elicitation with collaborative filtering for predicting ratings given to items [9]; however this differs form our setting as we consider multi attribute utilities.

The idea of pooling together information about related learning tasks is not new. Our work is related to multi-task approaches (see for instance [2]), where information (data or parameters) is transferred beteween similar tasks to reduce the labelling effort required to achieve good generalization. Like in our work, task similarities are often expressed with a kernel function [20]. The multi-task active learning approach of Saha et al. [19] is perhaps the most closely related: in both methods the task similarity is estimated during the learning process. However, Saha et al. assume that labelled examples are received from some external source, and therefore do not propose any query selection strategy. On the contrary, we rely on the proven setwise max-margin approach for selecting informative queries especially designed for interaction with human decision makers.

3 Background

Notation. We indicate scalars x in italics, column vectors $\boldsymbol{x}$ in bold, and sets $\mathcal{X}$ in calligraphic letters. Important scalar constants N, M, K are upper-case. The inner product between vectors is written as $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_i w_i x_i$, the Euclidean ($\ell_2$) norm as $\|\boldsymbol{x}\| := \sqrt{\sum_i x_i^2}$, and the ℓ_1 norm as $\|\boldsymbol{x}\|_1 := \sum_i |x_i|$. We abbreviate the set $\{\boldsymbol{w}_i^u\}_{i=1}^K$ as $\{\boldsymbol{w}_i^u\}$ whenever the range for index i is clear from the context, and the set $\{1, \ldots, n\}$ as $[n]$. Table 1 summarizes the most frequently used symbols.

$N, M \in \mathbb{N}$	Number of attributes and users, respectively
$K \in \mathbb{N}$	Cardinality of the query sets
$\mathcal{X} \subseteq \{0,1\}^N$	Set of feasible configurations
$\boldsymbol{w}_*^u \in \mathbb{R}_+^N$	True preferences of user $u \in [M]$
$\boldsymbol{x}_*^u \in \mathcal{X}$	One of the configurations most preferred by user u
$\boldsymbol{w}_1^u, \ldots, \boldsymbol{w}_K^u \in \mathbb{R}_+^N$	Estimated preferences of user u
$\boldsymbol{x}_1^u, \ldots, \boldsymbol{x}_K^u \in \mathcal{X}$	Query set made to user u
$\boldsymbol{x}^u \in \mathcal{X}$	Recommendation made to user u
$v(u) \geq 0$	Variability within $\{\boldsymbol{w}_i^u\}$
$k(u, y) \geq 0$	Similarity between $\{\boldsymbol{w}_i^u\}$ and $\{\boldsymbol{w}_i^y\}$
$\boldsymbol{\alpha} := (\alpha, \beta, \gamma) \in \mathbb{R}_+^3$	Hyperparameters of the MU-SWMM algorithm

Table 1. Notation used throughout the paper.

Constructive setting. We consider a feasible set of products $\mathcal{X}$ populated by multi-attribute configurations $\boldsymbol{x} = (x_1, \ldots, x_N)$ over N attributes. In this presentation we will concentrate on 0-1 attributes only, a common choice in the preference elicitation literature [12,23]. Categorical attributes can be handled by using a one-hot encoding. Linearly dependent numerical attributes can be dealt with too; we refer the reader to the detailed discussion in [22] for space constraints. In contrast to standard recommendation, the set of products $\mathcal{X}$ is not explicitly provided, but rather defined by a set of hard (feasibility) constraints. These are assumed to be linear in the attributes. This setup is rather general, and naturally allows to encode both arithmetical and logical constraints. For instance, under the usual mapping *true* $\mapsto 1$ and *false* $\mapsto 0$, the logical disjunction between two 0-1 attributes $x_1 \vee x_2$ can be written as $x_1 + x_2 \geq 1$. Similarly, logical implication $x_1 \Rightarrow x_2$ can be translated to $(1 - x_1) + x_2 \geq 1$.

Following previous work on preference elicitation [12,23], user preferences are modeled as additive utility functions [14]. The *true* user preferences are represented by a non-negative weight vector $\boldsymbol{w}_* \in \mathbb{R}_+^N$, and the utility (i.e. subjective quality) of a feasible configuration $\boldsymbol{x}$ is given by the inner product $\langle \boldsymbol{w}_*, \boldsymbol{x} \rangle$. The *true* most preferred configurations, i.e. with maximal true utility, are analogously indicated as $\boldsymbol{x}_*$. In the remainder all weight vectors (both true and estimated) will be assumed to be non-negative and bounded, i.e. for every

Algorithm 1 The SWMM single-user algorithm. T is the maximum number of iterations, K the query set size, and $\boldsymbol{\alpha}$ the hyperparameters.

