Collaborative Gaussian Processes for Preference Learning

Jose Miguel Hernández-Lobato
University of Cambridge
jmh233@cam.ac.uk

Neil Houlsby
University of Cambridge
nmth2@cam.ac.uk

Ferenc Huszár
University of Cambridge
fh277@cam.ac.uk

Zoubin Ghahramani
University of Cambridge
zoubin@eng.cam.ac.uk

15/01/2012

Abstract

We present a new model based on Gaussian processes (GPs) for learning pairwise preferences expressed by multiple users. Our approach combines supervised GP learning of user preferences with unsupervised dimensionality reduction techniques for multi-user systems. The model not only exploits collaborative information from the shared structure in user behavior, but may also incorporate user features if they are available. We introduce the *preference kernel* which simplifies inference in this domain, allowing us to perform approximate inference using a combination of expectation propagation and variational Bayes. Finally, in real-world applications it is desirable to learn the model from minimal supervised data, for this purpose we present an efficient active learning strategy for querying preferences. The proposed model with active learning performs favorably on real-world data against state-of-the-art multi-user preference learning algorithms.

1 Introduction

Preference learning is concerned with making inferences and predictions from data that consist of pairs of items and corresponding binary labels indicating item preferences. Preference data is becoming increasingly prolific, appearing in a large number of contexts, including medical assistive technologies (Birlutiu et al., 2010), graphical design (Brochu et al., 2007) and online recommendation and decision making systems (De Gemmis et al., 2009). Particularly with the advent of online retail and reccomendation, there has been a dramatic increase in the abundance of preference data, it has therefore become a rapidly growing subfield of Machine Learning and AI (Fürnkranz and Hüllermeier, 2010).

A popular approach to modelling preference data assumes the existence of a latent function f such that $f(\mathbf{x})$ gives the value of a particular item with feature vector $\mathbf{x}$. In particular, $f(\mathbf{x}_i) > f(\mathbf{x}_j)$ indicates that item i is preferred to item j. Bayesian methods can be used to learn f, for example Chu *et al.* model f independently for each user as a draw from a Gaussian process (GP) prior and approximate the posterior distribution for this function using the Laplace approximation. Such an approach allows f to come from a large class of continuous functions (Chu and Ghahramani, 2005).

However, this approach models each individual independently and does not carry any information between multiple users. In many applications e.g. online reccomendation, preference data will be available for many individuals and we would like to leverage similarities between users' preferences behavior. For example, one may discover that users' preferences for news articles can be summarized by their preferences towards latent themes such as sports, politics or technology. We may identify these common themes from the data across all users, and then at the individual level, only need to infer the user's relative interest in each of them.

Recent work has tackled this multi-user preference learning scenario (Birlutiu et al., 2010; Bonilla et al., 2010). However, the current approaches to the multi-user case have some limitations. For example the work of Bonilla *et al.* assumes that features are available for each user, and that users with similar features have similar behavior, even if the observed preferences indicates otherwise. Birlutiu *et al.* take an opposite approach, their model does not use any user features, but performs single-user learning, but tie information across users u with a hierarchical prior.

In summary, the current literature on probabilistic multi-user preference learning tackles one of two scenarios; when user features are available and they are useful for prediction, or when no features are available at all. Furthermore, these models both involve solving at least U Gaussian process problems, where U is the number of users, this cost becomes prohibitive with even modest U. We propose a more general model that can incorporate user features, if they are available, but does not require them, or can ignore them if they are not useful for making predictions, falling back on 'collaborative information'. We also show how to perform scalable inference with our model that can handle problems with large U.

Our approach is based upon two components: firstly, supervised GP utility function learning, as in (Chu and Ghahramani, 2005), that uses the observed preference data from the users to learn their individual preference functions. Secondly, unsupervised dimensionality-reduction techniques from collaborative filtering (Koren, 2008; Stern et al., 2009) are used to discover unobserved similarities in user behavior, such as the latent themes in news articles of sports, politics etc. These underlying themes allow similarities in user behavior to be discovered and exploited to assist when making predictions on new users. These predictions can be further augmented if user features are available.

This multi-user model is based on a connection between preference learning and binary classification with GP priors. We show that both problems are equivalent when the GP priors use a covariance function called the *preference kernel*. This connection simplifies the inference process and allows us to design more complex methods such as the proposed multi-user approach. For efficient approximate inference, we use a hybrid algorithm that combines expectation propagation and variational Bayes (Minka, 2001; Attias, 1999).

Finally, in real scenarios, it is desirable to learn user preferences using the least data possible. For this purpose we present BALD (Bayesian active learning by disagreement), a new active learning strategy for binary classification problems with GP priors. Using the preference kernel this approach can be applied directly to our preference learning model. BALD is based on the information theoretic approach to active learning and makes less approximations than other alternative strategies also based on this approach.

Our notation is summarized in Table 1. We review the single-user Gaussian process model for preference learning and describe our new approach using the preference kernel is Section 2. Our multi-user model is presented in Section 3. In Section 4 we describe our active sampling approach (BALD) for learning the model from the least possible data. Our hybrid inference scheme is presented in Section 5 and we compare our model and active learning scheme with other popular methods in Section 6. In Section 7 the proposed model and BALD are evaluated in experiments on

Symbol	**Meaning**
I	Number of items.
P	Total number of distinct pairs of items evaluated by all users.
U	Number of users.
M_u	Number of preference judgements made by user u.
D	Number of shared latent functions.
y	Preference label $\in \{0,1\}$.
$z_{i,u}$	Index for i-th item evaluated by user u.
$\mathcal{L}$	List, of length P, of pairs of items evaluated by all users.
$\mathcal{D}$	All multi-user data $\{\{z_{u,i}, y_{u,i}\}_{i=1}^{M_u}\}_{u=1}^{U}$.
$\mathbf{x}$	Item feature vector $\in \mathcal{X}$.
$\mathbf{u}$	User feature vector $\in \mathcal{U}$.
$\mathbf{X}$	Set of all feature vectors for the items.
$\mathbf{U}$	Set of all feature vectors for the users.
$\mathbf{G}^{(\mathcal{D})}$	$U \times P$ matrix of user latent functions g.
	Superscript $^{(\mathcal{D})}$ denotes matrix only evaluated at observed datapoints.
$\mathbf{W}$	$U \times D$ matrix of user weights w.
$\mathbf{H}$	$D \times P$ matrix of shared latent functions h.
$\mathbf{T}^{(\mathcal{D})}$	$U \times P$ 'target' matrix of preference judgements.
$\mathrm{H}[\mathcal{P}(\mathbf{x})]$	Differential entropy: $-\int_{\mathcal{X}} \mathcal{P}(\mathbf{x}) \log \mathcal{P}(\mathbf{x}) d\mathbf{x}$.
$\mathrm{H}[\mathcal{P}(y\|\mathbf{x})]$	Entropy of conditional distribution $\mathcal{P}(y\|\mathbf{x})$: $-\int_{\mathcal{Y}} \mathcal{P}(y\|\mathbf{x}) \log \mathcal{P}(y\|\mathbf{x}) dy$.
$\mathrm{h}(q)$	Binary entropy function: $-q \log q - (1-q) \log(1-q)$.

Figure 1: Summary of key notation.

simulated and a number of real-world datasets, including sushi-preference, electoral, movie-rating and geological data . In these experiments, we are able to outperform single-task methods (Chu and Ghahramani, 2005) and state-of-the-art methods for multi-user preference learning (Birlutiu et al., 2010; Bonilla et al., 2010). We conclude the paper in Section 8.

2 Pairwise Preference Learning as Special Case of Binary Classification

In preference learning models the likelihood takes a more complicated form to that in vanilla regression or binary classification. Therefore it is difficult to perform inference, and crude approximations, such as the Laplace approximation are used (Chu and Ghahramani, 2005; Bonilla et al., 2010). To implement effective inference in our model, we show that the problem of pairwise preference learning can be recast as a special case of binary classification. Let us consider two items i and j with corresponding feature vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. In the pairwise preference learning problem, we are given pairs of feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and corresponding class labels $y \in \{-1, 1\}$ such that $y = 1$ if the user prefers item i to item j and $y = -1$ otherwise. The task of interest is then to predict the class label for a new pair of feature vectors not seen before.

We begin by reviewing the single-user model of (Chu and Ghahramani, 2005). They address this problem by introducing a latent preference function $f : \mathcal{X} \mapsto \mathbb{R}$ such that $f(\mathbf{x}_i) > f(\mathbf{x}_j)$ whenever the user prefers item i to item j and $f(\mathbf{x}_i) < f(\mathbf{x}_j)$ otherwise. If we assume that the evaluations of f are corrupted with additive Gaussian noise with zero mean and variance σ_δ^2, we obtain the following likelihood function for f given $\mathbf{x}_i$, $\mathbf{x}_j$ and y

$$\mathcal{P}(y|\mathbf{x}_i, \mathbf{x}_j, f) = \Phi\left[y\frac{f(\mathbf{x}_i) - f(\mathbf{x}_j)}{\sqrt{2}\sigma_\delta}\right], \tag{1}$$

where Φ is the standard Gaussian cumulative distribution function. We can assume, without loss of generality, that $\sqrt{2}\sigma_\delta = 1$. The model is complete with a Gaussian Process prior over the latent preference function f:

$$f \sim GP(\mu, k) \tag{2}$$

This prior on f is combined with the likelihood function given by (1) and the posterior distribution for f is then used to make predictions on the user preferences.

Note, however, that the likelihood depends only upon the difference between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$. Let $g : \mathcal{X}^2 \mapsto \mathbb{R}$ be the latent function $g(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i) - f(\mathbf{x}_j)$. From now on this difference is the main parameter of our interest, and we recast the inference problem in terms of g and ignore f. When the evaluation of g is contaminated with additive standard Gaussian noise, the likelihood for g given $\mathbf{x}_i$, $\mathbf{x}_j$ and y is

$$\mathcal{P}(y|\mathbf{x}_i, \mathbf{x}_j, g) = \Phi[yg(\mathbf{x}_i, \mathbf{x}_j)]. \tag{3}$$

Since g is obtained from f via a linear operation, the Gaussian Process prior over f induces a Gaussian process prior over g. The mean μ_{pref}, and covariance function k_{pref} of this GP on g can be computed from the mean and covariance of f as follows:

$$\begin{aligned}
k_{pref}((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}_k, \mathbf{x}_l)) &= \mathrm{Cov}[g(\mathbf{x}_i, \mathbf{x}_j), g(\mathbf{x}_k, \mathbf{x}_l)] \\
&= \mathrm{Cov}\left[(f(\mathbf{x}_i) - f(\mathbf{x}_j)), (f(\mathbf{x}_k) - f(\mathbf{x}_l))\right] \\
&= \mathbb{E}\left[(f(\mathbf{x}_i) - f(\mathbf{x}_j)) \cdot (f(\mathbf{x}_k) - f(\mathbf{x}_l))\right] \\
&\quad - (\mu(\mathbf{x}_i) - \mu(\mathbf{x}_j))(\mu(\mathbf{x}_k) - \mu(\mathbf{x}_l)) \\
&= k(\mathbf{x}_i, \mathbf{x}_k) + k(\mathbf{x}_j, \mathbf{x}_l) - k(\mathbf{x}_i, \mathbf{x}_l) - k(\mathbf{x}_j, \mathbf{x}_k),
\end{aligned}$$

and

$$\begin{aligned}
\mu_{pref}(\mathbf{x}_i, \mathbf{x}_j) &= \mathbb{E}\left[g([\mathbf{x}_i, \mathbf{x}_j])\right] \\
&= \mathbb{E}\left[f(\mathbf{x}_i) - f(\mathbf{x}_j)\right] \\
&= \mu(\mathbf{x}_i) - \mu(\mathbf{x}_j).
\end{aligned} \tag{4}$$

We call k_{pref} the *preference kernel*. We now analyse further the properties of this kernel. The same kernel function can be derived from a large margin classification viewpoint Fürnkranz and Hüllermeier (2010). However, to our knowledge, the preference kernel has not been used previously for GP-based models.

