Machine Learning
https://doi.org/10.1007/s10994-019-05867-2

Scalable Bayesian preference learning for crowds

Edwin Simpson[1] · **Iryna Gurevych**[1]

Received: 21 January 2019 / Revised: 11 December 2019 / Accepted: 18 December 2019

Abstract
We propose a scalable Bayesian preference learning method for jointly predicting the preferences of individuals as well as the consensus of a crowd from pairwise labels. Peoples' opinions often differ greatly, making it difficult to predict their preferences from small amounts of personal data. Individual biases also make it harder to infer the consensus of a crowd when there are few labels per item. We address these challenges by combining matrix factorisation with Gaussian processes, using a Bayesian approach to account for uncertainty arising from noisy and sparse data. Our method exploits input features, such as text embeddings and user metadata, to predict preferences for new items and users that are not in the training set. As previous solutions based on Gaussian processes do not scale to large numbers of users, items or pairwise labels, we propose a stochastic variational inference approach that limits computational and memory costs. Our experiments on a recommendation task show that our method is competitive with previous approaches despite our scalable inference approximation. We demonstrate the method's scalability on a natural language processing task with thousands of users and items, and show improvements over the state of the art on this task. We make our software publicly available for future work (https://github.com/UKPLab/tacl2018-preference-convincing/tree/crowdGPPL).

1 Introduction

Preference learning involves comparing a set of alternatives according to a particular quality (Fürnkranz and Hüllermeier 2010), which often leads to a divergence of opinion between people. For example, in argument mining, a sub-field of natural language processing (NLP), one goal is to rank arguments by their *convincingness* (Habernal and Gurevych 2016). Whether a particular argument is convincing or not depends on the reader's point of view and prior knowledge (Lukin et al. 2017). Similarly, personal preferences affect recommender systems, which often perform better if they tailor

Editors: Karsten Borgwardt, Po-Ling Loh, Evimaria Terzi, and Antti Ukkonen.

✉ Edwin Simpson
simpson@ukp.informatik.tu-darmstadt.de

Iryna Gurevych
gurevych@ukp.informatik.tu-darmstadt.de

[1] Ubiquitous Knowledge Processing Lab, Department of Computer Science, Technische Universität Darmstadt, Darmstadt, Germany

recommendations to a specific user (Resnick and Varian 1997). Disagreements also occur when preference annotations are acquired from multiple annotators, for example, using crowdsourcing, and are often mitigated by redundant labelling (Snow et al. 2008; Banerji et al. 2010). Therefore, we require preference learning methods that can account for differences of opinion to (1) predict *personal* preferences for members of a crowd and (2) infer a *consensus* given observations from multiple users. For both tasks, our goal is to rank items or choose the preferred item from any given pair.

Recommender systems often predict a user's preferences via *collaborative filtering*, which overcomes data sparsity by exploiting similarities between the preferences of different users (Resnick and Varian 1997; Koren et al. 2009). Many recommender systems are based on *matrix factorisation* techniques that are trained using observations of numerical ratings. However, different annotators often disagree over numerical annotations and can label inconsistently over time (Ovadia 2004; Yannakakis and Hallam 2011), as annotators may interpret the values differently: a score of 4/5, say, from one annotator may be equivalent to 3/5 from another. The problem is avoided by *pairwise labelling*, in which the annotator selects their preferred item from a pair, which can be quicker (Kendall 1948; Kingsley and Brown 2010; Yang and Chen 2011), more accurate (Kiritchenko and Mohammad 2017), and facilitates the total sorting of items, as it avoids two items having the same value.

Pairwise labels provided by a crowd or extracted from user logs (Joachims 2002) are often noisy and sparse, i.e., many items or users have few or no labels. This motivates a Bayesian treatment, which has been shown to benefit matrix factorisation (Salakhutdinov and Mnih 2008) and preference learning (Chen et al. 2013). Some previous Bayesian methods for preference learning use *Gaussian processes (GPs)* to account for *input features* of items or users (Chu and Ghahramani 2005; Houlsby et al. 2012; Khan et al. 2014). These are features that can be extracted from content or metadata, such as *embeddings* (Mikolov et al. 2013; Devlin et al. 2019), which are commonly used by NLP methods to represent words or documents using a numerical vector. Input features allow the model to extrapolate to new items or users and mitigate labelling errors (Felt et al. 2016). However, previous Bayesian preference learning methods that account for input features using GPs do not scale to large numbers of items, users, or pairwise labels, as their computational and memory requirements grow with the size of the dataset.

In this paper, we propose a scalable Bayesian approach to pairwise preference learning with large numbers of users or annotators. Our method, *crowdGPPL*, jointly models personal preferences and the consensus of a crowd through a combination of matrix factorisation and Gaussian processes. We propose a *stochastic variational inference (SVI)* scheme (Hoffman et al. 2013) that scales to extremely large datasets, as its memory complexity and the time complexity of each iteration are fixed independently of the size of the dataset. Our new approach opens the door to novel applications involving very large numbers of users, items and pairwise labels, that would previously have exceeded computational or memory resources and were difficult to parallelise. We evaluate the method empirically on two real-world datasets to demonstrate the scalability of our approach, and its ability to predict both personal preferences and a consensus given preferences from thousands of users. Our results improve performance over the previous state-of-the-art (Simpson and Gurevych 2018) on a crowdsourced argumentation dataset, and show that modelling personal preferences improves predictions of the consensus, and vice versa.

2 Related work

To obtain a ranking from pairwise labels, many preference learning methods model the user's choices as a random function of the latent *utility* of the items. Inferring the utilities of items allows us to rank them, estimate numerical ratings and predict pairwise labels. Many popular instances of this approach, known as a *random utility model* (Thurstone 1927), are variants of the Bradley-Terry (BT) model (Bradley and Terry 1952; Plackett 1975; Luce 1959), which assumes a logistic likelihood, or the Thurstone-Mosteller model (Thurstone 1927; Mosteller 1951), which assumes a probit likelihood. Recent work on the BT model has developed computationally efficient active learning, but does not consider input features (Li et al. 2018). Another commonly-used ranking method, SVM-rank (Joachims 2002), predicts pairwise labels from input features without a random utility model, so cannot predict utilities. *Gaussian process preference learning (GPPL)* provides a Bayesian treatment of the random utility model, using input features to predict the utilities of test items and share information between similar items (Chu and Ghahramani 2005). As GPPL can only predict the preferences of a single user, we introduce a new, scalable approach to model individuals in a crowd.

Previous work on preference learning from crowdsourced data treats disagreements as annotation errors and infers only the consensus, rather than modelling personal preferences. For instance, Chen et al. (2013) and Wang et al. (2016) tackle annotator disagreement using Bayesian approaches that learn the labelling accuracy of each worker. Recently, Pan et al. (2018) and Han et al. (2018) introduced scalable methods that extend this idea from pairwise labels to noisy k-ary preferences, i.e., totally-ordered subsets of k items. Fu et al. (2016) improved SVM-rank by identifying outliers in crowdsourced data that correspond to probable errors, while Uchida et al. (2017) extend SVM-rank to account for different levels of confidence in each pairwise annotation expressed by the annotators. However, while these approaches differentiate the level of *noise* for each annotator, they ignore labelling *bias* as the differences between users are not random but depend on personal preferences toward particular items. With small numbers of labels per item, these biases may reduce the accuracy of the estimated consensus. Furthermore, previous aggregation methods for crowdsourced preferences do not consider item features, so cannot predict the utility of test items (Chen et al. 2013; Wang et al. 2016; Han et al. 2018; Pan et al. 2018; Li et al. 2018). Our approach goes beyond these methods by predicting personal preferences and incorporating input features.