1: **procedure** SWMM (M, K, T, $\boldsymbol{\alpha}$)
2: $\mathcal{D} \leftarrow \varnothing$
3: **for** $t = 1, \ldots, T$ **do**
4: $\{\boldsymbol{w}_i\}, \{\boldsymbol{x}_i\} \leftarrow$ SOLVE_OP1($\mathcal{D}, K, \boldsymbol{\alpha}$)
5: *the user selects* $\boldsymbol{x}^+$ *from* $\{\boldsymbol{x}_i\}$
6: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}^+ \succcurlyeq \boldsymbol{x}^- \ : \ \boldsymbol{x}^-$ was not selected$\}$
7: $\boldsymbol{w}, \boldsymbol{x} \leftarrow$ SOLVE_OP1($\mathcal{D}, 1, \boldsymbol{\alpha}$)

attribute $z \in [N]$ there exist two finite non-negative constants $w_z^\top$ and $w_z^\perp$ that bound w_z from above and below, respectively. Bipolar preferences, i.e. user dislikes, can be modeled by negated attributes $1-x_z$, $z \in [N]$, if needed.

The single-user SWMM *algorithm.* The true user preferences $\boldsymbol{w}_*$ are not directly observed, and must be estimated. The SWMM algorithm tracks an estimate of the preferences at all times, iteratively improving it through user interaction.

The pseudocode of SWMM is reported in Algorithm 1. At every iteration, the algorithm selects K query configurations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K \in \mathcal{X}$ based on the previously collected user feedback $\mathcal{D}$ (line 4). The query set $\{\boldsymbol{x}_i\}$ is presented to the user, who is invited to select a most preferred configuration $\boldsymbol{x}^+$ from the K alternatives (line 5). The cases where the user selects a sub-par item are accounted for in the mathematical formulation, as discussed later on. The user choice is interpreted as a set of pairwise ranking constraints $\{\boldsymbol{x}^+ \succcurlyeq \boldsymbol{x}^- \ : \ \boldsymbol{x}^-$ was not selected$\}$, and added to $\mathcal{D}$[4] (line 6). At this point, a recommendation $\boldsymbol{x}$ is computed by leveraging all user feedback and presented to the user (line 7). If the user is satisfied with the suggested product, the procedure ends. Otherwise it is repeated, up to a maximum number of rounds T.

The primary goal of any preference elicitation system is to recover a satisfactory recommendation with minimal user effort. The choice of queries is crucial for reaching this goal [4]: the number of queries that can be afforded is small, thus every query should be chosen to be as "informative" as possible. Bayesian approaches to preference elicitation [12,23] model uncertainty about user preferences as a probability distribution over utility weights $\boldsymbol{w}$, and select queries that maximize expected informativeness, as measured by expected value of information (EVOI) [7] or its approximations. However, even the approximate strategies for EVOI maximization [23] are extremely time consuming and cannot scale to fully constructive scenarios as the ones we are dealing with here [22].

The SWMM query selection strategy addresses this problem by taking a space decomposition perspective inspired by max-margin ideas. The algorithm jointly learns a set of weight vectors, each representing a candidate utility function, and

[4] In [22], the authors convert user choices to pairwise ranking constraints using a custom procedure. Here we opted for a straightforward winner-vs-others representation, as described in the main text. This modification did not appear to significantly alter the performance of the SWMM algorithm in our simulations (data not show).

a set of candidate configurations, one for each weight vector, maximizing diversity between the vectors, consistency with the available feedback, and quality of each configuration according to its corresponding weight vector.

More formally, user preferences are estimated by a *set* of K weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K \in \mathbb{R}^N_+$. Each weight vector $\boldsymbol{w}_i$ is required to agree with all collected feedback $\mathcal{D}$. More precisely, the weights $\{\boldsymbol{w}_i\}$ are chosen as to provide the largest possible separation margin, i.e. for all $i \in [K]$ and $(\boldsymbol{x}^+ \succcurlyeq \boldsymbol{x}^-) \in \mathcal{D}$ the utility difference $\langle \boldsymbol{w}_i, \boldsymbol{x}^+ - \boldsymbol{x}^- \rangle$ should be as large as possible. Mistakes are absorbed by slack variables ε, as customary. Query configurations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ are chosen according to two criteria: each $\boldsymbol{x}_i$ should have maximal utility with respect to the associated weight vector $\boldsymbol{w}_i$, and the K products should be as diverse as possible. Diversity is encouraged by requiring that each weight vector $\boldsymbol{w}_i$ separates its associated configuration $\boldsymbol{x}_i$ from all the others configurations in the query set with a high margin, i.e. for all $i, j \in [K]$ with $i \neq j$, $\langle \boldsymbol{w}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle$ should be larger than the margin.

The previous discussion leads directly to the *quadratic* version of the SWMM optimization problem over the variables $\mu, \{\boldsymbol{w}_i, \boldsymbol{x}_i, \boldsymbol{\varepsilon}_i\}$:

$$\max \ \mu - \alpha \textstyle\sum_{i=1}^{k} \|\boldsymbol{\varepsilon}_i\|_1 - \beta \sum_{i=1}^{k} \|\boldsymbol{w}_i\|_1 + \gamma \sum_{i=1}^{k} \langle \boldsymbol{w}_i, \boldsymbol{x}_i \rangle \tag{1}$$

$$\text{s.t.} \ \langle \boldsymbol{w}_i, \boldsymbol{x}_s^+ - \boldsymbol{x}_s^- \rangle \geq \mu - \varepsilon_{is} \qquad \forall i \in [k], \boldsymbol{x}_s^+ \succcurlyeq \boldsymbol{x}_s^- \in \mathcal{D} \tag{2}$$

$$\langle \boldsymbol{w}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle \geq \mu \qquad \forall i, j \in [k], i \neq j \tag{3}$$

$$\mu \geq 0, \ \boldsymbol{w}^\perp \leq \boldsymbol{w}_i \leq \boldsymbol{w}^\top, \ \boldsymbol{x}_i \in \mathcal{X}, \ \boldsymbol{\varepsilon}_i \geq 0 \qquad \forall i \in [k] \tag{4}$$