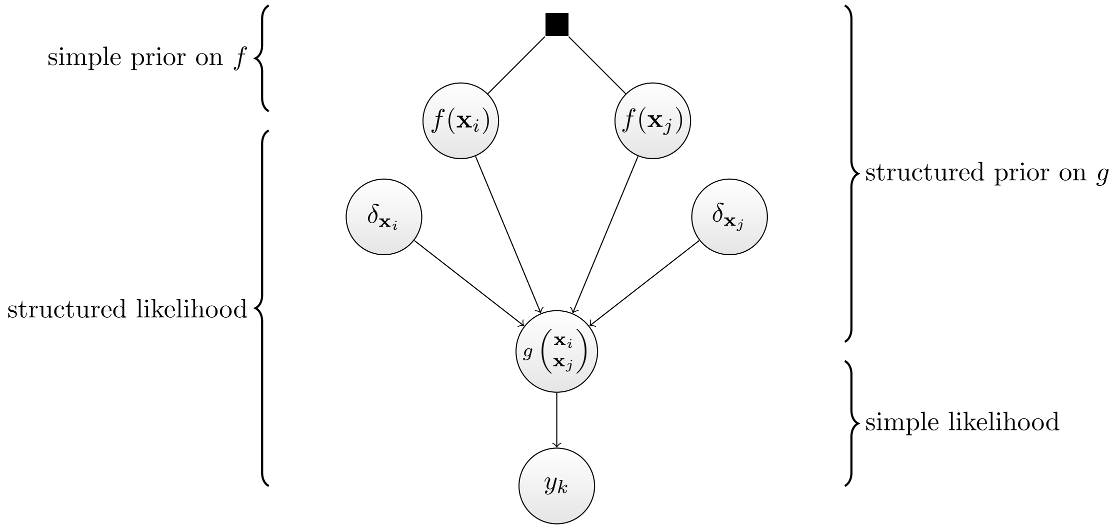

Figure 2: Generative model underlying the preference learning framework. *Left:* the original approach (Chu and Ghahramani, 2005) considers the latent preference function f as latent parameter, and the rest of the graphical model as a complex, structured likelihood. *Right:* Our approach reparametrises the problem in terms of g, and thus works with a simpler likelihood but with a more structured prior. The prior takes the form of a Gaussian Process with the preference judgement covariance function.

2.1 Properties of the Preference Kernel

Figure 2 illustrates the difference between the original approach where the quantity of central interest was f and our approach where the quantity of interest is g.

It is straightforward to show that the preference kernel, k_{pref} is positive semi-definite. We can also see how k_{pref} respects the anti-symmetry properties of preference learning, by computing the prior correlation between $g(\mathbf{x}_i, \mathbf{x}_j)$ and $g(\mathbf{x}_j, \mathbf{x}_i)$ as follows (assuming for brevity that $\mu_{pref} = 0$):

$$\begin{aligned}
\mathrm{Corr}(g(\mathbf{x}_i, \mathbf{x}_j), g(\mathbf{x}_j, \mathbf{x}_i)) &= \frac{k_{pref}((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}_j, \mathbf{x}_i))}{\sqrt{k_{pref}((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}_i, \mathbf{x}_j))}\sqrt{k_{pref}((\mathbf{x}_j, \mathbf{x}_i), (\mathbf{x}_j, \mathbf{x}_i))}} \\
&= \frac{k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{x}_j, \mathbf{x}_i) - k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_j, \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_j, \mathbf{x}_i)}\sqrt{k(\mathbf{x}_j, \mathbf{x}_j) + k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_j, \mathbf{x}_i) - k(\mathbf{x}_i, \mathbf{x}_j)}} \\
&= -1\,.
\end{aligned}$$

That is, the value at $(\mathbf{x}_i, \mathbf{x}_j)$ is perfectly anti-correlated with the value at $(\mathbf{x}_j, \mathbf{x}_i)$ under the prior. From this fact it can be shown that all elements g of the reproducing kernel Hilbert space (RKHS) corresponding to k_{pref} have the property that $g(\mathbf{x}_i, \mathbf{x}_j) = -g(\mathbf{x}_j, \mathbf{x}_i)$.

In summary, the combination of the likelihood in (3) with a GP prior based on this new preference kernel allows us to transform the problem of learning preferences into a GP binary classification problem on the space of features $\mathcal{X}^2$. This means that state-of-the art methods for GP binary classification, such as expectation propagation, can be trivially used for addressing pairwise preference learning problems. Furthermore, the simplified likelihood (3) allows us to easily implement complex methods such as the following multi-user approach.

3 A Multi-User Model for Preference Learning

Consider I items with feature vectors $\mathbf{x}_i \in \mathcal{X}$ for $i = 1, \ldots, I$. The single-user learning approach assumes an independent latent function for the u-th user, $g_u : \mathcal{X}^2 \mapsto \mathbb{R}$. Our approach to the multi-user problem is to assume common structure in the user latent functions. In particular, we assume a set of D shared latent functions, $h_d : \mathcal{X}^2 \mapsto \mathbb{R}$ for $d = 1, \ldots, D$, such that the user latent functions are generated by a linear combination of these functions, namely

$$g_u(\mathbf{x}_j, \mathbf{x}_k) = \sum_{d=1}^{D} w_{u,d} h_d(\mathbf{x}_j, \mathbf{x}_k)\,, \tag{5}$$

here $w_{u,d} \in \mathbb{R}$ is the weight given to function h_d for user u. We place a GP prior over the shared latent functions $h_1, \ldots, h_D$ using the preference kernel described in the previous section. This model allows the preferences of the different users to share some common structure represented by the latent functions $h_1, \ldots, h_D$. This approach is similar to unsupervised dimensionality reduction methods that are commonly used for addressing collaborative filtering problems Stern et al. (2009); Raiko et al. (2007).

We may extend this model further to the case in which, for each user u, there is a feature vector $\mathbf{u}_u \in \mathcal{U}$ containing information that might be useful for prediction. We denote by $\mathbf{U}$ the set of all the users' feature vectors, that is, $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_U\}$. The user features are incorporated now by placing a separate GP prior over the users weights. In particular, we replace the scalars $w_{u,d}$ in (5) with functions $w'_d(\mathbf{u}_u) : \mathcal{U} \to \mathbb{R}$. These weight functions describe the contribution of shared latent function h_d to the user latent function g_u as a function of the user feature vector $\mathbf{u}_u$.

In the multi-user setting we are given a list $\mathcal{L} = \{p_1, \ldots, p_P\}$ with all the *pairs* of items evaluated by the users, where $P \leq I(I-1)/2$ (the maximum number of pairs). The data consists of $\mathcal{L}$, the sets of feature vectors for the users $\mathbf{U}$ (if available) and the items $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_I\}$, and U different sets of preference judgements, one for each user, $\mathcal{D} = \{\{z_{u,i}, y_{u,i}\}_{i=1}^{M_u}\}_{u=1}^{U}$, where $z_{u,i}$ indexes the i-th pair evaluated by user u, $y_{i,u} = 1$ if this user prefers the first item in the pair to the second and $y_{i,u} = -1$ otherwise. M_u is the number of preference judgements made by the u-th user.

3.1 Probabilistic Description

The task of interest is to predict preference on unseen item pairs for a particular user. To do this we cast the model described above into a probabilistic framework. Let $\mathbf{G}$ be an $U \times P$ 'user-function' matrix, where each row corresponds to a particular user's latent function, that is, the entry in the u-th column and i-th row is $g_{u,i} = g_u(\mathbf{x}_{\alpha(i)}, \mathbf{x}_{\beta(i)})$ and $\alpha(i)$ and $\beta(i)$ denote respectively the first and second item in the i-th pair from $\mathcal{L}$. Let $\mathbf{H}$ be a $D \times P$ 'shared-function' matrix, where each row represents the shared latent functions, that is, the entry in the d-th row and i-th column is $h_{d,i} = h_d(\mathbf{x}_{\alpha(i)}, \mathbf{x}_{\beta(i)})$. Finally, we introduce the $U \times D$ weight matrix $\mathbf{W}$ such that each row contains a user's weights, that is, the entry in the u-th row and d-th column of this matrix is $w'_d(\mathbf{u}_u)$. Note that $\mathbf{G} = \mathbf{WH}$ represents equation (5) in matrix form. Let $\mathbf{T}$ be the $U \times P$ target matrix given by $\mathbf{T} = \text{sign}[\mathbf{G} + \mathbf{E}]$, where $\mathbf{E}$ is an $U \times P$ noise matrix whose entries are sampled i.i.d. from a standard Gaussian distribution and the function "sign" retains only the sign of the elements in a matrix. The observations $y_{u,i}$ in $\mathcal{D} = \{\{z_{u,i}, y_{u,i}\}_{i=1}^{M_u}\}_{u=1}^{U}$ are mapped to the corresponding entries of $\mathbf{T}$ using $t_{u,z_{u,i}} = y_{u,i}$. Let $\mathbf{T}^{(\mathcal{D})}$ and $\mathbf{G}^{(\mathcal{D})}$ represent the elements of $\mathbf{T}$ and $\mathbf{G}$ corresponding only to the available observations $y_{u,i}$ in $\mathcal{D}$. Then, the likelihood for $\mathbf{G}^{(\mathcal{D})}$ given $\mathbf{T}^{(\mathcal{D})}$ and conditional distribution for $\mathbf{G}^{(\mathcal{D})}$ given $\mathbf{H}$ and $\mathbf{W}$ are

$$\mathcal{P}(\mathbf{T}^{(\mathcal{D})}|\mathbf{G}^{(\mathcal{D})}) = \prod_{u=1}^{U}\prod_{i=1}^{M_u}\Phi[t_{u,z_{u,i}}g_{u,z_{u,i}}] \text{ and } \mathcal{P}(\mathbf{G}^{(\mathcal{D})}|\mathbf{W},\mathbf{H}) = \prod_{u=1}^{U}\prod_{i=1}^{M_u}\delta[g_{u,z_{u,i}} - \mathbf{w}_u\mathbf{h}_{\cdot,z_{u,i}}]$$

respectively, where $\mathbf{w}_u$ is the u-th row in $\mathbf{W}$, $\mathbf{h}_{\cdot,i}$ is the i-th column in $\mathbf{H}$ and δ represents a point probability mass at zero. We now select the priors for $\mathbf{W}$ and $\mathbf{H}$. We assume that each function $w'_1, \ldots, w'_D$ is sampled *a priori* from a GP with zero mean and specific covariance function. Let $\mathbf{K}_{\text{users}}$ be the $U \times U$ covariance matrix for entries in each column of matrix $\mathbf{W}$. Then

$$\mathcal{P}(\mathbf{W}|\mathbf{U}) = \prod_{d=1}^{D}\mathcal{N}(\mathbf{w}_{\cdot,d}|\mathbf{0},\mathbf{K}_{\text{users}})\,, \tag{6}$$

where $\mathbf{w}_{\cdot,d}$ is the d-th column in $\mathbf{W}$. If user features are unavailable, $\mathbf{K}_{\text{users}}$ becomes the identity matrix. Finally, we assume that each shared latent function $h_1, \ldots, h_D$ is sampled *a priori* from a GP with zero mean and covariance function given by a preference kernel. Let $\mathbf{K}_{\text{items}}$ be the $P \times P$ preference covariance matrix for the item pairs in $\mathcal{L}$. The prior for $\mathbf{H}$ is then

$$\mathcal{P}(\mathbf{H}|\mathbf{X},\mathcal{L}) = \prod_{j=1}^{D}\mathcal{N}(\mathbf{h}_j|\mathbf{0},\mathbf{K}_{\text{items}})\,, \tag{7}$$

where $\mathbf{h}_j$ is the j-th row in $\mathbf{H}$. The resulting posterior for $\mathbf{W}$, $\mathbf{H}$ and $\mathbf{G}^{(\mathcal{D})}$ is

$$\mathcal{P}(\mathbf{W},\mathbf{H},\mathbf{G}^{(\mathcal{D})}|\mathbf{T}^{(\mathcal{D})},\mathbf{X},\mathcal{L}) = \frac{\mathcal{P}(\mathbf{T}^{(\mathcal{D})}|\mathbf{G}^{(\mathcal{D})})\mathcal{P}(\mathbf{G}^{(\mathcal{D})}|\mathbf{W},\mathbf{H})\mathcal{P}(\mathbf{W}|\mathbf{U})\mathcal{P}(\mathbf{H}|\mathbf{X},\mathcal{L})}{\mathcal{P}(\mathbf{T}^{(\mathcal{D}}|\mathbf{X},\mathcal{L})}\,. \tag{8}$$