A number of methods use *matrix factorisation* to predict personal preferences from pairwise labels, including Yi et al. (2013), who focus on small numbers of pairs per user, and Salimans et al. (2012), who apply Bayesian matrix factorisation to handle sparse data. Matrix factorisation represents observed ratings in a user-item matrix, which it decomposes into two matrices of lower rank than the user-item matrix, one corresponding to users and one to items. Users with similar ratings have similar columns in the user matrix, where each entry is a weight over a latent rating. By multiplying the low-dimensional representations, we can predict ratings for unseen user-item pairs. Kim et al. (2014) use a simplification that assumes that each user's preferences depend on only one latent ranking. However, previous works combining matrix factorisation with pairwise preference labels do not account for input features. This contrasts with work on matrix factorisation with side information, where the ratings or preferences as well as input features are directly observed, including recent neural network approaches (Volkovs et al. 2017), Bayesian approaches that concatenate input feature vectors with the low-dimensional factored representations (Porteous

et al. 2010), and GP-based methods (Adams et al. 2010). Besides providing a Bayesian method for matrix factorisation with both input features and pairwise labels, this paper introduces a much more scalable inference method for a GP-based model.

GPs were previously used for personal preference prediction by Guo et al. (2010), who propose a GP over the joint feature space of users and items. Since this scales cubically in the number of users, Abbasnejad et al. (2013) propose to cluster users into behavioural groups, but distinct clusters do not allow for collaborative learning between users whose preferences only partially overlap, e.g. when two users both like one genre of music, but have different preferences over other genres. Khan et al. (2014) instead learn a GP for each user, then add a matrix factorisation term that performs collaborative filtering. However, this approach does not model the relationship between input features and the low-rank matrices, unlike Lawrence and Urtasun (2009) who place GP priors over latent ratings. Neither of these last two methods are fully Bayesian as the users' weights are optimised rather than marginalised. An alternative is the *collaborative GP (collabGP)* (Houlsby et al. 2012), which places GP priors over user weights and latent factors, thereby exploiting input features for both users and items. However, unlike our approach, collabGP predicts only pairwise labels, not the utilities of items, which are useful for rating and ranking, and can only be trained using pairwise labels, even if observations of the utilities are available. Furthermore, existing GP-based approaches suffer from scalability issues and none of the previous methods jointly model the consensus as well as personal preferences in a fully-Bayesian manner.

Established methods for GP inference with non-Gaussian likelihoods, such as the Laplace approximation and expectation propagation (Rasmussen and Williams 2006), have time complexity $\mathcal{O}(N^3)$ with N data points and memory complexity $\mathcal{O}(N^2)$. For collabGP, Houlsby et al. (2012) use a sparse *generalized fully independent training conditional* (GFITC) approximation (Snelson and Ghahramani 2006) to reduce time complexity to $\mathcal{O}(PM^2 + UM^2)$ and memory complexity to $\mathcal{O}(PM + UM)$, where P is the number of pairwise labels, $M \ll P$ is a fixed number of inducing points, and U is the number of users. However, this is not sufficiently scalable for very large numbers of users or pairs, due to increasing memory consumption and optimisation steps that cannot be distributed. Recent work on distributing and parallelising Bayesian matrix factorisation is not easily applicable to models that incorporate GPs (Ahn et al. 2015; Saha et al. 2015; Vander Aa et al. 2017; Chen et al. 2018).

To handle large numbers of pairwise labels, Khan et al. (2014) subsample the data rather than learning from the complete training set. An alternative is *stochastic variational inference (SVI)* (Hoffman et al. 2013), which optimises a posterior approximation using a different subsample of training data at each iteration, meaning it learns from all training data over multiple iterations while limiting costs per iteration. SVI has been applied to GP regression (Hensman et al. 2013) and classification (Hensman et al. 2015), further improving scalability over earlier sparse approximations. Nguyen and Bonilla (2014) introduce SVI for multi-output GPs, where each output is a weighted combination of latent functions. They apply their method to capture dependencies between regression tasks, treating the weights for the latent functions as hyperparameters. In this paper, we introduce a Bayesian treatment of the weights and apply SVI instead to preference learning. An SVI method for GPPL was previously introduced by Simpson and Gurevych (2018), which we detail in Sect. 4. However, as GPPL does not consider the individual preferences of users in a crowd, we propose a new model, crowdGPPL, which jointly models personal preferences and the crowd consensus using a combination of Gaussian processes and Bayesian matrix factorisation.

3 Bayesian preference learning for crowds

We assume that a pair of items, a and b, have utilities $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$, which represent their value to a user, and that $f : \mathbb{R}^D \mapsto \mathbb{R}$ is a function of item features, where $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are vectors of length D containing the features of items a and b, respectively. If $f(\boldsymbol{x}_a) > f(\boldsymbol{x}_b)$, then a is preferred to b (written $a \succ b$). The outcome of a comparison between a and b is a pairwise label, $y(a, b)$. Assuming that pairwise labels never contain errors, then $y(a, b) = 1$ if $a \succ b$ and 0 otherwise. Given knowledge of f, we can compute the utilities of items in a test set given their features, and the outcomes of pairwise comparisons.

Thurstone (1927) proposed the random utility model, which relaxes the assumption that pairwise labels, $y(a, b)$, are always consistent with the ordering of $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$. Under the random utility model, the likelihood $p(y(a, b) = 1)$ increases as $f_a - f_b$ increases, i.e., as the utility of item a increases relative to the utility of item b. This reflects the greater consistency in a user's choices when their preferences are stronger, while accommodating labelling errors or variations in a user's choices over time. In the Thurstone-Mosteller model, noise in the observations is explained by a Gaussian-distributed noise term, $\delta \sim \mathcal{N}(0, \sigma^2)$:

$$p(y(a,b)|f(\boldsymbol{x}_a)+\delta_a, f(\boldsymbol{x}_b)+\delta_b) = \begin{cases} 1 & \text{if } f(\boldsymbol{x}_a)+\delta_a \geq f(b)+\delta_b \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

Integrating out the unknown values of δ_a and δ_b gives:

$$\begin{aligned} &p(y(a,b)|f(\boldsymbol{x}_a), f(\boldsymbol{x}_b)) \\ &= \int\int p(y(a,b)|f(\boldsymbol{x}_a)+\delta_a, f(\boldsymbol{x}_b)+\delta_b)\mathcal{N}(\delta_a;0,\sigma^2)\mathcal{N}(\delta_b;0,\sigma^2)d\delta_a d\delta_b = \Phi(z), \end{aligned} \tag{2}$$

where $z = \frac{f(\boldsymbol{x}_a)-f(\boldsymbol{x}_b)}{\sqrt{2\sigma^2}}$, and Φ is the cumulative distribution function of the standard normal distribution, meaning that $\Phi(z)$ is a probit likelihood.[1] This likelihood is also used by Chu and Ghahramani (2005) for Gaussian process preference learning (GPPL), but here we simplify the formulation by assuming that $\sigma^2 = 0.5$, which leads to z having a denominator of $\sqrt{2 \times 0.5} = 1$, hence $z = f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b)$. Instead, we model varying degrees of noise in the pairwise labels by scaling f itself, as we describe in the next section.