The non-negative variable $\mu \in \mathbb{R}_+$ is the separation margin. The objective has four parts: the first part drives the maximization of the margin μ; the second minimizes the total sum of the ranking errors $\{\boldsymbol{\varepsilon}_i\}$; the third one introduces an ℓ_1 regularizer encouraging sparsity of the learned weights; finally, the last part requires the configurations $\{\boldsymbol{x}_i\}$ to have high utility with respect to the associated $\{\boldsymbol{w}_i\}$. The hyperparameters $\alpha, \beta, \gamma \geq 0$ modulate the contributions of the various parts. We refer to this optimization problem as **OP1**.

Constraint 2 encourages consistency of the learned weights with respect to the collected user feedback; ranking mistakes are absorbed by the slack variables $\{\boldsymbol{\varepsilon}_i\}$. Constraint 3 enforces the generated configurations to be as diverse as possible with respect to the corresponding weight vectors. Finally, Constraint 4 ensures that all variables lie in the corresponding feasible sets.

Unfortunately the above optimization problem is *quadratic* (due to Constraint 3) and difficult to optimize directly. Here we use the tight *mixed-integer linear* formulation proposed in [22], which can be solved using off-the-shelf MILP solvers. In that paper, the MILP formulation was shown to perform well empirically, reaching or outperforming two state-of-the-art Bayesian approaches.

4 Multi-user setwise max-margin

Now we generalize the SWMM algorithm to simultaneously elicit the preferences of M users. Our goal is to exploit preferences shared by similar users for computing

Algorithm 2 The MU-SWMM algorithm. M is the number of users; T, K and $\boldsymbol{\alpha}$ are as in Algorithm 1.

procedure MU-SWMM (M, K, T, $\boldsymbol{\alpha}$)
  $v(u) \leftarrow 1,\ k(u,y) \leftarrow \mathbb{I}(u=y)$
  for $u = 1, \dots, K$ **do**
    $\{\boldsymbol{w}_i^u\} \leftarrow$ SOLVE_OP2$(u, 0, k, \varnothing, K, \boldsymbol{\alpha})$
    $\mathcal{D}^u \leftarrow \varnothing$
  for $t = 1, \dots, T$ **do**
    select $u \in \operatorname{argmin}_u |\mathcal{D}^u|$ *uniformly at random*
    $\{\boldsymbol{w}_i^u\}, \{\boldsymbol{x}_i\} \leftarrow$ SOLVE_OP2$(u, v, k, \mathcal{D}^u, K, \boldsymbol{\alpha})$
    user u *selects* $\boldsymbol{x}^+$ *from* $\{\boldsymbol{x}_i\}$
    $\mathcal{D}^u \leftarrow \mathcal{D}^u \cup \{\boldsymbol{x}^+ \succcurlyeq \boldsymbol{x}^- \ :\ \boldsymbol{x}^-$ was not selected$\}$
    update $v(\cdot)$ *and* $k(\cdot,\cdot)$ *based on* $\{\boldsymbol{w}_i^1\}, \dots, \{\boldsymbol{w}_i^M\}$ *using Eq. 5 and 6*
    $\boldsymbol{w}^u, \boldsymbol{x}^u \leftarrow$ SOLVE_OP2$(u, v, k, \mathcal{D}^u, 1, \boldsymbol{\alpha})$

queries and recommendations. As a consequence, the cognitive effort of query answering can be distributed (fairly) among users, while maintaining the same or better recommendation quality.

Our strategy, dubbed MU-SWMM, is outlined in Algorithm 2. At every iteration, MU-SWMM picks a user u to be queried, based on some criterion (discussed later on). Then it proceeds like SWMM, by selecting a query set $\{\boldsymbol{x}_i^u\}$ for user u, adding the feedback to $\mathcal{D}^u$, and suggesting a recommendation $\boldsymbol{x}^u$. Once a user has received a satisfactory recommendation, it is removed from the pool of selectable users and skipped at later iterations[5]. The algorithm iterates until all users are satisfied, or T iterations are reached.

We remark that only *one* user is queried at every iteration. The MU-SWMM algorithm relies on **OP2**, a modification of **OP1** where the utility of configurations is determined both by the estimated preferences of the selected user u, as well as those of similar (non-selected) users $y \neq u$. This "aggregate utility" takes into consideration both the degree of similarity of u to the other users y and how good their preference estimates are. This avoids interferences due to similar users with unreliable preference estimates.