Given a new item pair p_{P+1}, we can compute the predictive distribution for the preference of the u-th user ($1 \leq u \leq U$) on this pair by integrating out the parameters $\mathbf{H}, \mathbf{W}$ and $\mathbf{G}^{(\mathcal{D})}$ as follows:

$$\begin{aligned}\mathcal{P}(t_{u,P+1}|\mathbf{T}^{(\mathcal{D})},\mathbf{X},\mathcal{L},p_{P+1}) = \int & \mathcal{P}(t_{u,P+1}|g_{u,P+1})\mathcal{P}(g_{u,P+1}|\mathbf{w}_u,\mathbf{h}_{\cdot,P+1}) \\ & \mathcal{P}(\mathbf{h}_{\cdot,P+1}|\mathbf{H},\mathbf{X},\mathcal{L},p_{P+1})\mathcal{P}(\mathbf{H},\mathbf{W},\mathbf{G}^{(\mathcal{D})}|\mathbf{T}^{(\mathcal{D})},\mathbf{X},\mathcal{L})\,d\mathbf{H}\,d\mathbf{W}\,d\mathbf{G}^{(\mathcal{D})}\,,\end{aligned} \tag{9}$$

where $\mathcal{P}(t_{u,P+1}|g_{u,P+1}) = \Phi[t_{u,P+1}g_{u,P+1}]$, $\mathcal{P}(g_{u,P+1}|\mathbf{w}_u,\mathbf{h}_{\cdot,P+1}) = \delta[g_{u,P+1} - \mathbf{w}_u\mathbf{h}_{\cdot,P+1}]$,

$$\mathcal{P}(\mathbf{h}_{\cdot,P+1}|\mathbf{H},\mathbf{X},\mathcal{L},p_{P+1}) = \prod_{d=1}^{D}\mathcal{N}(h_{d,P+1}|\mathbf{k}_\star^{\text{T}}\mathbf{K}_{\text{items}}^{-1}\mathbf{h}_d, k_\star - \mathbf{k}_\star^{\text{T}}\mathbf{K}_{\text{items}}^{-1}\mathbf{k}_\star) \tag{10}$$

$k_\star$ is the prior variance of $h_d(\mathbf{x}_{\alpha(P+1)}, \mathbf{x}_{\beta(P+1)})$ and $\mathbf{k}_\star$ is a P-dimensional vector that contains the prior covariances between $h_d(\mathbf{x}_{\alpha(P+1)}, \mathbf{x}_{\beta(P+1)})$ and $h_d(\mathbf{x}_{\alpha(1)}, \mathbf{x}_{\beta(1)}), \ldots, h_d(\mathbf{x}_{\alpha(P)}, \mathbf{x}_{\beta(P)})$.

In practice, computing (8) or (10) is infeasible and approximate inference has to be used. For this task, we propose to use a combination of expectation propagation (EP) Minka (2001) and variation

Bayes (VB) Attias (1999). Empirical studies show that EP obtains state-of-the-art performance in the related problem of GP binary classification Nickisch and Rasmussen (2008).

Typically, data with which to learn the model is limited, and furthermore the computational time required to train and make predictions can become large when the volume of data becomes very large (there is further discussion of computational complexity in Sections 5.4 and 6). Therefore, we want to learn about the users' preferences with the proposed model using the least amount of data possible, or equivalently, extract maximal useful information from a limited number of observations. Therefore we query users actively about their preferences on the most informative pairs of items Brochu et al. (2007). Next, we describe a novel method to implement this strategy. This method exploits the preference kernel and so may be trivially generalized to GP binary classification problems also.

4 Active Learning

The goal of active learning is to choose item pairs such that we learn the user's latent preference functions from the least data possible. For this purpose we derive a fast information-theoretic motivated active learning criterion that, as far as the authors are aware, makes the fewest approximations to this objective to date.

4.1 Information Theoretic Approach

Information theoretic approaches have become popular for deciding which data to acquire, or which data to retain when sub-sampling a large dataset. Information theoretic approaches to active learning are popular because they do not require prior knowledge of loss functions or test domains. The goal of information theoretic active learning is to reduce uncertainty in the model as fast as possible with the number of observed datapoints. Although this approach was proposed several decades ago (Lindley, 1956; Bernardo, 1979), it is not always straightforward to apply the criteria to complicated models such as Gaussian processes with infinite parameter spaces. Current solutions make approximations to achieve tractability, for example (Lawrence et al., 2002) computes approximate entropies, (Freund et al., 1997) provides a sampling-based approach, and (Tong and Koller, 2001) tackle the equivalent problem for non-probabilistic models. We return to this problem, and demonstrate how to apply a simple reformulation of the active learning problem to GP preference learning that provides a fast and accurate application of the full information theoretic criterion.

The central goal of information theoretic active learning is to reduce the number of possible hypotheses maximally fast, i.e. to minimize the uncertainty about the parameters using Shannon's entropy (Cover et al., 1991). Data points $\mathcal{D}'$ are selected that satisfy $\arg\min_{\mathcal{D}'} \mathrm{H}[p(\boldsymbol{\theta}|\mathcal{D}')] = -\int p(\boldsymbol{\theta}|\mathcal{D}') \log p(\boldsymbol{\theta}|\mathcal{D}') d\boldsymbol{\theta}$, where we use general notation $\boldsymbol{\theta}$ to denote the parameters of the model; for multi-user preference learning, these parameters correspond to the latent functions g for each user. Solving this problem in general is NP-hard. However, as is common in sequential decision making tasks a myopic (greedy) approximation is made (Heckerman et al., 1995). It has been shown that the myopic policy can perform near-optimally (Dasgupta, 2005; Golovin and Krause, 2010). Furthermore, in an data streaming setting, such as the querying user's preferences online, one is constrained to select datapoints sequentially, rather than in batch anyway. For preference learning (see Section 2), this implies identifying the new item features $\mathbf{x}_i$ and $\mathbf{x}_j$ that maximize

$$\mathrm{H}[\mathcal{P}(g|\mathcal{D})] - \mathbb{E}_{\mathcal{P}(y|\mathbf{x}_i,\mathbf{x}_j,\mathcal{D})}\left[\mathrm{H}[\mathcal{P}(g|y,\mathbf{x}_i,\mathbf{x}_j,\mathcal{D})]\right] \ , \tag{11}$$

Note that expectation over the unseen output y is required. Many works e.g. (MacKay, 1992; Krishnapuram et al., 2004; Lawrence et al., 2002) propose using this objective directly. However, parameter posteriors are often high dimensional and computing their entropies is usually intractable. Furthermore, for nonparametric processes the parameter space is infinite dimensional, so Equation (11) becomes poorly defined. To avoid griding parameter space (which is exponentially hard with dimensionality), or sampling (from which it is notoriously hard to estimate entropies without introducing bias (Panzeri S. and R.S., 2007)), these papers make Gaussian or low dimensional approximations and calculate the entropy of the approximate posterior. However, a second difficulty arises; if n new data points are available for selection, with $|\{-1,1\}| = 2$ possible values for y. Then $\mathcal{O}(2n)$ potentially expensive posterior updates are required to find the maximizer of (11); one for every available feature vector and possible class value. This is often too expensive in practice.

An solution arises if we note that the objective in (11) is equivalent to the conditional mutual information between y and g. Using this insight it is simple to show that the objective can be rearranged to compute entropies in y space:

$$\mathrm{H}[\mathcal{P}(y|\mathbf{x}_i,\mathbf{x}_j,\mathcal{D})] - \mathbb{E}_{\mathcal{P}(g|\mathcal{D})}\left[\mathrm{H}\left[\mathcal{P}(y|\mathbf{x}_i,\mathbf{x}_j,g)\right]\right] \ . \tag{12}$$

(12) overcomes the challenges we described for (11). Entropies are now calculated in output space, which usually has low dimension. For binary preferences, these are just entropies of Bernoulli variables. Furthermore g is now conditioned only on $\mathcal{D}$, so only $\mathcal{O}(1)$ posterior updates are required i.e. we only need to recompute the posterior once per datapoint selected, not for every possible datapoint under consideration.

Expression (12) also provides us with an intuition about the objective; we seek the $\mathbf{x}_i$ and $\mathbf{x}_j$ for which a) the model is marginally uncertain about y (high $\mathrm{H}[\mathcal{P}(y|\mathbf{x}_i,\mathbf{x}_j,\mathcal{D})]$) and b) the model is confident about the value of g at that location (low $\mathbb{E}_{\mathcal{P}(g|\mathcal{D})}\left[\mathrm{H}[\mathcal{P}(y|\mathbf{x}_i,\mathbf{x}_j,g])\right]$). This can be interpreted as seeking the $\mathbf{x}_i$ and $\mathbf{x}_j$ for which g, under the posterior, disagrees the most about the outcome. Therefore, we refer to this objective as Bayesian Active Learning by Disagreement (BALD). In the following section we show how (12) can be applied to binary classification problems with Gaussian processes. The proposed method is independent of the approach used for inference; something which does not hold in (MacKay, 1992; Krishnapuram et al., 2004; Lawrence et al., 2002) where the entropy calculation is built around the method for approximate inference.