In practice, $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$ must be inferred from pairwise training labels, $\boldsymbol{y}$, to obtain a posterior distribution over their values. If this posterior is a multivariate Gaussian distribution, then the probit likelihood allows us to analytically marginalise $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$ to obtain the probability of a pairwise label:

$$p(y(a,b)|\boldsymbol{y}) = \Phi(\hat{z}), \hat{z} = \frac{\hat{f}_a - \hat{f}_b}{\sqrt{1 + C_{a,a} + C_{b,b} - 2C_{a,b}}}, \tag{3}$$

where $\hat{f}_a$ and $\hat{f}_b$ are the means and $\boldsymbol{C}$ is the posterior covariance matrix of the multivariate Gaussian over $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$. Unlike other choices for the likelihood, such as a sigmoid, the probit allows us to compute the posterior over a pairwise label without further approximation, hence we assume this pairwise label likelihood for our proposed preference learning model.

[1] Please note that a full list of symbols is provided for reference in "Appendix 5".

3.1 GPPL for single user preference learning

We can model the preferences of a single user by assuming a Gaussian process prior over the user's utility function, $f \sim \mathcal{GP}(0, k_\theta/s)$, where k_θ is a kernel function with hyperparameters θ and s is an inverse scale parameter. The kernel function takes numerical item features as inputs and determines the covariance between values of f for different items. The choice of kernel function and its hyperparameters controls the shape and smoothness of the function across the feature space and is often treated as a model selection problem. Kernel functions suitable for a wide range of tasks include the *squared exponential* and the *Matérn* (Rasmussen and Williams 2006), which both make minimal assumptions but assign higher covariance to items with similar feature values. We use k_θ to compute a covariance matrix $\boldsymbol{K}_\theta$, between a set of N observed items with features $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

Here we extend the original definition of GPPL (Chu and Ghahramani 2005), by introducing the inverse scale, s, which is drawn from a gamma prior, $s \sim \mathcal{G}(\alpha_0, \beta_0)$, with shape α_0 and scale β_0. The value of $1 / s$ determines the variance of f, and therefore the magnitude of differences between $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$ for items a and b. This in turn affects the level of certainty in the pairwise label likelihood as per Eq. 2.

Given a set of P pairwise labels, $\boldsymbol{y} = \{y_1, \ldots, y_P\}$, where $y_p = y(a_p, b_p)$ is the preference label for items a_p and b_p, we can write the joint distribution over all variables as follows:

$$p\big(\boldsymbol{y}, \boldsymbol{f}, s | k_\theta, \boldsymbol{X}, \alpha_0, \beta_0\big) = \prod_{p=1}^{P} p(y_p|\boldsymbol{f}) \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s) \mathcal{G}(s; \alpha_0, \beta_0) \tag{4}$$

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)\}$ is a vector containing the utilities of the N items referred to by $\boldsymbol{y}$, and $p(y_p|\boldsymbol{f}) = \Phi\big(z_p\big)$ is the pairwise likelihood (Eq. 2).

3.2 Crowd preference learning

To predict the preferences of individuals in a crowd, we could use an independent GPPL model for each user. However, by modelling all users jointly, we can exploit correlations between their interests to improve predictions when preference data is sparse, and reduce the memory cost of storing separate models. Correlations between users can arise from common interests over certain subsets of items, such as in one particular genre in a book recommendation task. Identifying such correlations helps to predict preferences from fewer observations and is the core idea of collaborative filtering (Resnick and Varian 1997) and matrix factorisation (Koren et al. 2009).

As well as individual preferences, we wish to predict the consensus by aggregating preference labels from multiple users. Individual biases of different users may affect consensus predictions, particularly when data for certain items comes from a small subset of users. The consensus could also help predict preferences of users with little or no data by favouring popular items and avoiding generally poor items. We therefore propose *crowdGPPL*, which jointly models the preferences of individual users as well as the underlying consensus of the crowd. Unlike previous methods for inferring the consensus, such as *CrowdBT* (Chen et al. 2013), we do not treat differences between users as simply the result of labelling errors, but also account for their subjective biases towards particular items.

For crowdGPPL, we represent utilities in a matrix, $\boldsymbol{F} \in \mathbb{R}^{N \times U}$, with U columns corresponding to users. Within $\boldsymbol{F}$, each entry $F_{a,j} = f(\boldsymbol{x}_a, \boldsymbol{u}_j)$ is the utility for item a for user j

with user features $\boldsymbol{u}_j$. We assume that $\boldsymbol{F} = \boldsymbol{V}^T\boldsymbol{W} + \boldsymbol{t}\boldsymbol{1}^T$ is the product of two low-rank matrices plus a column vector of consensus utilities, $\boldsymbol{t} \in \mathbb{R}^N$, where $\boldsymbol{W} \in \mathbb{R}^{C\times U}$ is a latent representation of the users, $\boldsymbol{V} \in \mathbb{R}^{C\times N}$ is a latent representation of the items, C is the number of latent *components*, i.e., the dimension of the latent representations, and $\boldsymbol{1}$ is a column vector of ones of length U. The column $\boldsymbol{v}_{.,a}$ of $\boldsymbol{V}$, and the column $\boldsymbol{w}_{.,j}$ of $\boldsymbol{W}$, are latent vector representations of item a and user j, respectively. Each row of $\boldsymbol{V}$, $\boldsymbol{v}_c = \{v_c(\boldsymbol{x}_1), \ldots, v_c(\boldsymbol{x}_N)\}$, contains evaluations of a latent function, $v_c \sim \mathcal{GP}(\boldsymbol{0}, k_\theta/s_c^{(v)})$, of item features, $\boldsymbol{x}_a$, where k is a kernel function, $s_c^{(v)}$ is an inverse function scale, and θ are kernel hyperparameters. The consensus utilities, $\boldsymbol{t} = \{t(\boldsymbol{x}_1), \ldots, t(\boldsymbol{x}_N)\}$, are values of a consensus utility function over item features, $t \sim \mathcal{GP}(\boldsymbol{0}, k_\theta/s^{(t)})$, which is shared across all users, with inverse scale $s^{(t)}$. Similarly, each row of $\boldsymbol{W}$, $\boldsymbol{w}_c = \{w_c(\boldsymbol{u}_1), \ldots, w_c(\boldsymbol{u}_U)\}$, contains evaluations of a latent function, $w_c \sim \mathcal{GP}(\boldsymbol{0}, k_\eta/s_c^{(w)})$, of user features, $\boldsymbol{u}_j$, with inverse scale $s_c^{(w)}$ and kernel hyperparameters η. Therefore, each utility in $\boldsymbol{F}$ can be written as a weighted sum over the latent components:

$$f(\boldsymbol{x}_a, \boldsymbol{u}_j) = \sum_{c=1}^{C} v_c(\boldsymbol{x}_a) w_c(\boldsymbol{u}_j) + t(\boldsymbol{x}_a), \tag{5}$$

where $\boldsymbol{u}_j$ are the features of user j and $\boldsymbol{x}_a$ are the features of item a. Each latent component corresponds to a utility function for certain items, which is shared by a subset of users to differing degrees. For example, in the case of book recommendation, c could relate to science fiction novels, v_c to a ranking over them, and w_c to the degree of agreement of users with that ranking. The individual preferences of each user j deviate from a consensus across users, t, according to $\sum_{c=1}^{C} v_c(\boldsymbol{x}_a) w_c(\boldsymbol{u}_j)$. This allows us to subtract the effect of individual biases when inferring the consensus utilities. The consensus can also help when inferring personal preferences for new combinations of users and items that are very different to those in the training data by accounting for any objective or widespread appeal that an item may have.

Although the model assumes a fixed number of components, C, the GP priors over $\boldsymbol{w}_c$ and $\boldsymbol{v}_c$ act as *shrinkage* or *ARD priors* that favour values close to zero (MacKay 1995; Psorakis et al. 2011). Components that are not required to explain the data will have posterior expectations and scales $1/s^{(v)}$ and $1/s^{(w)}$ approaching zero. Therefore, it is not necessary to optimise the value of C by hand, providing a sufficiently large number is chosen.

Equation 5 is similar to *cross-task crowdsourcing* (Mo et al. 2013), which uses matrix factorisation to model annotator performance in different tasks, where $\boldsymbol{t}$ corresponds to the objective difficulty of a task. However, unlike crowdGPPL, they do not use GPs to model the factors, nor apply the approach to preference learning. For preference learning, collabGP (Houlsby et al. 2012) is a related model that excludes the consensus and uses values in $\boldsymbol{v}_c$ to represent pairs rather than individual items, so does not infer item ratings. It also omits scale parameters for the GPs that encourage shrinkage when C is larger than required.

We combine the matrix factorisation method with the preference likelihood of Eq. 2 to obtain the joint preference model for multiple users, *crowdGPPL*:

$$\begin{aligned} &p\Big(\boldsymbol{y}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{t}, s_1^{(v)}, \ldots, s_C^{(v)}, s_1^{(w)}, \ldots, s_C^{(w)}, s^{(t)} | k_\theta, \boldsymbol{X}, k_\eta, \boldsymbol{U}, \alpha_0^{(t)}, \beta_0^{(t)}, \alpha_0^{(v)}, \beta_0^{(v)}, \alpha_0^{(w)}, \beta_0^{(w)}\Big) \\ &= \prod_{p=1}^{P} \Phi(z_p) \mathcal{N}(\boldsymbol{t}; \boldsymbol{0}, \boldsymbol{K}_\theta/s^{(t)}) \mathcal{G}\Big(s^{(t)}; \alpha_0^{(t)}, \beta_0^{(t)}\Big) \prod_{c=1}^{C} \Big\{ \mathcal{N}(\boldsymbol{v}_c; \boldsymbol{0}, \boldsymbol{K}_\theta/s_c^{(v)}) \\ &\quad \mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{L}_\eta/s_c^{(w)}) \mathcal{G}\Big(s_c^{(v)}; \alpha_0^{(v)}, \beta_0^{(v)}\Big) \mathcal{G}\Big(s_c^{(w)}; \alpha_0^{(w)}, \beta_0^{(w)}\Big) \Big\}, \end{aligned} \tag{6}$$

where $z_p = \boldsymbol{v}_{.,a_p}^T \boldsymbol{w}_{.,u_p} + t_{a_p} - \boldsymbol{v}_{.,b_p}^T \boldsymbol{w}_{.,u_p} - t_{b_p}$, index p refers to a user and a pair of items, $\{u_p, a_p, b_p\}$, $\boldsymbol{U}$ is the set of feature vectors for all users, $\boldsymbol{K}_\theta$ is the prior covariance for the items as in GPPL, and $\boldsymbol{L}_\eta$ is the prior covariance for the users computed using k_η.

4 Scalable inference

Given a set of pairwise training labels, $\boldsymbol{y}$, we aim to find the posterior over the matrix $\boldsymbol{F}^* = \boldsymbol{V}^{*T}\boldsymbol{W}^*$ of utilities for test items and test users, and the posterior over consensus utilities for test items, $\boldsymbol{t}^*$. The non-Gaussian likelihood (Eq. 2) makes exact inference intractable, hence previous work uses the Laplace approximation for GPPL (Chu and Ghahramani 2005) or combines expectation propagation (EP) with variational Bayes for a multi-user model (Houlsby et al. 2012). The Laplace approximation is a maximum a-posteriori solution that takes the most probable values of parameters rather than integrating over their distributions, and has been shown to perform poorly for classification compared to EP (Nickisch and Rasmussen 2008). However, a drawback of EP is that convergence is not guaranteed (Minka 2001). More importantly, inference for a GP using either method has computational complexity $\mathcal{O}(N^3)$ and memory complexity $\mathcal{O}(N^2)$, where N is the number of data points.

The cost of inference can be reduced using a *sparse* approximation based on a set of *inducing points*, which act as substitutes for the points in the training dataset. By choosing a fixed number of inducing points, $M \ll N$, the computational cost is cut to $\mathcal{O}(NM^2)$, and the memory complexity to $\mathcal{O}(NM)$. Inducing points must be selected using either heuristics or by optimising their positions to maximise an estimate of the marginal likelihood. One such sparse approximation is the *generalized fully independent training conditional* (GFITC) (Naish-guzman and Holden 2008; Snelson and Ghahramani 2006), used by Houlsby et al. (2012) for collabGP. However, time and memory costs that grow linearly with $\mathcal{O}(N)$ start to become a problem with thousands of data points, as all data must be processed in every iterative update, before any other parameters such as s are updated, making GFITC unsuitable for very large datasets (Hensman et al. 2015).

We derive a more scalable approach for GPPL and crowdGPPL using stochastic variational inference (SVI) (Hoffman et al. 2013). For GPPL, this reduces the time complexity of each iteration to $\mathcal{O}(P_i M^2 + P_i^2 M + M^3)$, and memory complexity to $\mathcal{O}(P_i M + M^2 + P_i^2)$, where P_i is a mini-batch size that we choose in advance. Neither P_i nor M are dependent on the size of the dataset, meaning that SVI can be run with arbitrarily large datasets, and other model parameters such as s can be updated before processing all data to encourage faster convergence. First, we define a suitable likelihood approximation to enable the use of SVI.