In order to implement this strategy, we must solve several problems: how to quantify the quality of the estimated preferences of a user; how to measure the similarity between two users; how to appropriately inject the information about other users into SWMM optimization problem **OP1**; and finally how to select the user to be queried. We discuss these points separately.

Measuring how much we know about a user. Ideally, one could quantify how much is known about a user u by using the regret

$$\left(\max_{\boldsymbol{x}} \langle \boldsymbol{w}_*^u, \boldsymbol{x} \rangle\right) - \langle \boldsymbol{w}_*^u, \boldsymbol{x}^u \rangle$$

[5] This detail is omitted from Algorithm 2 for simplicity.

i.e. the difference in true utility between a true most preferred recommendation $\boldsymbol{x}_*^u$ and the current recommendation $\boldsymbol{x}^u$. Unfortunately, computing the regret requires to observe $\boldsymbol{w}_*^u$, so we must rely on surrogate measures.

A simple surrogate is given by the number of times a user was queried, i.e. the number of collected user responses $|\mathcal{D}^u|$. This quantity, however, may give a simplistic estimate: in general, two users who replied to the same number of queries may have widely different regrets, depending on how difficult their preferences are to learn. This is especially true when $\mathcal{D}^u$ is redundant, i.e. contains repeated or similar constraints.

Taking care of redundancy requires us to look at the geometry of the problem. In particular, we use the *spread* of the weight vectors $\{\boldsymbol{w}_i^u\}$ computed during the query selection. As informative feedback is added to $\mathcal{D}^u$, the space of optimal weight vectors shrinks, and so does the distance between the vectors $\boldsymbol{w}_1^u, \ldots, \boldsymbol{w}_K^u$. More formally, we define the spread of user u as follows:

$$v(u) := c \sum_{i \neq j} \|\boldsymbol{w}_i^u - \boldsymbol{w}_j^u\|^2 \tag{5}$$

where the constant c is chosen so that $v(u) \in [0, 1]$. In other words, the spread is simply the *empirical variance* of the estimated weights $\{\boldsymbol{w}_i^u\}$. Note that the spread is much less affected by redundant constraints than $|\mathcal{D}^u|$ [6].

Evaluating user similarity. The most general mechanism for defining similarities between objects are *kernels* [20]. In particular, two users are similar when their estimated preferences are. While a variety of kernels could be used, we propose using a Gaussian kernel, similar to [26,19]:

$$k(u, y) := \exp\left(-\tau \sum_{i,j} \|\boldsymbol{w}_i^u - \boldsymbol{w}_j^y\|^2\right) \tag{6}$$

Here $\tau \in \mathbb{R}_+$ is an "inverse temperature" parameter controlling the shape of the kernel. Note that, similarly to [19], the kernel is not fixed: rather, it is adapted dynamically as soon as new user weight estimates $\{\boldsymbol{w}_i^u\}$ are computed.

Transferring preferences across users. Given a user u, we want to alter its estimated utility $\langle \boldsymbol{w}_i^u, \boldsymbol{x} \rangle$, $i = 1, \ldots, K$, based on the preferences of other similar, well-known users. We propose the following aggregate utility:

$$(1 - v(u))\langle \boldsymbol{w}_i^u, \boldsymbol{x} \rangle + v(u) \sum_{y \neq u} (1 - v(y)) k(u, y) \langle \boldsymbol{w}^y, \boldsymbol{x} \rangle \tag{7}$$

This is the convex combination of the utility of user u (first term) and a weighted combination of the utilities of the other users. Intuitively, the more u is known

[6] A constraint repeated l times instantiates l slack variables in **OP1**, thus becoming "harder" by a factor of αl. The effect however is much softer than for $|\mathcal{D}^u|$.

(e.g. at the end of the elicitation procedure) the closer $v(u)$ is to zero, and the first term dominates, and vice versa. The contributions of the users $y \neq u$ are decided, again, by how much is known about them $(1 - v(y))$ and by their similarity to user u, measured by the kernel $k(u, y)$. This formulation implies that there is little influence from users whose preferences are not well known.

It is straightforward to introduce Eq. 7 into **OP1**. By rearranging the terms, the aggregate utility can be written as:

$$\langle a\boldsymbol{w}_i^u + \boldsymbol{b}, \boldsymbol{x} \rangle \text{ with } \begin{cases} a = (1 - v(u)) \\ \boldsymbol{b} = v(u) \sum_{y \neq u} (1 - v(y)) k(u, y) \boldsymbol{w}^y \end{cases}$$

Note that this is a linear transformation of the original, single-user utility. As shown in [22], it is easy to incorporate linear transformations of this kind into the SWMM optimization problem. In our case, we need to rewrite Constraint 4 as:

$$a\boldsymbol{w}^{\perp} + \boldsymbol{b} \leq \boldsymbol{w}_i \leq a\boldsymbol{w}^{\top} + \boldsymbol{b} \qquad \forall i \in [K]$$

We use **OP2** to refer to the modified optimization problem; in Algorithm 2 we use SOLVE_OP2$(u, v, k, \mathcal{D}^u, K, \boldsymbol{\alpha})$ to denote a solution of the problem with respect to user u, spread v, kernel k, input preferences $\mathcal{D}^u$, set cardinality K, and using $\boldsymbol{\alpha} = (\alpha, \beta, \gamma)$ as coefficients in the objective function.