4.2 Application to GP Preference Learning

Most approximate inference methods for the problem of binary classification with GPs produce a Gaussian approximation to the posterior distribution of f, the latent function of interest. In the binary GP classifier, the entropy of y given the corresponding value of f can be expressed in terms of the binary entropy function, $\mathrm{h}[f] = -f \log f - (1-f)\log(1-f)$. In particular,

$$\mathrm{H}[p(y|\mathbf{x},f)] = \mathrm{h}\left[\Phi(f(\mathbf{x})\right] \ .$$

Expectations over the posterior need to be computed. When using a Gaussian approximation to the posterior, for each $\mathbf{x}$, $f_{\mathbf{x}} = f(\mathbf{x})$ will follow a Gaussian distribution with mean $\mu_{\mathbf{x}}$ and

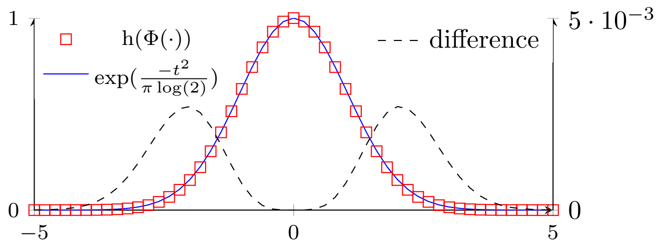

Figure 3: Analytic approximation ($\overset{1}{\approx}$) to the binary entropy of the error function by a squared exponential. The absolute error remains under $3 \cdot 10^{-3}$.

variance $\sigma_{\mathbf{x}}^2$. To compute (12) we have to compute two entropy quantities. The first term in (12), $\mathrm{H}[p(y|\mathbf{x},\mathcal{D})]$ can be handled analytically:

$$\begin{aligned}\mathrm{H}[p(y|\mathbf{x},\mathcal{D})] &\overset{1}{\approx} \mathrm{h}\left(\int \Phi(f_{\mathbf{x}})\mathcal{N}(f_{\mathbf{x}}|\mu_{\mathbf{x}},\sigma_{\mathbf{x}}^2)df_{\mathbf{x}}\right)\\ &= \mathrm{h}\left(\Phi\left(\frac{\mu_{\mathbf{x}}}{\sqrt{\sigma_{\mathbf{x}}^2+1}}\right)\right), \end{aligned} \tag{13}$$

where $\overset{1}{\approx}$ denotes the Gaussian approximation to the posterior of $f_{\mathbf{x}}$. The second term, $\mathbb{E}_{f\sim p(f|\mathcal{D})}\left[\mathrm{H}[p(y|\mathbf{x},f)]\right]$ can be computed approximately as follows:

$$\begin{aligned}\mathbb{E}_{f\sim p(f|\mathcal{D})}\left[\mathrm{H}[p(y|\mathbf{x},f)]\right] &\overset{1}{\approx} \int \mathrm{h}(\Phi(f_{\mathbf{x}}))\mathcal{N}(f_{\mathbf{x}}|\mu_{\mathbf{x}},\sigma_{\mathbf{x}}^2)df_{\mathbf{x}} \qquad (14)\\ &\overset{2}{\approx} \int \exp\left(-\frac{f_{\mathbf{x}}^2}{\pi\ln 2}\right)\mathcal{N}(f_{\mathbf{x}}|\mu_{\mathbf{x}},\sigma_{\mathbf{x}}^2)df_{\mathbf{x}}\\ &= \frac{C}{\sqrt{\sigma_{\mathbf{x}}^2+C^2}}\exp\left(-\frac{\mu_{\mathbf{x}}^2}{2\left(\sigma_{\mathbf{x}}^2+C^2\right)}\right),\end{aligned}$$

where $C=\sqrt{\frac{\pi\ln 2}{2}}$. The integral in the left hand side of (14) is intractable. By performing a Taylor expansion on $\ln \mathrm{h}(\Phi(f_{\mathbf{x}}))$ (see the Supplementary material) we can see that it can be approximated up to $\mathcal{O}(f_{\mathbf{x}}^4)$ by a squared exponential curve, $\exp(-f_{\mathbf{x}}^2/\pi\ln 2)$; this approximation is denoted $\overset{2}{\approx}$. Fig. 3 depicts the striking accuracy of this simple approximation. The maximum possible error in the integral in (14) that can be incurred is only 0.27%. This happens when $\mathcal{N}(f_{\mathbf{x}}|\mu_{\mathbf{x}},\sigma_{\mathbf{x}}^2)$ is centered at $\mu_{\mathbf{x}}=\pm 2.05$ with $\sigma_{\mathbf{x}}^2$ tending to zero. Now the convolution formula for Gaussians yields an closed form expression.

To summarize, the BALD algorithm for Gaussian process active binary GP classification/ preference learning consists of two steps. First it applies any standard approximate inference algorithm (such as EP) to obtain the posterior predictive mean $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ for each point of interest $\mathbf{x}$. Then, it selects the data $\mathbf{x}$ that maximizes the following objective function:

$$\mathrm{h}\left(\Phi\left(\frac{\mu_{\mathbf{x}}}{\sqrt{\sigma_{\mathbf{x}}^2+1}}\right)\right)-\frac{C}{\sqrt{\sigma_{\mathbf{x}}^2+C^2}}\exp\left(-\frac{\mu_{\mathbf{x}}^2}{2\left(\sigma_{\mathbf{x}}^2+C^2\right)}\right) \tag{15}$$

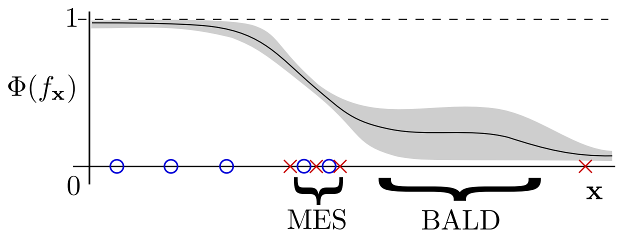

Figure 4: Toy classification example with a 1D input. Circles and crosses are the data. We plot the mean and variance of the predictive distribution. Maximum Entropy Sampling (MES, see Section 6) samples from the region of highest marginal uncertainty, ignoring the second term in (12). BALD samples from the region of greatest model uncertainty i.e. the area with the largest shaded error-area.

The BALD objective assigns high value to an $\mathbf{x}$ when the model is both uncertain in the output ($\mu_{\mathbf{x}} = 0$) and the there is high uncertainty in the parameters ($\sigma^2_{\mathbf{x}}$ is large). The second term prevents BALD from sampling in regions where the model knows that the output is uncertain. Figure 4 illustrates the differences between BALD and Maximum Entropy Sampling. MES uses only the first term in (15), and hence seeks data in an uninformative region. BALD samples data from the region of greatest uncertainty in the model.

For most practically relevant kernels, the objective (15) is a smooth and differentiable function of $\mathbf{x}$, so should we be able to select items from a continuous feature space gradient-based optimization methods can be employed. We now describe how we perform inference in our multi-user model in order to get the distributions required for making predictions and active queries.

5 Expectation Propagation and Variational Bayes

This proposed method for approximate inference in our multi-user model is based on the combination of expectation propagation Minka and Lafferty (2002); van Gerven et al. (2010) and variational inference Stern et al. (2009). We first describe the general version of the algorithm. Finally, in Section 5.4, we describe the version which employs sparse approximations to the covariance matrices $\mathbf{K}_{\text{users}}$ and $\mathbf{K}_{\text{items}}$ for speeding up computations.

The proposed EP method approximates the exact posterior distribution by the following parametric distribution:

$$\begin{aligned}\mathcal{Q}(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) \quad = \quad & \left[\prod_{u=1}^{U}\prod_{d=1}^{D}\mathcal{N}(w_{ud}|m^w_{u,d}, v^w_{u,d})\right]\left[\prod_{d=1}^{D}\prod_{i=1}^{P}\mathcal{N}(h_{d,i}|m^h_{d,i}, v^h_{d,i})\right] \\ & \left[\prod_{u=1}^{N}\prod_{j=1}^{M_u}\mathcal{N}(g_{u,z_{u,j}}|m^g_{u,j}, v^g_{u,j})\right], \qquad (16)\end{aligned}$$

where $m^w_{u,d}$, $v^w_{u,d}$, $m^h_{d,i}$, $v^h_{d,i}$, $m^g_{u,j}$, and $v^g_{u,j}$ are free parameters to be determined by EP. The joint distribution of the model parameters and the data $\mathcal{P}(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}, \mathbf{T}^{(\mathcal{D})}, \mathbf{X}, \ell)$ can be factorized into four factors $f_1, \ldots, f_4$, namely,

$$\mathcal{P}(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}, \mathbf{T}^{(\mathcal{D})}, \mathbf{X}, \ell) = \prod_{k=1}^{4} f_a(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H})\,, \tag{17}$$

where $f_1(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) = \mathcal{P}(\mathbf{T}^{(\mathcal{D})}|\mathbf{G}^{(\mathcal{D})})$, $f_2(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) = \mathcal{P}(\mathbf{G}^{(\mathcal{D})}|\mathbf{W}, \mathbf{H})$, $f_3(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) = \mathcal{P}(\mathbf{W}|\mathbf{U})$ and $f_4(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) = \mathcal{P}(\mathbf{H}|\mathbf{X}, \ell)$. EP approximates each of these exact factors by approximate factors $\hat{f}_1(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})}), \ldots, \hat{f}_4(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})$ that have the same functional form as (16), namely,

$$\begin{aligned}\hat{f}_a(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H}) \;\; = \;\; & \left[\prod_{u=1}^{U}\prod_{d=1}^{D}\mathcal{N}(w_{ud}|\hat{m}^{a,w}_{u,d}, \hat{v}^{a,w}_{u,d})\right]\left[\prod_{d=1}^{D}\prod_{i=1}^{P}\mathcal{N}(h_{d,i}|\hat{m}^{a,h}_{d,i}, \hat{v}^{a,h}_{d,i})\right] \\ & \left[\prod_{u=1}^{N}\prod_{j=1}^{M_u}\mathcal{N}(g_{u,z_{u,j}}|\hat{m}^{a,g}_{u,j}, \hat{v}^{a,g}_{u,j})\right]\hat{s}_a\,,\end{aligned} \tag{18}$$

where $a = 1, \ldots, 4$ and $\hat{m}^{a,w}_{u,d}$, $\hat{v}^{a,w}_{u,d}$, $\hat{m}^{a,h}_{d,i}$, $\hat{v}^{a,h}_{d,i}$, $\hat{m}^{a,g}_{u,j}$, $\hat{v}^{a,g}_{u,j}$ and $\hat{s}_a$ are free parameters to be determined by EP. The posterior approximation $\mathcal{Q}(\mathbf{w}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})$ is obtained as the normalized product of the approximate factors $\hat{f}_1, \ldots, \hat{f}_4$, that is,

$$\mathcal{Q}(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})}) \propto \hat{f}_1(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})}) \cdots \hat{f}_4(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})\,. \tag{19}$$

The first step of EP is to initialize all the approximate factors $\hat{f}_1, \ldots, \hat{f}_4$ and the posterior approximation $\mathcal{Q}$ to be uniform. In particular, $m^w_{u,d} = m^h_{d,i} = m^g_{u,j} = \hat{m}^{w,a}_{u,d} = \hat{m}^{a,h}_{d,i} = \hat{m}^{g,a}_{u,j} = 0$ and $v^w_{u,d} = v^h_{d,i} = v^g_{u,j} = \hat{v}^{a,w}_{u,d} = \hat{v}^{a,h}_{d,i} = \hat{v}^{a,h}_{u,j} = \infty$ for $a = 1, \ldots, 4$, $u = 1, \ldots, U$, $d = 1, \ldots, D$, $i = 1, \ldots, P$ and $j = 1, \ldots, M_u$. After that, EP refines the parameters of the approximate factors by iteratively minimizing the Kullback-Leibler (KL) divergence between $\mathcal{Q}^{\setminus a}(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})f_a(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})$ and $\mathcal{Q}^{\setminus a}(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})\hat{f}_a(\mathbf{W}, \mathbf{H}, \mathbf{G}^{(\mathcal{D})})$, for $a = 1, \ldots, 4$, where $\mathcal{Q}^{\setminus a}$ is the ratio between $\mathcal{Q}$ and $\hat{f}_a$. That is, EP iteratively minimizes

$$\mathrm{D}_{\mathrm{KL}}(Q^{\setminus a}f_a \| Q^{\setminus a}\hat{f}_a) = \int \left[Q^{\setminus a}f_a \log\frac{Q^{\setminus a}f_a}{Q^{\setminus a}\hat{f}_a} + Q^{\setminus a}\hat{f}_a - Q^{\setminus a}f_a\right] d\mathbf{W}\, d\mathbf{H}\, d\mathbf{G}^{(\mathcal{D})} \tag{20}$$

with respect to $\hat{f}_a$, for $a = 1, \ldots, 4$. The arguments to $Q^{\setminus a}f_a$ and $Q^{\setminus a}\hat{f}_a$ have been omitted in the right-hand side of (20) to improve the readability of the expression. However, the minimization of (20) does not perform well when we have to refine the parameters of $\hat{f}_2$. The reason for this is that the corresponding exact factor f_2 (equation (7) in the main document) is invariant to simultaneous changes in sign, scalings, or rotations of the entries of $\mathbf{W}$ and $\mathbf{H}$. This non-identifiability in the latent space spanned by $\mathbf{W}$ and $\mathbf{H}$ originates multiple modes in the distribution $Q^{\setminus 2}f_2$. The minimization of the direct version of the KL divergence results in an approximation that averages across all of the modes, leading to poor predictive performance. We solve this problem by following an approach similar to the one described by Stern et al. (2009). Instead of minimizing $\mathrm{KL}(\mathcal{Q}^{\setminus 2}f_2\|\mathcal{Q}^{\setminus 2}\hat{f}_2)$, we refine $\hat{f}_2$ by minimizing the reversed version of the KL divergence, that is, we minimize $\mathrm{KL}(\mathcal{Q}^{\setminus 2}\hat{f}_2\|\mathcal{Q}^{\setminus 2}f_2)$ with respect to the parameters of $\hat{f}_2$. The reversed version of the divergence has mode seeking properties (Bishop, 2007) and tends to approximate only a single mode of the target distribution, leading to better predictive accuracy.