4.1 Approximating the posterior with a pairwise likelihood

The preference likelihood in Eq. 2 is not conjugate with the Gaussian process, which means there is no analytic expression for the exact posterior. For single-user GPPL, we therefore approximate the preference likelihood with a Gaussian:

$$p(\boldsymbol{f}|\boldsymbol{y},s) \propto \prod_{p=1}^{P} p\left(y_p|z_p\right) p(\boldsymbol{f}|\boldsymbol{K},s) = \prod_{p=1}^{P} \Phi\left(z_p\right) \mathcal{N}(\boldsymbol{f};\boldsymbol{0},\boldsymbol{K}/s)$$
$$\approx \prod_{p=1}^{P} \mathcal{N}\left(y_p;\Phi(z_p),Q_{p,p}\right) \mathcal{N}(\boldsymbol{f};\boldsymbol{0},\boldsymbol{K}/s) = \mathcal{N}(\boldsymbol{y};\Phi(\boldsymbol{z}),\boldsymbol{Q}) \mathcal{N}(\boldsymbol{f};\boldsymbol{0},\boldsymbol{K}/s), \tag{7}$$

where $\boldsymbol{Q}$ is a diagonal noise covariance matrix and we omit the kernel hyperparameters, θ, to simplify notation. For crowdGPPL, we use the same approximation to the likelihood, but replace $\boldsymbol{f}$ with $\boldsymbol{F}$. We estimate the diagonals of $\boldsymbol{Q}$ by moment matching our approximate likelihood with $\Phi(z_p)$, which defines a Bernoulli distribution with variance $Q_{p,p} = \Phi(z_p)(1 - \Phi(z_p))$. However, this means that $\boldsymbol{Q}$ depends on $\boldsymbol{z}$ and therefore on $\boldsymbol{f}$, so the approximate posterior over $\boldsymbol{f}$ cannot be computed in closed form. To resolve this, we approximate $Q_{p,p}$ using an estimated posterior over $\Phi(z_p)$ computed independently for each pairwise label, p. We obtain this estimate by updating the parameters of the conjugate prior for the Bernoulli likelihood, which is a beta distribution with parameters γ and λ. We find γ and λ by matching the moments of the beta prior to the prior mean and variance of $\Phi(z_p)$, estimated using numerical integration. The prior over $\Phi(z_p)$ is defined by a GP for single-user GPPL, $p(\Phi(z_p)|\boldsymbol{K},\alpha_0,\beta_0)$, and a non-standard distribution for crowdGPPL. Given the observed label y_p, we estimate the diagonals in $\boldsymbol{Q}$ as the variance of the posterior beta-Bernoulli:

$$Q_{p,p} \approx \frac{(\gamma + y_p)(\lambda + 1 - y_p)}{(\gamma + \lambda + 1)^2}. \tag{8}$$

The covariance $\boldsymbol{Q}$ therefore approximates the expected noise in the observations, hence captures variance due to σ in Eq. 2. This approximation performs well empirically for Gaussian process classification (Reece et al. 2011; Simpson et al. 2017) and classification using extended Kalman filters (Lee and Roberts 2010; Lowne et al. 2010).

Unfortunately, the nonlinear term $\Phi(\boldsymbol{z})$ means that the posterior is still intractable, so we replace $\Phi(\boldsymbol{z})$ with a linear function of $\boldsymbol{f}$ by taking the first-order Taylor series expansion of $\Phi(\boldsymbol{z})$ about the expectation $\mathbb{E}[\boldsymbol{f}] = \hat{\boldsymbol{f}}$:

$$\Phi(\boldsymbol{z}) \approx \tilde{\Phi}(\boldsymbol{z}) = \boldsymbol{G}\left(\boldsymbol{f} - \hat{\boldsymbol{f}}\right) + \Phi(\hat{\boldsymbol{z}}), \tag{9}$$

$$G_{p,i} = \frac{\partial \Phi(\hat{z}_p)}{\partial f_i} = \Phi(\hat{z}_p)\left(1 - \Phi(\hat{z}_p)\right)\left(2y_p - 1\right)\left([i = a_p] - [i = b_p]\right), \tag{10}$$

where $\hat{\boldsymbol{z}}$ is the expectation of $\boldsymbol{z}$ computed using Eq. 3, and $[i = a] = 1$ if $i = a$ and is 0 otherwise. There is a circular dependency between $\hat{\boldsymbol{f}}$, which is needed to compute $\hat{\boldsymbol{z}}$, and $\boldsymbol{G}$. We estimate these terms using a variational inference procedure that iterates between updating $\boldsymbol{f}$ and $\boldsymbol{G}$ (Steinberg and Bonilla 2014) as part of Algorithm 1. The complete approximate posterior for GPPL is now as follows:

$$p(\boldsymbol{f}|\boldsymbol{y},s) \approx \mathcal{N}(\boldsymbol{y};\boldsymbol{G}(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}]) + \Phi(\hat{\boldsymbol{z}}),\boldsymbol{Q}) \mathcal{N}(\boldsymbol{f};\boldsymbol{0},\boldsymbol{K}/s)/Z = \mathcal{N}\left(\boldsymbol{f};\hat{\boldsymbol{f}},\boldsymbol{C}\right), \tag{11}$$

where Z is a normalisation constant. Linearisation means that our approximate likelihood is conjugate to the prior, so the approximate posterior is also Gaussian. Gaussian approximations to the posterior have shown strong empirical results for classification (Nickisch and

Rasmussen 2008) and preference learning (Houlsby et al. 2012), and linearisation using a Taylor expansion has been widely tested in the extended Kalman filter (Haykin 2001) as well as Gaussian processes (Steinberg and Bonilla 2014; Bonilla et al. 2016).

4.2 SVI for single user GPPL

Using the linear approximation in the previous section, posterior inference requires inverting $\boldsymbol{K}$ with computational cost $\mathcal{O}(N^3)$ and taking an expectation with respect to s, which remains intractable. We address these problems using stochastic variational inference (SVI) with a sparse approximation to the GP that limits the size of the covariance matrices we need to invert. We introduce $M \ll N$ inducing items with inputs $\boldsymbol{X}_m$, utilities $\boldsymbol{f}_m$, and covariance $\boldsymbol{K}_{mm}$. The covariance between the observed and inducing items is $\boldsymbol{K}_{nm}$. For clarity, we omit θ from this point on. We assume a *mean-field* approximation to the joint posterior over inducing and training items that factorises between different sets of latent variables:

$$p\left(\boldsymbol{f},\boldsymbol{f}_m,s|\boldsymbol{y},\boldsymbol{X},\boldsymbol{X}_m,k_\theta,\alpha_0,\beta_0\right) \approx q\left(\boldsymbol{f},\boldsymbol{f}_m,s\right) = q(s)q(\boldsymbol{f})q\left(\boldsymbol{f}_m\right), \tag{12}$$

where $q(.)$ are *variational factors* defined below. Each factor corresponds to a subset of latent variables, $\boldsymbol{\zeta}_i$, and takes the form $\ln q(\boldsymbol{\zeta}_i) = \mathbb{E}_{j\neq i}[\ln p(\boldsymbol{\zeta}_i,\boldsymbol{x},\boldsymbol{y})]$. That is, the expectation with respect to all other latent variables, $\boldsymbol{\zeta}_j, \forall j \neq i$, of the log joint distribution of the observations and latent variables, $\boldsymbol{\zeta}_i$. To obtain the factor for $\boldsymbol{f}_m$, we marginalise $\boldsymbol{f}$ and take expectations with respect to $q(s)$:

$$\ln q\left(\boldsymbol{f}_m\right) = \ln \mathcal{N}\left(\boldsymbol{y};\tilde{\Phi}(z),\boldsymbol{Q}\right) + \ln \mathcal{N}\left(\boldsymbol{f}_m;\boldsymbol{0},\frac{\boldsymbol{K}_{mm}}{\mathbb{E}[s]}\right) + \text{const} = \ln \mathcal{N}\left(\boldsymbol{f}_m;\hat{\boldsymbol{f}}_m,\boldsymbol{S}\right), \tag{13}$$

where the variational parameters $\hat{\boldsymbol{f}}_m$ and $\boldsymbol{S}$ are computed using an iterative SVI procedure described below. We choose an approximation of $q(\boldsymbol{f})$ that depends only on the inducing point utilities, $\boldsymbol{f}_m$, and is independent of the observations:

$$\ln q(\boldsymbol{f}) = \ln \mathcal{N}\left(\boldsymbol{f};\boldsymbol{A}\hat{\boldsymbol{f}}_m,\boldsymbol{K} + \boldsymbol{A}\left(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s]\right)\boldsymbol{A}^T\right), \tag{14}$$

where $\boldsymbol{A} = \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}$. Therefore, we no longer need to invert an $N \times N$ covariance matrix to compute $q(\boldsymbol{f})$. The factor $q(s)$ also depends only the inducing points:

$$\ln q(s) = \mathbb{E}_{q(\boldsymbol{f}_m)}\left[\ln \mathcal{N}\left(\boldsymbol{f}_m|\boldsymbol{0},\boldsymbol{K}_{mm}/s\right)\right] + \ln \mathcal{G}(s;\alpha_0,\beta_0) + \text{const} = \ln \mathcal{G}(s;\alpha,\beta), \tag{15}$$

where $\alpha = \alpha_0 + \frac{M}{2}$ and $\beta = \beta_0 + \frac{1}{2}\text{tr}\left(\boldsymbol{K}_{mm}^{-1}\left(S + \hat{\boldsymbol{f}}_m\hat{\boldsymbol{f}}_m^T\right)\right)$. The expected value is $\mathbb{E}[s] = \frac{\alpha}{\beta}$.

We apply variational inference to iteratively reduce the KL-divergence between our approximate posterior and the true posterior (Eq. 12) by maximising a lower bound, $\mathcal{L}$, on the log marginal likelihood (detailed equations in "Appendix 1"), which is given by:

$$\begin{aligned} \ln p\left(\boldsymbol{y}|\boldsymbol{K},\alpha_0,\beta_0\right) &= \text{KL}\left(q\left(\boldsymbol{f},\boldsymbol{f}_m,s\right)||p\left(\boldsymbol{f},\boldsymbol{f}_m,s|\boldsymbol{y},\boldsymbol{K},\alpha_0,\beta_0\right)\right) + \mathcal{L} \\ \mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f})}\left[\ln p(\boldsymbol{y}|\boldsymbol{f})\right] + \mathbb{E}_{q(\boldsymbol{f}_m,s)}\left[\ln p\left(\boldsymbol{f}_m,s|\boldsymbol{K},\alpha_0,\beta_0\right)\right. \\ &\quad \left. - \ln q\left(\boldsymbol{f}_m\right) - \ln q(s)\right]. \end{aligned} \tag{16}$$

To optimise $\mathcal{L}$, we initialise the q factors randomly, then update each one in turn, taking expectations with respect to the other factors.

The only term in $\mathcal{L}$ that refers to the observations, $\boldsymbol{y}$, is a sum of P terms, each of which refers to one observation only. This means that $\mathcal{L}$ can be maximised by considering a random subset of observations at each iteration (Hensman et al. 2013). For the ith update of $q(\boldsymbol{f}_m)$, we randomly select P_i observations $\boldsymbol{y}_i = \{y_p \forall p \in \boldsymbol{P}_i\}$, where $\boldsymbol{P}_i$ is a random subset of indexes of observations, and P_i is a mini-batch size. The items referred to by the pairs in the subset are $\boldsymbol{N}_i = \{a_p \forall p \in \boldsymbol{P}_i\} \cup \{b_p \forall p \in \boldsymbol{P}_i\}$. We perform updates using $\boldsymbol{Q}_i$ (rows and columns of $\boldsymbol{Q}$ for pairs in $\boldsymbol{P}_i$), $\boldsymbol{K}_{im}$ and $\boldsymbol{A}_i$ (rows of $\boldsymbol{K}_{nm}$ and $\boldsymbol{A}$ in $\boldsymbol{N}_i$), $\boldsymbol{G}_i$ (rows of $\boldsymbol{G}$ in $\boldsymbol{P}_i$ and columns in $\boldsymbol{N}_i$), and $\hat{\boldsymbol{z}}_i = \{\hat{z}_p \forall p \in P_i\}$. The updates optimise the natural parameters of the Gaussian distribution by following the natural gradient (Hensman et al. 2015):

$$\boldsymbol{S}_i^{-1} = (1 - \rho_i)\boldsymbol{S}_{i-1}^{-1} + \rho_i\left(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1} + \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i\right) \tag{17}$$

$$\hat{\boldsymbol{f}}_{m,i} = \boldsymbol{S}_i\left((1 - \rho_i)\boldsymbol{S}_{i-1}^{-1}\hat{\boldsymbol{f}}_{m,i-1} + \rho_i \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1}\left(\boldsymbol{y}_i - \Phi(\hat{\boldsymbol{z}}_i) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{f}}_{m,i-1}\right)\right) \tag{18}$$

where $\rho_i = (i + \epsilon)^{-r}$ is a mixing coefficient that controls the update rate, $\pi_i = \frac{P}{P_i}$ weights each update according to sample size, ϵ is a delay hyperparameter and r is a forgetting rate (Hoffman et al. 2013).

By performing updates in terms of mini-batches, the time complexity of Eqs. 17 and 18 is $\mathcal{O}(P_i M^2 + P_i^2 M + M^3)$ and memory complexity is $\mathcal{O}(M^2 + P_i^2 + MP_i)$. The only parameters that must be stored between iterations relate to the inducing points, hence the memory consumption does not grow with the dataset size as in the GFITC approximation used by Houlsby et al. (2012). A further advantage of stochastic updating is that the s parameter (and any other global parameters not immediately depending on the data) can be learned before the entire dataset has been processed, which means that poor initial estimates of s are rapidly improved and the algorithm can converge faster.