The spread $v(\cdot)$, the user similarity $k(\cdot, \cdot)$, and the transformation parameters $(a, \boldsymbol{b})$ are adapted whenever new feedback is received. The update is computationally inexpensive: whenever user u provides a response, only $k(u, \cdot)$ (i.e. a single row of the Gram matrix) needs to be recomputed.

Fairness-based user selection. The missing piece is how to choose a user at every iteration. Many strategies may be adopted, depending on the objective. In multi-user preference elicitation we are most concerned about fair distribution of queries among users, so to minimize the individual cognitive effort. Therefore we propose to select the user u that received the least queries so far:

$$u \in \operatorname*{argmin}_u |\mathcal{D}^u|$$

with ties broken at random. Although other strategies can be conceived, this simple strategy was observed to work well in our simulations, as shown in Section 5. Additionally, we found empirically that it is surprisingly difficult to beat: all of the more sophisticated strategies we tested failed to improve on it (data not shown due to space constraints).

5 Empirical Analysis

We studied the behavior of MU-SWMM on a synthetic and a realistic preference elicitation tasks[7], both taken from [22]. Our goal is to provide empirical answers

[7] Our experimental setup is available at: `https://github.com/stefanoteso/musm-adt17`

to these research questions: (**Q1**) Does aggregating the utility of similar users, as per Eq. 7, reduce the cognitive effort required to produce good recommendations? (**Q2**) Is the number of queries $|\mathcal{D}^u|$ a reasonable user selection criterion? (**Q3**) How does the algorithm behave when there are no common preferences to be shared between users?

Our experimental setup follow closely the ones of [22,11]. We randomly generated 20 groups of $M = 20$ users each using a hierarchical sampling procedure. Each group was split into C clusters, with $\approx M/C$ users each. The users within a cluster are chosen to have similar preferences, simulating different sub-groups of users. For instance, in a PC recommendation scenario, there may be a cluster of users who prefer energy efficient laptops and a cluster of power users who need more capable machines. This cluster structure enables preference information to be transferred. Note that the clusters are *not known to* MU-SWMM *beforehand*: the algorithm estimates them dynamically from the collected user replies.

For each cluster, we sampled the centroid from a uniform distribution in $[1, 100]$. The true preferences of the users in the cluster were obtained by perturbing the centroid randomly according to a normal distribution of mean 0 and standard deviation $25/6$. As done in [22], we considered users with both sparse and dense weight vectors: for sparse users, 80% of the entries of the true weight vector $\boldsymbol{w}_*^u$ were set to zero.

The user responses were simulated with a Plackett-Luce model [18,16], where the probability of a particular answer is dictated by a Boltzmann distribution:

$$P(\text{user chooses } \boldsymbol{x}_i \text{ from } \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}) = \frac{\exp(\lambda \langle \boldsymbol{w}_*, \boldsymbol{x}_i \rangle)}{\sum_{j=1}^{K} \exp(\lambda \langle \boldsymbol{w}_*, \boldsymbol{x}_j \rangle)}$$

with λ fixed to 1 as in [22,11]. For $K = 2$, this model reduces to the classical Bradley-Terry model for pairwise ranking feedback [6].

Synthetic setting. The first experiment is performed on the synthetic problem introduced in [22]. In this setting the space of products $\mathcal{X}$ is taken to be the Cartesian product of r categorical attributes, each having r possible values. We use a one-hot encoding to represent products, for a total of r^2 0-1 variables. Here we focus on the $r = 4$ case with $r^2 = 16$ variables and $r^r = 256$ total products. While simple, this problem proved to be non-trivial [22].

We compared the MU-SWMM algorithm against a straightforward multi-user adaptation of SWMM where all users are elicited independently. We also included in the comparison an unrealistic variant of MU-SWMM where the user to be queried is selected according to the maximal *true* regret (i.e. assuming that an oracle gives us this information). We assessed the ability of MU-SWMM to propagate preference information between users by varying the number of clusters C in $\{1, 2, 5\}$. We also varied the query set size $K \in \{2, 3\}$. The kernel inverse temperature parameter τ was fixed to 2 in all experiments.

The results for sparse users can be found in Figure 1. All algorithms were run for $T = 100$ elicitation rounds, i.e. 5 queries per user on average (x-axis). We report the median performance of the three recommenders across the 20 groups.

The performance on a group is the average regret over all M users (y-axis). As above, the regret is simply the difference in true utility between a best product $\boldsymbol{x}_*$ and the actual recommendation $\boldsymbol{x}$, i.e. $\langle \boldsymbol{w}_*^u, \boldsymbol{x}_* - \boldsymbol{x} \rangle$.