The EP algorithm iteratively refines the approximate factors until convergence. We assume the algorithm has converged when the absolute value of the change in the parameters $m^g_{u,i}$ of $\mathcal{Q}$, where $u = 1, \ldots, U$ and $i = 1, \ldots, M_u$, is less than a threshold $\delta = 10^{-2}$ between two consecutive cycles of EP, where a cycle consists in the sequential update of all the approximate factors. However, convergence is not guaranteed and EP may end up oscillating without ever stopping (Minka, 2001). This undesirable behavior can be prevented by *damping* the EP updates (Minka and Lafferty, 2002). Let $\hat{f}^{\text{new}}_a$ denote the value of the approximate factor that minimizes the Kullback-Leibler divergence. Damping consists of using

$$\hat{f}^{\text{damp}}_a = \left[\hat{f}^{\text{new}}_a\right]^{\epsilon} \left[\hat{f}_a\right]^{(1-\epsilon)}\,, \tag{21}$$

instead of $\hat{f}^{\text{new}}_a$ for the update of each approximate factor $a = 1, \ldots, 4$. The quantity $\hat{f}_a$ represents in (21) the factor before the update. The parameter $\epsilon \in [0, 1]$ controls the amount of damping. The original EP update (that is, without damping) is recovered in the limit $\epsilon = 1$. For $\epsilon = 0$, the approximate factor $\hat{f}_a$ is not modified. To improve the converge of EP, we use a damping scheme with a parameter ϵ that is initialized to 1 and then progressively annealed as recommended by Hernández-Lobato (2010). After each iteration of EP, the value of this parameter is multiplied by a constant $k < 1$. The value selected for k is $k = 0.95$. In the experiments performed, EP performs on average about 50 iterations.

5.1 The EP predictive distribution

EP can also approximate the predictive distribution, given by equation (11) in the main manuscript. For this, we replace the exact posterior with the EP approximation $\mathcal{Q}$. In this way, we obtain

$$\mathcal{P}(t_{u,P+1}|\mathbf{T}^{(\mathcal{D})}, \mathbf{X}, \ell, p_{P+1}) \approx \Phi\left[t_{u,P+1} m^g_{u,P+1} (v^g_{u,P+1} + 1)^{-\frac{1}{2}}\right]\,, \tag{22}$$

where

$$m^g_{u,P+1} = \sum_{d=1}^{D} m^w_{u,d} m^h_{d,P+1}\,, \tag{23}$$

$$v^g_{u,P+1} = \sum_{d=1}^{D} [m^w_{u,d}]^2 v^h_{d,P+1} + \sum_{d=1}^{D} v^w_{u,d} [m^h_{d,P+1}]^2 + \sum_{d=1}^{D} v^w_{u,d} v^h_{d,P+1} \tag{24}$$

and $m^h_{d,P+1}$ and $v^h_{d,P+1}$ for $d = 1, \ldots, D$ are given by

$$m^h_{d,P+1} = \mathbf{k}_\star^{\text{T}} \left[\mathbf{K}_{\text{items}} + \text{diag}[\hat{\mathbf{v}}^{h,2}_d]\right]^{-1} \hat{\mathbf{m}}^{h,2}_d\,, \tag{25}$$

$$v^h_{d,P+1} = k_\star - \mathbf{k}_\star^{\text{T}} \left[\mathbf{K}_{\text{items}} + \text{diag}[\hat{\mathbf{v}}^{h,2}_d]\right]^{-1} \mathbf{k}_\star\,, \tag{26}$$

where $k_\star$ is the prior variance of $h_d(\mathbf{x}_{\alpha(P+1)}, \mathbf{x}_{\beta(P+1)})$, $\mathbf{k}_\star$ is a P-dimensional vector that contains the prior covariances between $h_d(\mathbf{x}_{\alpha(P+1)}, \mathbf{x}_{\beta(P+1)})$ and $h_d(\mathbf{x}_{\alpha(1)}, \mathbf{x}_{\beta(1)}), \ldots, h_d(\mathbf{x}_{\alpha(P)}, \mathbf{x}_{\beta(P)})$ for $d = 1, \ldots, D$, the function diag$(\cdot)$ applied to a vector returns a diagonal matrix with that vector in its diagonal and the vectors $\hat{\mathbf{m}}^{h,2}_d$ and $\hat{\mathbf{v}}^{h,2}_d$ are given by $\hat{\mathbf{m}}^{h,2}_d = (\hat{m}^{h,2}_{1,d}, \ldots, \hat{m}^{h,2}_{P,d})^{\text{T}}$ and $\hat{\mathbf{v}}^{h,2}_d = (\hat{v}^{h,2}_{1,d}, \ldots, \hat{v}^{h,2}_{P,d})^{\text{T}}$.

5.2 The EP update operations

In this section we describe the EP updates for refining the approximate factors $\hat{f}_1, \ldots, \hat{f}_4$. For the sake of clarity, we only include the update rules with no damping ($\epsilon = 1$). Incorporating the effect of damping in these operations is straightforward. With damping, the natural parameters of the approximate factors become a convex combination of the natural parameters before and after the update with no damping

$$[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{damp}} = \epsilon[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{old}}\,, \tag{27}$$

$$[\hat{m}^{w,a}_{u,d}]_{\text{damp}}[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{damp}} = \epsilon[\hat{m}^{w,a}_{u,d}]_{\text{new}}[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{m}^{w,a}_{u,d}]_{\text{old}}[\hat{v}^{w,a}_{u,d}]^{-1}_{\text{old}}\,, \tag{28}$$

$$[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{damp}} = \epsilon[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{old}}\,, \tag{29}$$

$$[\hat{m}^{h,a}_{d,i}]_{\text{damp}}[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{damp}} = \epsilon[\hat{m}^{h,a}_{d,i}]_{\text{new}}[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{m}^{h,a}_{d,i}]_{\text{old}}[\hat{v}^{h,a}_{d,i}]^{-1}_{\text{old}}\,, \tag{30}$$

$$[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{damp}} = \epsilon[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{old}}\,, \tag{31}$$

$$[\hat{m}^{g,a}_{d,j}]_{\text{damp}}[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{damp}} = \epsilon[\hat{m}^{g,a}_{u,j}]_{\text{new}}[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{new}} + (1-\epsilon)[\hat{m}^{g,a}_{u,j}]_{\text{old}}[\hat{v}^{g,a}_{u,j}]^{-1}_{\text{old}}\,, \tag{32}$$

where $u = 1, \ldots, U$, $d = 1, \ldots, D$, $i = 1, \ldots, P$ and $j = 1, \ldots, M_u$. The subscript *new* denotes the value of the parameter given by the full EP update operation with no damping. The subscript *damp* denotes the parameter value given by the damped update rule. The subscript *old* refers to the value of the parameter before the EP update. The updates for the parameters $\hat{s}_1, \ldots, \hat{s}_4$ are not damped. These parameters are initialized to 1 and are only updated once the EP algorithm has converged.

The factors are refined in the order $\hat{f}_4, \hat{f}_3, \hat{f}_2, \hat{f}_1$ by applying the EP and VB updates described above. Full details of the specific update operations have been relegated to the Supplementary Material.

5.3 The EP Approximation of the Model Evidence

The model evidence, or marginal likelihood is given by $\mathcal{P}(\mathbf{T}^{(\mathcal{D})}|\mathbf{X}, \ell)$. It is a useful quantity for assessing the quality of the model, and provides a formal tool for selecting hyper-parameters. Although, like the posterior, it cannot be computed analytically, once EP has converged, we can approximate it using

$$\mathcal{P}(\mathbf{T}^{(\mathcal{D})}|\mathbf{X}, \ell) \approx \int \prod_{a=1}^{4} \hat{f}_a(\mathbf{G}^{(\mathcal{D})}, \mathbf{W}, \mathbf{H})\, d\mathbf{G}^{(\mathcal{D})}\, d\mathbf{H}\, d\mathbf{W}\,. \tag{33}$$

The required computations are presented in the Supplementary Material.

Finally, some of the EP updates may generate a negative value for $\hat{v}^{g,a}_{u,i}$, $\hat{v}^{w,a}_{u,d}$ or $\hat{v}^{h,a}_{d,j}$, where $u = 1, \ldots, U$, $i = 1, \ldots, M_u$, $j = 1, \ldots, P$ and $i = 1, \ldots, 4$. Negative variances in Gaussian approximate factors are common in many EP implementations (Minka, 2001; Minka and Lafferty, 2002). When this happens, the marginals of the approximate factor with negative variances are not density functions. Instead, they are correction factors that compensate the errors in the corresponding marginals of other approximate factors. However, these negative variances can lead to failure of the proposed EP algorithm - details in the Supplementary Material. To address this problem, whenever an EP update yields a negative number for any of the $\hat{v}^{g,a}_{u,i}$, $\hat{v}^{w,a}_{u,d}$ or $\hat{v}^{h,a}_{d,j}$, we do not update this parameter, nor the corresponding $\hat{m}^{g,a}_{u,i}$, $\hat{m}^{w,a}_{u,d}$ or $\hat{m}^{h,a}_{d,j}$.