Input: Pairwise labels, $\boldsymbol{y}$, training item features, $\boldsymbol{x}$, test item features $\boldsymbol{x}^*$
1 Select inducing point locations $\boldsymbol{x}_{mm}$ and compute kernel matrices $\boldsymbol{K}$, $\boldsymbol{K}_{mm}$ and $\boldsymbol{K}_{nm}$ given $\boldsymbol{x}$;
2 Initialise $\mathbb{E}[s]$ and $\hat{\boldsymbol{f}}_m$ to prior means and $\boldsymbol{S}$ to prior covariance $\boldsymbol{K}_{mm}$;
while $\mathcal{L}$ *not converged* **do**
3 Select random sample, $\boldsymbol{P}_i$, of P observations;
  while $\boldsymbol{G}_i$ *not converged* **do**
4 Compute $\mathbb{E}[\boldsymbol{f}_i]$;
5 Compute $\boldsymbol{G}_i$ given $\mathbb{E}[\boldsymbol{f}_i]$;
6 Compute $\hat{\boldsymbol{f}}_{m,i}$ and $\boldsymbol{S}_i$;
  end
7 Update $q(s)$ and compute $\mathbb{E}[s]$ and $\mathbb{E}[\ln s]$;
end
8 Compute kernel matrices for test items, $\boldsymbol{K}_{**}$ and $\boldsymbol{K}_{*m}$, given $\boldsymbol{x}^*$;
9 Use converged values of $\mathbb{E}[\boldsymbol{f}]$ and $\hat{\boldsymbol{f}}_m$ to estimate posterior over $\boldsymbol{f}^*$ at test points ;
Output: Posterior mean of the test values, $\mathbb{E}[\boldsymbol{f}^*]$, and covariance, $\boldsymbol{C}^*$

Algorithm 1: The SVI algorithm for GPPL: preference learning with a single user.

The complete SVI algorithm is summarised in Algorithm 1. It uses a nested loop to learn $\boldsymbol{G}_i$, which avoids storing the complete matrix, $\boldsymbol{G}$. It is possible to distribute computation in lines 3-6 by selecting multiple random samples to process in parallel. A global estimate of $\hat{\boldsymbol{f}}_m$ and $\boldsymbol{S}$ is passed to each compute node, which runs the loop over lines 4 to 6. The resulting

updated $\hat{\boldsymbol{f}}_m$ and $\boldsymbol{S}$ values are then passed back to a central node that combines them by taking a mean weighted by π_i to account for the size of each batch.

Inducing point locations can be learned as part of the variational inference procedure, which breaks convergence guarantees, or by an expensive optimisation process (Hensman et al. 2015). We obtain good performance by choosing inducing points up-front using K-means++ (Arthur and Vassilvitskii 2007) with M clusters to cluster the feature vectors, then taking the cluster centres as inducing points that represent the distribution of observations.

The inferred distribution over the inducing points can be used to estimate the posteriors of test items, $f(\boldsymbol{x}^*)$, according to:

$$\boldsymbol{f}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m, \quad \boldsymbol{C}^* = \boldsymbol{K}_{**} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{*m}^T, \tag{19}$$

where $\boldsymbol{C}^*$ is the posterior covariance of the test items, $\boldsymbol{K}_{**}$ is their prior covariance, and $\boldsymbol{K}_{*m}$ is the covariance between test and inducing items.

4.3 SVI for crowdGPPL

We now provide the variational posterior for the crowdGPPL model defined in Eq. 6:

$$\begin{aligned} &p\Big(\boldsymbol{V}, \boldsymbol{V}_m, \boldsymbol{W}, \boldsymbol{W}_m, \boldsymbol{t}, \boldsymbol{t}_m, s_1^{(v)}, \ldots, s_C^{(v)}, s_1^{(w)}, \ldots, s_C^{(w)}, s^{(t)} | \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_m, \boldsymbol{U}, \boldsymbol{U}_m, k, \alpha_0, \beta_0\Big) \\ &\quad \approx q(\boldsymbol{t})q(\boldsymbol{t}_m)q\big(s^{(t)}\big)\prod_{c=1}^{C} q(\boldsymbol{v}_c)q(\boldsymbol{w}_c)q(\boldsymbol{v}_{c,m})q(\boldsymbol{w}_{c,m})q\big(s_c^{(v)}\big)q\big(s_c^{(w)}\big), \end{aligned} \tag{20}$$

where $\boldsymbol{U}_m$ are the feature vectors of inducing users and the variational q factors are defined below. We use SVI to optimise the lower bound on the log marginal likelihood (detailed in "Appendix 2"), which is given by:

$$\begin{aligned} \mathcal{L}_{cr} = {} & \mathbb{E}_{q(\boldsymbol{F})}[\ln p(\boldsymbol{y}|\boldsymbol{F})] + \mathbb{E}_{q(\boldsymbol{t}_m, s^{(t)})}\Big[\ln p\Big(\boldsymbol{t}_m, s^{(t)}|\boldsymbol{K}_{mm}, \alpha_0^{(t)}, \beta_0^{(t)}\Big) - \ln q(\boldsymbol{t}_m) - \ln q\big(s^{(t)}\big)\Big] \\ & + \sum_{c=1}^{C}\Bigg\{\mathbb{E}_{q\left(\boldsymbol{v}_{m,c}, s_c^{(v)}\right)}\Big[\ln p\Big(\boldsymbol{v}_{m,c}, s_c^{(v)}|\boldsymbol{K}_{mm}, \alpha_0^{(v)}, \beta_0^{(v)}\Big) - \ln q(\boldsymbol{v}_{m,c}) - \ln q\big(s_c^{(v)}\big)\Big] \\ & + \mathbb{E}_{q\left(\boldsymbol{w}_{m,c}, s_c^{(w)}\right)}\Big[\ln p\Big(\boldsymbol{w}_{m,c}, s_c^{(w)}|\boldsymbol{L}_{mm}, \alpha_0^{(w)}, \beta_0^{(w)}\Big) - \ln q(\boldsymbol{w}_{m,c}) - \ln q\big(s_c^{(w)}\big)\Big]\Bigg\}. \end{aligned} \tag{21}$$

The SVI algorithm follows the same pattern as Algorithm 1, updating each q factor in turn by computing means and covariances for $\boldsymbol{V}_m$, $\boldsymbol{W}_m$ and $\boldsymbol{t}_m$ instead of $\boldsymbol{f}_m$ (see Algorithm 2). The time and memory complexity of each update are $\mathcal{O}(CM_{\text{items}}^3 + CM_{\text{items}}^2 P_i + CM_{\text{items}}P_i^2 + CM_{\text{users}}^3 + CM_{\text{users}}^2 P_i + CM_{\text{users}}P_i^2)$ and $\mathcal{O}(CM_{\text{items}}^2 + P_i^2 + M_{\text{items}}P_i + CM_{\text{users}}^2 + M_{\text{users}}P_i)$, respectively. The variational factor for the cth inducing item component is:

$$\begin{aligned} \ln q(\boldsymbol{v}_{m,c}) &= \mathbb{E}_{q(\boldsymbol{t}, \boldsymbol{w}_{m,c'}\forall c', \boldsymbol{v}_{m,c'}\forall c' \setminus c)}\big[\ln \mathcal{N}\big(\boldsymbol{y}; \tilde{\Phi}(z), Q\big)\big] + \ln \mathcal{N}\left(\boldsymbol{v}_{m,c}; \boldsymbol{0}, \frac{\boldsymbol{K}_{mm}}{\mathbb{E}\left[s_c^{(v)}\right]}\right) + \text{const} \\ &= \ln \mathcal{N}\big(\boldsymbol{v}_{m,c}; \hat{\boldsymbol{v}}_{m,c}, \boldsymbol{S}_c^{(v)}\big), \end{aligned} \tag{22}$$