The plots show clearly that when users are similar enough, transferring preferences across them (MU-SWMM, red line) is better than no transfer at all (SWMM, gray line). This result is not obvious, since the kernel $k(\cdot, \cdot)$ is estimated *dynamically* from the collected feedback to reflect the hidden cluster structure of the users. The regret-based user selection strategy (blue line) provides an upper bound on the performance of MU-SWMM.

Understandably, the amount of improvement depends on C. The simplistic $C = 1$ case showcases the potential of preference transfer: both multi-user methods converge much faster than SWMM. In the more realistic $C = 2$ case, MU-SWMM takes about *half* the number of queries than SWMM to reach zero median regret. In particular, the number of per-user queries drops from more than 5 to about 3.3 for $K = 2$, and from 4 to less than 2 for $K = 3$. For $C = 5$, i.e. 4 users per cluster, both MU-SWMM and the regret-based strategy are closer to the baseline. The results for dense users in Figure 2 follow the same trend, despite elicitation being more difficult in this case.

In general, MU-SWMM fares better than or similarly to the no-transfer baseline. Notably, enlarging the query set size K from 2 to 3 further improves the performance of MU-SWMM: more ranking constraints are collected at each iteration, thus improving the estimate of the kernel $k(\cdot, \cdot)$ and preference transfer among similar users. These results validate Eq. 7 and allow us to answer affirmatively to question **Q1**.

They also partially answer **Q2**. Clearly selecting the user with the minimal number of queries does improve on the baseline, as shown in Figures 1 and 2. In addition, we checked whether it distributes queries fairly among users. Indeed, in the $C = 2$, $K = 2$ case the std. dev. in dataset size $|\mathcal{D}^u|$ between users is rather small (1.47) for the MU-SWMM user selection strategy, and 2.58 for regret-based user selection. The other cases behave similarly (data not shown).

PC recommendation. In the second experiment we consider a realistic PC configuration task, as in [22]. The recommender is required to suggest a fully customized PC. A PC configuration is defined by 7 categorical attributes—type (laptop, desktop or tower), manufacturer, CPU model, monitor size, RAM amount, storage amount—and a linearly dependent numerical attribute, the price. The attributes are mutually constrained via Horn clauses, expressing statements like "manufacturer X does not sell CPUs of brand Y", for a total of 16 Horn constraints. The product space has about 700,000 distinct configurations.

In this experiment we increased the number of average queries per user to 10, due to the very large number of products. We also restricted ourselves to sparse users, as in [22], which are more realistic in this setting: a typical customer will be indifferent about many aspects of a PC configuration.

The performance of the three methods, with $K = 2$, can be found in Figure 3. Again, MU-SWMM behaves better than the baseline for all values of $C \in \{1, 2, 5\}$, while the degree of improvement depends on C. For instance, for $C = 2$ the regret

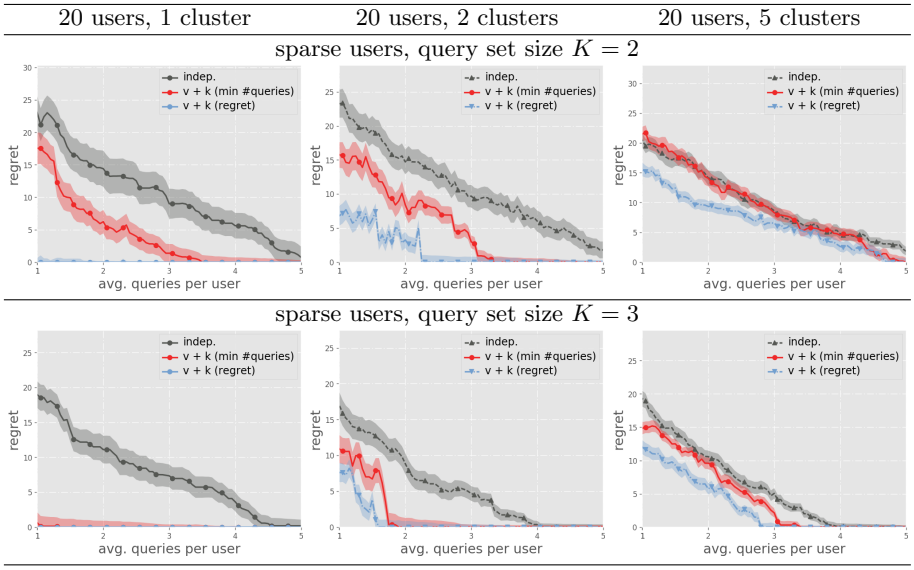

Fig. 1. Results for the synthetic setting with $1, 2, 5$ clusters of *sparse* users, for $K = 2, 3$. The three lines represent the average regret over users of the SWMM baseline (grey line), MU-SWMM with the fair user selection strategy (red), and MU-SWMM with the irrealistic regret-based user selection strategy (blue). The shaded area represents the standard deviation. Best viewed in color.

achieved after 10 queries per user is much closer to the performance upper bound (blue line) than to the baseline (gray). These results highlight that, despite the complexity of this recommendation task, user similarity is still estimated sufficiently well for preference transfer to work well.