5.4 Sparse Approximations to speed up Computations

The computational cost of EP is determined by the operations needed to refine the approximate factors $\hat{f}_3$ and $\hat{f}_4$. Refinement of $\hat{f}_4$ required inversion of D $P \times P$ matrices, which has cost $\mathcal{O}(DP^3)$. Similarly, for $\hat{f}_3$ we must invert D $U \times U$ matrices, which has cost $\mathcal{O}(DU^3)$. These costs can be prohibitive when P or U are very large. Nevertheless, they can be reduced by using sparse approximations to the covariance matrices $\mathbf{K}_{\text{users}}$ and $\mathbf{K}_{\text{items}}$. We use the fully independent training conditional or FITC approximation, also known as the sparse pseudo-input GP (SPGP) Snelson and Ghahramani (2005). With FITC, the $U \times U$ covariance matrix $\mathbf{K}_{\text{users}}$ is approximated by $\mathbf{K}'_{\text{users}} = \mathbf{Q}_{\text{users}} + \text{diag}(\mathbf{K}_{\text{users}} - \mathbf{Q}_{\text{users}})$, where $\mathbf{Q}_{\text{users}} = \mathbf{K}_{\text{users},U,U_0}\mathbf{K}^{-1}_{\text{users},U_0,U_0}\mathbf{K}^{\text{T}}_{\text{users},U,U_0}$. In this expression, $\mathbf{K}_{\text{users},U_0,U_0}$ is an $U_0 \times U_0$ covariance matrix given by the evaluation of the covariance function for the users at all possible pairs of $U_0 < U$ locations or *user pseudo-inputs* $\{\mathbf{u}'_1, \ldots, \mathbf{u}'_{U_0}\}$, where $\mathbf{u}'_i \in \mathcal{U}$ for $i = 1, \ldots, U_0$, and $\mathbf{K}_{\text{users},U,U_0}$ is an $U \times U_0$ matrix with the evaluation of the covariance function for the users at all possible pairs of original user feature vectors and user pseudo-inputs, that is, $(\mathbf{u}_i, \mathbf{u}'_j)$, for $i = 1, \ldots, U$ and $j = 1, \ldots, U_0$. Similarly, the $P \times P$ covariance matrix $\mathbf{K}_{\text{items}}$ is also approximated by $\mathbf{K}'_{\text{items}} = \mathbf{Q}_{\text{items}} + \text{diag}(\mathbf{K}_{\text{items}} - \mathbf{Q}_{\text{items}})$, where $\mathbf{Q}_{\text{items}} = \mathbf{K}_{\text{items},P,P_0}\mathbf{K}^{-1}_{\text{items},P_0,P_0}\mathbf{K}^{\text{T}}_{\text{items},P,P_0}$, $\mathbf{K}_{\text{items},P_0,P_0}$ is a $P_0 \times P_0$ covariance matrix given by the evaluation of the preference kernel at all possible pairs of $P_0 < P$ locations or *item-pair pseudo-inputs* $\{(\mathbf{x}'_1, \mathbf{x}''_1), \ldots, (\mathbf{x}'_{P_0}, \mathbf{x}''_{P_0})\}$, where $\mathbf{x}'_i, \mathbf{x}''_i \in \mathcal{X}$ for $i = 1, \ldots, P_0$, and $\mathbf{K}_{\text{items},P,P_0}$ is a $P \times P_0$ matrix with the evaluation of the preference kernel at all possible combinations of feature vectors for the original item pairs and item-pair pseudo-inputs, that is, $((\mathbf{x}_{\alpha(i)}, \mathbf{x}_{\beta(i)}), (\mathbf{x}'_j, \mathbf{x}''_j))$, for $i = 1, \ldots, P$ and $j = 1, \ldots, P_0$.

Detailed description of how to refine the third and fourth approximate factors when $\mathbf{K}_{\text{users}}$ and $\mathbf{K}_{\text{items}}$ are replaced by $\mathbf{K}'_{\text{users}}$ and $\mathbf{K}'_{\text{items}}$, respectively may be found in the Supplementary material. The required operations are can be efficiently implemented using the formulas described in (Naish-Guzman and Holden, 2007) and (Gredilla, 2010). The resulting complexity of inference, after applying the FITC approximation to update $\hat{f}_3$ and $\hat{f}_4$ is $\mathcal{O}(DU_0^2U)$ and $\mathcal{O}(DP_0^2P)$ operations, respectively.

We now have a flexible multi-user model in which we can perform inference in reasonable computational time. In the following section we will compare our model to two current approached to probabilistic multi-user preference learning, and compare BALD to other popular algorithms for active learning.

6 Related Methods

6.1 Multi-User Preference Learning

Model of Birlutiu et al. As in the single-task preference learning model, a different GP classifier is fitted to the data generated by each user. However, the different classifiers are now connected by a common GP prior for the latent preference functions which is optimized to fit the data Birlutiu et al. (2010). Let g_u be the u-th user's latent preference function and let $\boldsymbol{g}_u$ be the k-dimensional vector with the evaluation of this function at all the observed pairs of items, that is, $k = P$. Let

$\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$ denote the prior mean and prior covariance matrix of $\boldsymbol{g}_u$. Then $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$ are iteratively refined by an EM algorithm which iterates the following steps:

E-step Estimate the sufficient statistics (mean $\boldsymbol{\mu}_u$ and covariance matrix $\boldsymbol{\Sigma}_u$) of the posterior distribution of $\boldsymbol{g}_u$ for user $u = 1, \dots, U$, given the current estimates at step t of the parameters $\bar{\boldsymbol{\mu}}^{(t)}$ and $\bar{\boldsymbol{\Sigma}}^{(t)}$ of the common GP prior.

M-step Re-estimate the parameters of the GP prior using

$$\bar{\boldsymbol{\mu}}^{(t+1)} = \frac{1}{U}\sum_{u=1}^{U}\boldsymbol{\mu}_u \,,$$

$$\bar{\boldsymbol{\Sigma}}^{(t+1)} = \frac{1}{U}\sum_{u=1}^{U}(\bar{\boldsymbol{\mu}}^{(t)} - \boldsymbol{\mu}_u)^{\mathrm{T}}(\bar{\boldsymbol{\mu}}^{(t)} - \boldsymbol{\mu}_u) + \frac{1}{U}\sum_{u=1}^{U}\boldsymbol{\Sigma}_u \,.$$

On the first iteration of the EM algorithm we fix $\bar{\boldsymbol{\mu}}^{(0)} = \mathbf{0}$ and compute $\bar{\boldsymbol{\Sigma}}^{(0)}$ by evaluating a preference covariance function at all the possible pairs of items. This preference covariance function is generated by a squared exponential kernel with unit lengthscale. The computational cost of the EM algorithm is rather high since each iteration requires the inversion of U covariance matrices of dimension $P \times P$, where P is the total number of observed item pairs, is $\mathcal{O}(UP^3)$. As described in Section 5.4, the cost of our equivalent algorithm (i.e. where we do not incorporate user features) is $\mathcal{O}(DP^3)$, which, even before including the FITC approximation is significantly cheaper because $D \ll U$. To reduce the computational burden, we limit the number of iterations of the EM algorithm to 20. In our experiments, increasing the number of EM iterations above 20 did not lead to improvements in the predictive performance of this method.

Birlutiu's approach has an advantage that it learns the full covariance matrix over the P item pairs for each user, this makes the model flexible, but causes the poor scalability with the number of users. We note that the original implementation in (Birlutiu et al., 2010) does not use the preference kernel, and used a sampling-based implementation. We expect that our implementation of this model that uses the preference kernel propagation, which is known to perform well with Gaussian processes (Nickisch and Rasmussen, 2008), will augment its performance, but it provides the fairest comparison of the underlying model.

Model of Bonilla et al. An alternative multi-user model for learning pairwise user preferences is described in Bonilla et al. (2010). This approach is based on the assumption that users with similar characteristics or feature vectors should have similar preferences. In particular, there is a single large latent function g which depends on both the features of the two items to be compared, $\mathbf{x}_i$ and $\mathbf{x}_j$, and the specific feature vector for the user who makes the comparison, $\mathbf{u}_u$. Within the framework of the preference kernel, the likelihood function for Bonilla's model is

$$\mathcal{P}(y|\mathbf{u}_u, \mathbf{x}_i, \mathbf{x}_j, g) = \Phi(yg(\mathbf{x}_i, \mathbf{x}_j, \mathbf{u}_u) \tag{34}$$

and the prior for the utility function g is a Gaussian process with zero mean and covariance function

$$k_{\text{Bonilla}}((\mathbf{u}_u, \mathbf{x}_i, \mathbf{x}_j), (\mathbf{u}_s, \mathbf{x}_k, \mathbf{x}_l),) = k_{\text{users}}(\mathbf{u}_u, \mathbf{u}_s) k_{\text{pref}}((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}_k, \mathbf{x}_l)) \,, \tag{35}$$

where k_{pref} is a preference kernel and k_{users} is a covariance function for user features. This latter function will be large when $\mathbf{u}_u$ and $\mathbf{u}_s$ are similar to each other and small otherwise. Therefore, the effect of k_{users} in (35) is to encourage users with similar feature vectors to agree on their preferences. The preference kernel allows us to do efficient approximate inference in Bonilla's model using any standard implementation of expectation propagation for the binary classification problem with GPs. However, the computational cost of Bonilla's method is rather high. When the preference kernel is used, the cost of this technique is $\mathcal{O}((\sum_{u=1}^{U} M_u)^3)$, where U is the total number of users and M_u is the number of pairs evaluated by the u-th user. By contrast, our approach, when including user features also has complexity $\mathcal{O}(DU^3 + DP^3)$. In the original work Bonilla et al. (2010) they do not use the preference kernel, this reduces their complexity to $O((\sum_{u=1}^{U} I_u)^3)$, where I_u denotes the number of unique items compared by the u-th user. However, this makes inference more complex and they are constrained to using the Laplace approximation, which is shown to perform less well than EP for with Gaussian Process models (Nickisch and Rasmussen, 2008). Furthermore, this cost is still typically much greater than for our model. In practice, Bonilla's method is infeasible when we have observations for more than a few hundred users. Additionally, this method requires that feature vectors are available for the different users and that users with similar feature vectors generate similar preference observations. When these conditions do not hold, Bonilla's method may lead to poor predictive performance.

6.2 Active Learning

Maximum Entropy Sampling (MES) (Sebastiani and Wynn, 2000) is similar to BALD in the sense that it also works explicitly in data space (that is, using equation (12)). MES was proposed for regression models with input-independent observation noise. In this scenario, the noise in the target variable y does not depend on the input $\mathbf{x}$ and the second term in equation (12) is constant and can be safely ignored. However, if noise in the target variable is not input-independent, MES will tend to sample regions of the input space where uncertainty in g is low but uncertainty in the labels (because of observation noise) is high, as illustrated in Figure 4.

Query by Committee (QBC) makes a different approximation to (12) (Freund et al., 1997). QBC samples parameters from the posterior (called committee members). These parameters are then used to perform a deterministic vote on the outcome of each candidate $\mathbf{x}$. The $\mathbf{x}$ with the most balanced vote is selected for the next active inclusion in the training set. This objective is termed the 'principle of maximal disagreement'. QBC is similar to BALD when the objective used by BALD is approximated by sampling from the posterior, with the exception that BALD uses a probabilistic measure of disagreement (equation (12)). Note that the deterministic vote criterion used by QBC does not take into account the confidence of the learning method on its predictions. Because of this, QBC can exhibit the same pathologies as MES.

The Informative Vector Machine (IVM) (Lawrence et al., 2002) is also motivated by information theory. This method was originally designed for sub-sampling a dataset and not for addressing online active learning problems. The IVM requires that the target variables y are observed prior to making a query and it is therefore not applicable online active learning tasks. Nonetheless, BALD can be applied to the dataset sub-sampling problem for which the IVM is designed, it is simply equipped with less information. The IVM works with equation (11) instead of (12). Entropies for the latent function g are calculated approximately in the marginal subspace corresponding to the observed data points. For this, the IVM employs a Gaussian approximation to the posterior distribution at these locations. The posterior approximation must be updated to evaluate the entropy

Dataset	BALD	Random	Entropy	QBC_2	QBC_{100}	IVM	SVM
austra	**18.54 ± 2.94**	44.15 ± 12.63	22.46 ± 6.20	68.38 ± 1.38	29.31 ± 5.06	28.46 ± 6.58	55.00 ± 1.00
cancer	**16.80 ± 0.59**	22.20 ± 1.25	21.10 ± 0.48	39.65 ± 0.41	18.95 ± 1.34	21.35 ± 0.50	24.40 ± 8.30
crabs	9.80 ± 0.58	11.40 ± 1.29	**9.20 ± 0.49**	17.00 ± 1.26	10.20 ± 0.97	13.60 ± 1.86	23.20 ± 7.29
letter D v. P	**45.30 ± 1.14**	92.10 ± 2.41	51.50 ± 0.83	48.80 ± 1.34	49.10 ± 1.38	51.00 ± 0.84	N/A
letter E v. F	**30.17 ± 1.11**	71.50 ± 17.72	34.33 ± 0.42	44.67 ± 2.12	30.67 ± 1.65	33.00 ± 2.27	N/A
vehicle	**33.20 ± 2.11**	75.30 ± 7.38	36.60 ± 1.74	85.20 ± 7.16	35.00 ± 1.80	38.20 ± 2.00	41.60 ± 1.64
wine	**8.80 ± 0.37**	26.60 ± 8.57	10.80 ± 1.66	36.40 ± 8.36	12.60 ± 1.78	20.40 ± 9.92	23.80 ± 3.48
wdbc	**18.15 ± 0.37**	47.00 ± 1.46	22.55 ± 1.05	43.85 ± 1.39	23.40 ± 1.05	21.40 ± 0.85	45.70 ± 1.75

Table 1: Performance of BALD and other active learning algorithms on several binary classification datasets from the UCI repository. Entries indicate the number of datapoints (plus or minus one standard error of the mean) required to achieve 95% of the predictive performance achieved by including the entire pool set. Bold typeface indicates the best performing algorithm for each dataset. N/A indicates that the corresponding algorithms did not meet the 95% performance level by the end of the simulation.

decrease after the inclusion of each candidate data point. If there are n candidate inputs under consideration, a total of $\mathcal{O}(n)$ posterior updates are required. By contrast, BALD only requires $\mathcal{O}(1)$ updates. In practice, the IVM approach is infeasible in sophisticated models such as the proposed multi-user approach.

Finally, Tong and Koller (2001) propose an algorithm for active learning with support vector machines. This method approximates the version space (the set of hyper-planes consistent with the data) with a simpler object, such as a hyper-sphere. The algorithm selects the data point whose dual plane is closest to bisecting this hyper-sphere.

6.3 Binary Classification Investigation

We describe an experiment used to to compare BALD to these approaches in the context of binary classification with GPs. The datasets were divided randomly into pool and test sets. Each algorithm was initialized with two data points, one from each class, drawn randomly from the pool. The algorithms select points sequentially, and their classification error was assessed on the test set after each query. The procedure was repeated for several random splits of the data to assess statistical significance. Figure 1 provides a summary of the results. BALD can be seen to outperform consistently the alternative algorithms across many datasets. The closest competitor is Maximum Entropy Sampling, which we use as a benchmark active learning algorithm for use with the multi-user preference model in the main paper.

Table 1 depicts an investigation of BALD's power against the aforementioned algorithms and a non-probabilistic SVM-based active learning algorithm of (Tong and Koller, 2001); this algorithm seeks to reduce the size of 'Version Space' maximally fast, which is equivalent to the information theoretic approach but for non-probabilistic models.

We find that BALD is consistently the best performing algorithm, the closest method is MES, whose performance approaches BALD when the data has very low noise. We therefore use MES as our benchmark active learning algorithm for our experiments.

7 Experiments

7.1 Datasets

The performance of our collaborative preference model with the BALD active learning strategy is evaluated in a series of experiments with simulated and real-world data. The datasets were pre-processed as follows.

Synthetic. We generated 10 items with feature vectors $\mathbf{x}_i = (x_{i1}, x_{i2})$, where x_{i1} and x_{i2} are uniformly distributed with zero mean and unit variance, for $i = 1, \ldots, 10$. The user preferences are obtained using $D = 5$ latent functions $h_1, \ldots, h_5$ sampled from a Gaussian process with zero mean and preference kernel given by a squared exponential kernel with unit length-scale. The preferences for the u-th user are generated according to the sign of $g_u(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{5} w'_d(\mathbf{u}_u) h_d(\mathbf{x}_i, \mathbf{x}_j) + \epsilon_{ij}$, where $\epsilon_{ij} \sim \mathcal{N}(0, 1)$, the user features $\mathbf{u}_u$ are generated in the same manner as the feature vectors for the items and the functions $w'_1, \ldots, w_D$ follow the same prior distribution as $h_1, \ldots, h'_5$.

Jura. This dataset contains concentration measurements for 7 heavy metals in soils of the Swiss Jura region at 359 locations (Atteia et al., 1994; Webstet et al., 1994). We standardized the measurements of each heavy metal to have zero mean and unit standard deviation across the whole dataset. The standardized measurements are used as utility values to generate preferences for any pair of heavy metals at each location. Therefore, in this dataset, the locations correspond to users and each heavy metal represents a different item. To generate the item features, we randomly singled out 20 locations. The item features are given by the standardized measurements obtained at these locations. The user features correspond to the x and y coordinates for the measurements as well as the rock and land type.

MovieLens. This dataset contains 1 million ratings from 6,000 users on 4,000 movies. A total of 10 movies were randomly selected from the 50 movies with most ratings. We also selected those users with at least 7 ratings on these 10 movies. The remaining missing ratings were estimated using a nearest neighbor method. The ratings for each user were used as utility values in order to generate preferences for each pair of movies. The features for each user are gender, age and occupation. The features for each movie are genres such as *action*, *comedy* or *adventure*.

Sushi. This dataset contains complete rankings given by 5,000 users on 10 different types of sushi (Kamishima et al., 2005), where each sushi includes as features style, major group, minor group, heaviness, consumption frequency, normalized price and sale frequency. The different users are also represented by a set of features which include gender, age and geographical/regional information.

Election. This dataset contains the votes obtained by 8 political parties (items) at 650 constituencies (users) in the 2010 general elections in the UK. We only kept data for those constituencies with at least votes for more than 6 parties. Missing votes were estimated using a nearest neighbor method. To generate feature vectors for each item, we randomly singled out 20 constituencies and used the corresponding votes as features. The features for each 'user' are the corresponding coordinates (latitude and longitude) of the centroid of the constituency on the map.

7.2 Tuning the kernel lengthscale

Before the main experiments we perform an initial investigation to show that approximation of the model evidence given by EP can be used to tune the kernel hyper-parameters in the proposed multi-user model. For this, we use the synthetic dataset described in the main document. Figure 5 shows a contour plot of the log-evidence returned by EP when run on the first training set of the experiments

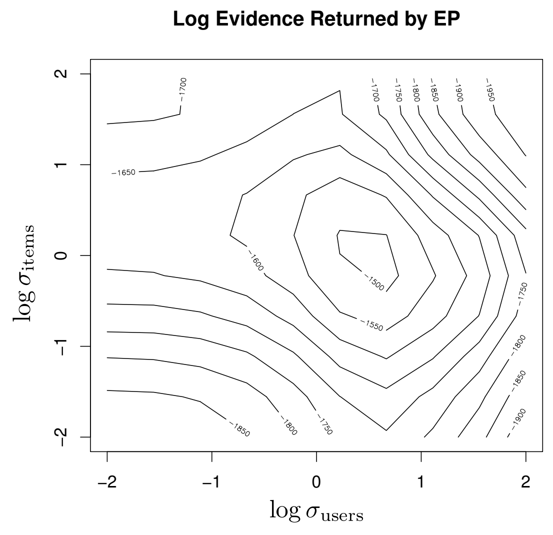

Figure 5: Logarithm of the evidence returned by EP when run on the first training set of the experiments with synthetic data. Different values are considered for the lengthscale parameters σ_{users} and σ_{items}. The synthetic data are generated using $\log \sigma_{\text{users}} = 0$ and $\log \sigma_{\text{items}} = 0$. The highest evidence returned by EP corresponds to values of $\log \sigma_{\text{users}}$ and $\log \sigma_{\text{items}}$ close to zero.

with synthetic data and 100 users. Different values are considered for the lengthscale parameters σ_{users} and σ_{items}. The synthetic data are generated using $\log \sigma_{\text{users}} = 0$ and $\log \sigma_{\text{items}} = 0$. The highest evidence returned by EP corresponds to values of $\log \sigma_{\text{users}}$ and $\log \sigma_{\text{items}}$ close to zero. In this experiment we are running EP using a total of 20 latent functions, while the data are generated using only 5 latent functions. As mentioned in the main document, the proposed multi-user model seems to be robust to over-fitting and over-estimation of the number of latent functions does not seem to harm predictive performance.

7.3 Comparison with other multi-user methods

Alternative models. Two versions of the proposed collaborative preference (CP) model are used. The first version (CPU) takes into account the available user features, as described in Section 3. The second version (CP) ignores these features by replacing $\mathbf{K}_{\text{users}}$ in (6) with the identity matrix. We compare to the two multi-user methods described in Section 6. We denote to the model of Birlutiu *et al.* as (BI), and that of Bonilla *et al.* as (BO). We implement BO and BI using the preference kernel and EP for approximate inference. Remember, that BI does not incorporate user features, and BO must include them. Finally, we consider a single user approach (SU) which fits a different GP classifier independently to the data of each user.

Experimental procedure. Due to the high computational cost of BI and BO, to compare to these methods we must subsample the datasets, keeping only 100 users. The available data were split randomly into training and test sets of item pairs, where the training sets contain 20 pairs per user in Sushi, MovieLens and Election, 15 pairs in Jura and 30 in Synthetic. This was repeated

Table 2: Average test error with 100 users.

Dataset	CPU	CP	BI	BO	SU
Synthetic	0.162	0.180	0.175	**0.157**	0.226
Sushi	0.171	0.163	**0.160**	0.266	0.187
MovieLens	0.182	**0.166**	0.168	0.302	0.217
Election	0.199	0.123	**0.077**	0.401	0.300
Jura	0.159	**0.153**	**0.153**	0.254	0.181

Table 3: Training times (s) with 100 users.

Dataset	CPU	CP	BI	BO	SU
Synthetic	7.793	9.498	22.524	311.574	0.927
Sushi	5.694	4.307	20.028	215.136	0.817
MovieLens	5.313	4.013	19.366	69.048	0.604
Election	13.134	12.408	20.880	120.011	0.888
Jura	3.762	2.404	15.234	88.502	0.628

25 times to obtain statistically meaningful results. In CPU and CP, we selected the number of latent functions D to be 20 (see Table 7.3). In general, the proposed models, CPU and CP, are robust to over-fitting and over-estimation of D does not harm predictive performance. Note that the Synthetic dataset is generated using $D = 5$ and CPU and CP still obtain very good results using $D = 20$. This automatic pruning of unnecessary degrees of freedom seems to be common in methods based on variational Bayes MacKay (2001). We selected the kernel lengthscales to be equal to the median distance between feature vectors. This leads to good empirical performance for most methods. An exception is BO, where the kernel hyperparameters are tuned to some held-out data using automatic relevance determination. In our model, we can also estimate the kernel lengthscales by maximizing the EP approximation of the model evidence, as illustrated in the Supplementary material. This alternative approach can be used when it is necessary to fine tune the lengthscale parameters to the data. In CPU we use $U_0 = 25$ pseudo inputs for approximating $\mathbf{K}_{\text{users}}$. These pseudo inputs are selected randomly from the set of available data points. Similarly, in CP and CPU, we use $P_0 = 25$ pseudo inputs for approximating $\mathbf{K}_{\text{items}}$, except in the Jura and Election datasets (which contain fewer items) where we use $P_0 = 15$. The results obtained are not sensitive to the number of pseudo inputs used, as long as the number is not excessively low.