Worst-case behavior. To answer **Q3**, we studied MU-SWMM when there is only one user in each cluster. This artificial setting is intended to verify the robustness of MU-SWMM against missing user similarities. We ran both the synthetic recommendation (for both sparse and dense users) and the PC recommendation settings (sparse users only) with $C = M = 20$ and $K = 2$; results in Figure 4.

In the two most difficult settings (i.e. synthetic with dense users and PC), MU-SWMM does not perform worse than the baseline. Unfortunately, in the remaining case (synthetic with sparse users) MU-SWMM performs worse than the baseline, and so does the regret-based user selection strategy. This is probably due to the kernel estimation taking too long to converge to a suitable value, therefore propagating preferences between unrelated users. In order to avoid this kind of interference, it may make sense to reduce the effect of the kernel, especially during the first elicitation rounds, for instance by making the inverse temperature τ increase during the process. We will explore this possibility in a future work.

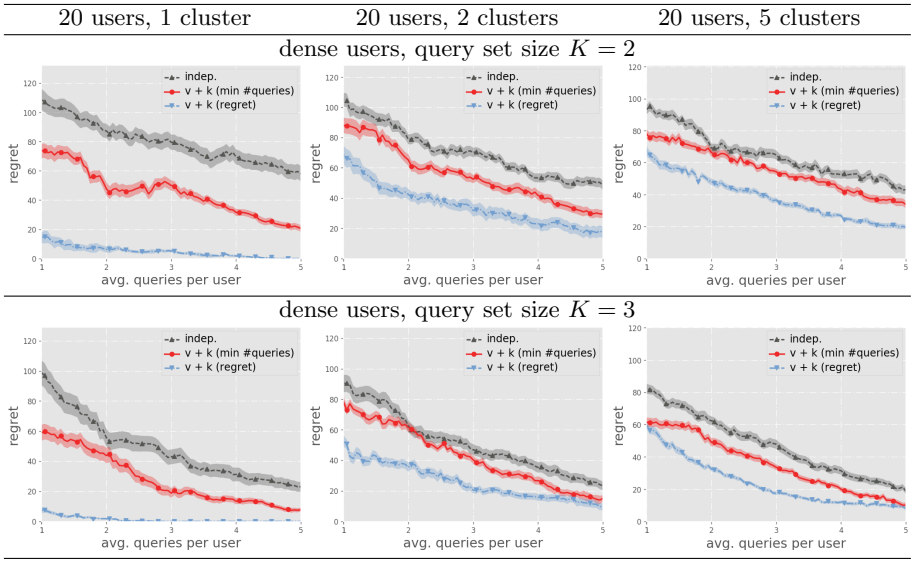

Fig. 2. Results for the synthetic setting with $1, 2, 5$ clusters of *dense* users, for $K = 2, 3$.

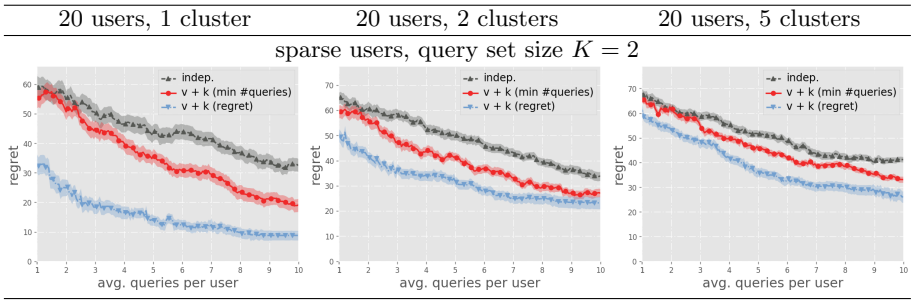

Fig. 3. Results for PC recommendation with $1, 2, 5$ clusters of *sparse* users, for $K = 2$.

6 Conclusion

We consider the problem of simultaneously eliciting the preferences of different users. The goal is to provide high utility recommendations to each user by considering all available preference information. A key idea is that, by asking informative questions and leveraging using similarities, we can produce good recommendations while distributing cognitive effort fairly among users.

This work tackles the problem of multi-user preference elicitation by generalizing setwise max-margin [22], thereby inheriting its core features. Namely, our method can gracefully deal with inconsistencies in user feedback and it can work in *constructive* recommendation scenarios, where the product to be recommended is synthesized by searching a large constrained configuration space rather than selected from a set of enumerated options.

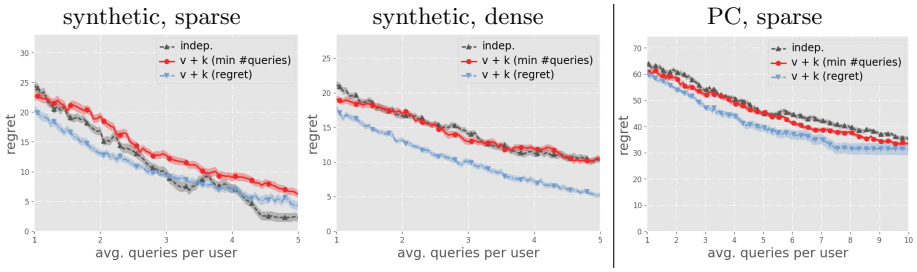

Fig. 4. Results for the $C = M = 20$ case for the synthetic setting with sparse users (left column), synthetic setting with dense users (middle column) and for PC recommendation with sparse users (right column). Here $K = 2$.