Results. Average test errors are shown in Table 2. Those highlighted in bold are statistically different to those not highlighted (calculated using a paired t test). Overall, CP and CPU outperform SU and BO, and breaks even with BI; the final result is notable as BI learns the full mean and covariance structure across all users, ours uses only a few latent dimensions, which provides the key to scaling to many more users. CP outperforms CPU in all cases except in the Synthetic dataset. In the real-world datasets, users with similar features do not seem to have similar preferences and so correlating behavior of users with similar features is detrimental. In this case, the unsupervised learning of similarities in user preferences is more useful for prediction than the user features. This also explains the poor overall results obtained by BO. Finally, running times in seconds are presented in Table 3. The entries for BO do not include the time spent by this method to tune the kernel hyper-parameters. CP and CPU are faster than BO and BI. The FITC approximation imposes a large multiplicative constant in the cost of CP and CPU so for larger datasets the gains are much larger.

7.4 Active learning on large datasets

Here we evaluate the performance of BALD, in particular, we compare CPU, CP, and SU using BALD (-B), Maximum Entropy Sampling (-E) and random sampling (-R). We now use all the available users from each dataset, with a maximum of 1000 users. For each user the available preference data are split randomly into training, pool and test sets with 5, 35 and 5 data points

Table 4: Test error for each method and active learning strategy with at most 1000 users. Standard deviations ommitted for readability.

Dataset	**CPU-B**	**CPU-E**	**CPU-R**	**CP-B**	**CP-E**	**CP-R**	**SU-B**	**SU-E**	**SU-R**
Synthetic	**0.136**	**0.136**	0.14	**0.154**	0.161	0.179	**0.253**	0.263	0.276
Sushi	**0.150**	0.155	0.185	**0.143**	0.151	0.179	**0.183**	0.199	0.215
MovieLens	**0.172**	0.177	0.200	**0.165**	0.171	0.196	**0.233**	0.24	0.253
Election	0.221	**0.188**	0.265	**0.107**	**0.103**	0.184	**0.331**	0.349	0.344
Jura	**0.143**	**0.144**	0.17	**0.138**	**0.138**	0.17	0.183	**0.169**	0.201

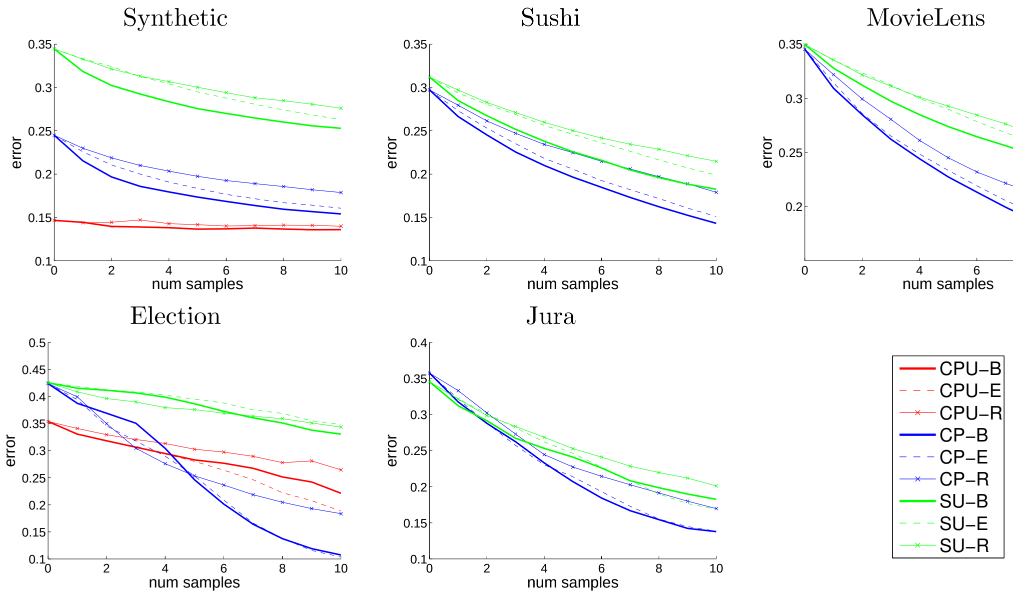

Figure 6: Average test error for CPU, CP and SU, using the strategies BALD (-B), entropy (-E) and random (-R) for active learning. For visual clarity, the curves for CPU are included only in the Synthetic and Election datasets.

respectively in Synthetic, Sushi and MovieLens, 3, 22 and 3 data points in Election and 3, 15 and 3 data points in Jura. Each method is fitted using the training sets and its performance is then evaluated on the corresponding test sets. After this, the most informative data point is identified in each of the pool sets. These data points are moved into the corresponding training sets and the process repeats until 10 of these active additions to the training sets have been completed. The entire process, including the dataset splitting is repeated 25 times. Figure 6 shows the learning curve for each method. For clarity, the curve for CPU is included only for the Synthetic and Election datasets; in the other datasets CPU is marginally outperformed by CP. Average errors after 10 queries from the pool set of each user are summarized in Table 4. For each model (CPU, CP and SU), the results of the best active learning strategy are highlighted in bold. The results of the best model/active learning strategy combination are underlined. Highlighted results are statistically significant with respect to non-highlighted results according to a paired t test. BALD always outperforms random sampling and usually outperforms or obtains equivalent performance to MES. In particular, BALD significantly outperforms MES in nine cases, while MES is better than BALD in only two cases.

8 Conclusions

We have proposed a multi-user model that combines collaborative filtering methods with GP binary preference modeling. Our approach extends probabilistic multi-user preference learning to allow both behavioral similarities between users and their feature vectors to be exploited to make predictions. The proposed approach advances current work in that it can handle both the case when user features are present, and when they are not useful or non-existent. The model takes advantage of a new reformulation of preference learning as a particular case of binary classification with GPs, when a covariance function called the preference kernel is used. This reformulation significantly reduces the complexity of inference, allowing us to perform efficient inference. Importantly the complexity of our approach scales sub-linearly in the number of users, allowing us to apply the model to larger datasets than the current state-of-the-art. We have also presented BALD, a new active learning strategy for binary classification models with GPs. The proposed multi-user model with BALD performs as well as, or better than single-user methods and existing approaches for multi-user preference learning on simulated and real-world data against single-user methods, whilst having significantly lower computational times.

Although we have made advances in scalability for this class of probabilistic models, one direction of future work is making this approach 'web-scalable'. One potential application is for search click-through data, which can be interpreted as users making implicit preference judgements between the returned items (Joachims et al., 2005). Applying this model to such a domain would require very few pseudo-inputs in the FITC approximation, and hence they should be chosen wisely, rather than randomly. An active sampling approach, such as BALD would need to be developed for this purpose.

Furthermore, our model is designed for the scenario when we have a full user feature vector, and none at all. However, if we are missing just part of the feature vector, we simply replace the missing elements with the empirical mean taken from the other users. The problem of performing Gaussian process inference with partial inputs is still an open area of research because even if we have a distribution that captures our uncertainty over the missing element, integrating this distribution against the GP likelihood is highly intractable. One approach for example is to integrate only the kernel against a prior over the missing element (Girard et al., 2003) which yields a new length scale. However, similar to simply using an empirical mean to replace the element this does not smoothly reduce to the case when we have no user features. Finding a solution to the missing input problem that satisfies this is an interesting future direction.

And a final paragraph wrapping everything up?

References

Atteia, O., Dubois, J.-P., and Webster, R. (1994). Geostatistical analysis of soil contamination in the swiss jura. *Environmental Pollution*, 86(3):315 – 327.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 21–30.

Bernardo, J. (1979). Expected information as expected utility. *The Annals of Statistics*, 7(3):686–690.

Birlutiu, A., Groot, P., and Heskes, T. (2010). Multi-task preference learning with an application to hearing aid personalization. *Neurocomputing*, 73(79):1177 – 1185.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

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

Brochu, E., de Freitas, N., and Ghosh, A. (2007). Active preference learning with discrete choice data. *Advances in Neural Information Processing Systems 20*, 20:409–416.

Chu, W. and Ghahramani, Z. (2005). Preference learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144.

Cover, T., Thomas, J., and Wiley, J. (1991). *Elements of information theory*, volume 6. Wiley Online Library.

Dasgupta, S. (2005). Analysis of a greedy active learning strategy. In *Advances in neural information processing systems 17*, pages 337–344.

De Gemmis, M., Iaquinta, L., Lops, P., Musto, C., Narducci, F., and Semeraro, G. (2009). Preference learning in recommender systems. In *ECML/PKDD-09 Workshop on Preference Learning*.

Freund, Y., Seung, H., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168.

Fürnkranz, J. and Hüllermeier, E. (2010). *Preference learning*. Springer-Verlag New York Inc.

Girard, A., Rasmussen, C., Quinonero-Candela, J., and Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs. application to multiple-step ahead time series forecasting.

Golovin, D. and Krause, A. (2010). Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Proceedings of International Conference on Learning Theory (COLT)*.

Gredilla, M. L. (2010). *Sparse Gaussian Processes for Large-scale Machine Learning*. PhD thesis, Universidad Carlos III de Madrid.

Heckerman, D., Breese, J., and Rommelse, K. (1995). Troubleshooting under uncertainty. *Communications of the ACM*, 38(3):27–41.

Hernández-Lobato, J. M. (2010). *Balancing Flexibility and Robustness in Machine Learning: Semi-parametric Methods and Sparse Linear Models*. PhD thesis, Universidad Autónoma de Madrid.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM.

Kamishima, T., Kazawa, H., and Akaho, S. (2005). Supervised ordering - an empirical survey. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 673–676.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434.

Krishnapuram, B., Williams, D., Xue, Y., Hartemink, A., Carin, L., and Figueiredo, M. (2004). On semi-supervised classification. In *Advances in neural information processing systems 17*, pages 721–728.

Lawrence, N., Seeger, M., and Herbrich, R. (2002). Fast sparse gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems 15*, 15:609–616.

Lindley, D. (1956). On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005.

MacKay, D. (1992). Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.

MacKay, D. J. C. (2001). Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Available at http://www.inference.phy.cam.ac.uk/mackay/minima.pdf.

Minka, T. (2001). *A family of algorithms for approximate Bayesian inference.* PhD thesis, MIT.

Minka, T. and Lafferty, J. (2002). Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359.

Naish-Guzman, A. and Holden, S. B. (2007). The generalized fitc approximation. In *Advances in Neural Information Processing Systems 20.*

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *The Journal of Machine Learning Research*, 9:2035–2078.

Panzeri S., Senatore R., M. M. and R.S., P. (2007). Correcting for the sampling bias problem in spike train information measures. *Journal of neurophysiology*, 98(3):1064.

Raiko, T., Ilin, A., and Juha, K. (2007). Principal component analysis for large scale problems with lots of missing values. In Kok, J., Koronacki, J., Mantaras, R., Matwin, S., Mladenic, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 691–698. Springer Berlin / Heidelberg.

Sebastiani, P. and Wynn, H. (2000). Maximum entropy sampling and optimal Bayesian experimental design. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(1):145–157.

Snelson, E. and Ghahramani, Z. (2005). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18.*

Stern, D. H., Herbrich, R., and Graepel, T. (2009). Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pages 111–120.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

van Gerven, M., Cseke, B., de Lange, F., and Heskes, T. (2010). Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *Neuroimage*, 50:150–161.

Webstet, R., Atteia, O., and Dubois, J.-P. (1994). Coregionalization of trace metals in the soil in the swiss jura. *European Journal of Soil Science*, 45(2):205–218.