A key novelty of this work is that our method can effectively propagate preference information from user to user. Our method estimates a kernel (similarity) function between users utility models, as well as a reliability estimate of the learned model for each user. Preferences of similar, well-known users are combined into an aggregate utility function, which is incorporated into the setwise max-margin optimization problem while retaining an efficient MILP formulation.

We tested our approach on a synthetic and a realistic recommendation task. The experimental results show that our method is able to dynamically recover the similarity between users from their responses, and to exploit it to propagate preference information between more and less known users. When applied to sufficiently similar users, our method often performs much better than a straightforward adaptation of single-user setwise max-margin. On the other hand, it fares well in cases where users are not similar at all. Finally, our simple user selection strategy minimizes the cognitive effort of individual users by distributing queries fairly among them, and was shown to work well in practice.

Acknowledgements This work is partially supported by the ANR project CoCoRICo-CoDec ANR-14-CE24-0007-01 and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models). ST was partially supported by the CARITRO Foundation through grant 2014.0372.

References

1. Ah-Pine, J., Mayag, B., Rolland, A.: Identification of a 2-additive bi-capacity by using mathematical programming. In: Proceedings of ADT. pp. 15–29 (2013)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. Advances in neural information processing systems 19, 41 (2007)
3. Benabbou, N., Di Sabatino Di Diodoro, S., Perny, P., Viappiani, P.: Incremental preference elicitation in multi-attribute domains for choice and ranking with the borda count. In: Proceedings of SUM. pp. 81–95 (2016)

4. Boutilier, C.: A POMDP formulation of preference elicitation problems. In: Proceedings of AAAI/IAAI. pp. 239–246 (2002)
5. Boutilier, C., Patrascu, R., Poupart, P., Schuurmans, D.: Constraint-based optimization and utility elicitation using the minimax decision criterion. Artif. Intell. 170(8-9), 686–713 (2006)
6. Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 39(3/4), 324–345 (1952)
7. Chajewska, U., Koller, D., Parr, R.: Making rational decisions using adaptive utility elicitation. In: Proceedings of AAAI. pp. 363–369 (2000)
8. Dery, L.N., Kalech, M., Rokach, L., Shapira, B.: Reaching a joint decision with minimal elicitation of voter preferences. Information Sciences 278, 466–487 (2014)
9. Elahi, M., Ricci, F., Rubens, N.: Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. ACM TIST 5(1), 13:1–13:33 (2013)
10. Gajos, K., Weld, D.S.: Preference elicitation for interface optimization. In: Proceedings of UIST. pp. 173–182. ACM (2005)
11. Guo, S., Sanner, S.: Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In: Proceedings of AISTAT. pp. 289–296 (2010)
12. Guo, S., Sanner, S., Bonilla, E.V.: Gaussian process preference elicitation. In: Advances in Neural Information Processing Systems (NIPS). pp. 262–270 (2010)
13. Hines, G., Larson, K.: Efficiently eliciting preferences from a group of users. In: Proceedings of ADT. pp. 96–107 (2011)
14. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley and Sons, New York (1976)
15. Lu, T., Boutilier, C.: Robust approximation and incremental elicitation in voting protocols. In: Proceedings of IJCAI. pp. 287–293 (2011)
16. Luce, R.D.: Individual choice behavior: A theoretical analysis. Wiley (1959)
17. Pigozzi, G., Tsoukiàs, A., Viappiani, P.: Preferences in artificial intelligence. Ann. Math. Artif. Intell. 77(3-4), 361–401 (2016)
18. Plackett, R.L.: The analysis of permutations. Applied Statistics pp. 193–202 (1975)
19. Saha, A., Rai, P., III, H.D., Venkatasubramanian, S.: Online learning of multiple tasks and their relationships. In: Proceedings of AISTATS. pp. 643–651 (2011)
20. Scholkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press (2001)
21. Shi, Y., Larson, M., Hanjalic, A.: Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. ACM Comput. Surv. 47(1), 3:1–3:45 (May 2014)
22. Teso, S., Passerini, A., Viappiani, P.: Constructive preference elicitation by setwise max-margin learning. In: Proceedings of IJCAI. pp. 2067–2073 (2016)
23. Viappiani, P., Boutilier, C.: Optimal bayesian recommendation sets and myopically optimal choice query sets. In: Proceedings of NIPS. pp. 2352–2360 (2010)
24. Viappiani, P., Boutilier, C.: Regret-based optimal recommendation sets in conversational recommender systems. In: Proceedings of RecSys. pp. 101–108. ACM (2009)
25. Xia, L., Conitzer, V.: Determining possible and necessary winners given partial orders. Journal of Artificial Intelligence Research (JAIR) 41, 25–67 (2011)
26. Zhu, X., Lafferty, J., Ghahramani, Z.: Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In: ICML 2003 Workshop on the continuum from labeled to unlabeled data in machine learning and data mining (2003)