where posterior mean $\hat{\boldsymbol{v}}_{m,c}$ and covariance $\boldsymbol{S}_c^{(v)}$ are computed using equations of the same form as Eqs. 17 and 18, except $\boldsymbol{Q}^{-1}$ is scaled by expectations over $\boldsymbol{w}_{m,c}$, and $\hat{\boldsymbol{f}}_{m,i}$ is replaced by $\hat{\boldsymbol{v}}_{m,c,i}$. The factor for the inducing points of $\boldsymbol{t}$ follows a similar pattern to $\boldsymbol{v}_{m,c}$:

$$\begin{aligned} \ln q(\boldsymbol{t}_m) &= \mathbb{E}_{q(\boldsymbol{w}_{m,c}\forall c, \boldsymbol{v}_{m,c}\forall c)}\left[\ln \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \ln \mathcal{N}\left(\boldsymbol{t}_m; \boldsymbol{0}, \frac{\boldsymbol{K}_{mm}}{\mathbb{E}[s^{(t)}]}\right) + \text{const} \\ &= \ln \mathcal{N}\left(\boldsymbol{t}_m; \hat{\boldsymbol{t}}_m, \boldsymbol{S}^{(t)}\right), \end{aligned} \tag{23}$$

where the equations for $\hat{\boldsymbol{t}}$ and $\boldsymbol{S}^{(t)}$ are the same as Eqs. 17 and 18, except $\hat{\boldsymbol{f}}_{m,i}$ is replaced by $\hat{\boldsymbol{t}}_{m,i}$. Finally, the variational distribution for each inducing user's component is:

$$\begin{aligned} \ln q(\boldsymbol{w}_{m,c}) =& \mathbb{E}_{q(\boldsymbol{t}, \boldsymbol{w}_{m,c'}\forall c' \setminus c, \boldsymbol{v}_{m,c'}\forall c')}\left[\ln \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \ln \mathcal{N}\left(\boldsymbol{w}_{m,c}; \boldsymbol{0}, \frac{\boldsymbol{L}_{mm}}{\mathbb{E}[s_c^{(w)}]}\right) + \text{const} \\ =& \ln \mathcal{N}\left(\boldsymbol{w}_{m,c}; \hat{\boldsymbol{w}}_{m,c}, \boldsymbol{\Sigma}_c\right), \end{aligned} \tag{24}$$

where $\hat{\boldsymbol{w}}_c$ and $\boldsymbol{\Sigma}_c$ also follow the pattern of Eqs. 17 and 18, with $\boldsymbol{Q}^{-1}$ scaled by expectations of $\boldsymbol{w}_{c,m}$, and $\hat{\boldsymbol{f}}_{m,i}$ replaced by $\hat{\boldsymbol{w}}_{m,c,i}$. We provide the complete equations for the variational means and covariances for $\boldsymbol{v}_{m,c}$, $\boldsymbol{t}_m$ and $\boldsymbol{w}_{m,c}$ in "Appendix 3". The expectations for inverse scales, $s_1^{(v)}, \ldots, s_c^{(v)}$, $s_1^{(w)}, \ldots, s_c^{(w)}$ and $s^{(t)}$ can be computed using Eq. 15 by substituting the corresponding terms for $\boldsymbol{v}_c$, $\boldsymbol{w}_c$ or $\boldsymbol{t}$ instead of $\boldsymbol{f}$.

Predictions for crowdGPPL can be made by computing the posterior mean utilities, $\boldsymbol{F}^*$, and the covariance $\boldsymbol{\Lambda}_u^*$ for each user, u, in the test set:

$$\boldsymbol{F}^* = \hat{\boldsymbol{t}}^* + \sum_{c=1}^{C} \hat{\boldsymbol{v}}_c^{*T} \hat{\boldsymbol{w}}_c^*, \qquad \boldsymbol{\Lambda}_u^* = \boldsymbol{C}_t^* + \sum_{c=1}^{C} \omega_{c,u}^* \boldsymbol{C}_{v,c}^* + \hat{w}_{c,u}^2 \boldsymbol{C}_{v,c}^* + \omega_{c,u}^* \hat{\boldsymbol{v}}_c \hat{\boldsymbol{v}}_c^T, \tag{25}$$

where $\hat{\boldsymbol{t}}^*$, $\hat{\boldsymbol{v}}_c^*$ and $\hat{\boldsymbol{w}}_c^*$ are posterior test means, $\boldsymbol{C}_t^*$ and $\boldsymbol{C}_{v,c}^*$ are posterior covariances of the test items, and $\omega_{c,u}^*$ is the posterior variance of the user components for u. (see "Appendix 4", Eqs. 39 to 41). The mean $\boldsymbol{F}^*$ and covariances Λ_u^* can be inserted into Eq. 2 to predict pairwise labels. In practice, the full covariance terms are needed only for Eq. 2, so need only be computed between items for which we wish to predict pairwise labels.

5 Experiments

Our experiments test key aspects of crowdGPPL: predicting consensus utilities and personal preferences from pairwise labels and the scalability of our proposed SVI method. In Sect. 5.1, we use simulated data to test the robustness of crowdGPPL to noise and unknown numbers of latent components. Section 5.2 compares different configurations of the model against alternative methods using the *Sushi* datasets[2] (Kamishima 2003). Section 5.3 evaluates prediction performance and scalability of crowdGPPL in a high-dimensional NLP task with sparse, noisy crowdsourced preferences (*UKPConvArgCrowdSample*,[3] Simpson

[2] http://www.kamishima.net/sushi/.

[3] https://github.com/ukplab/tacl2018-preference-convincing.

Table 1 Summary of datasets showing average counts for the training and test sets used in each fold/subsample

Dataset	#folds/ samples	#users	total #items	training set #pairs	test set #pairs	test set #items	#features items	#features users
Simulation a and b	25	25	100	900	0	100	2	2
Simulation c	25	25	100	36–2304	0	100	2	2
Sushi A-small	25	100	10	500	2500	10	18	123
Sushi A	25	100	10	2000	2500	10	18	123
Sushi B	25	5000	100	50000	5000	100	18	123
UKPConvArgCrowdSample	32	1442	1052	16398	529	33	32310	0

The test sets all contain gold-standard rankings over items as well as pairwise labels, except the simulations, which are not generated as we evaluate using the rankings only. Numbers of features are given after categorical labels have been converted to one-hot encoding, counting each category as a separate feature

and Gurevych (2018)). Finally, Sect. 5.4 evaluates whether crowdGPPL ignores redundant components. The datasets are summarised in Table 1.

As baselines, we compare crowdGPPL against *GPPL*, which we train on all users' preference labels to learn a single utility function, and *GPPL-per-user*, in which a separate GPPL instance is learned for each user with no collaborative learning. We also compare against the *GPVU* model (Khan et al. 2014) and *collabGP* (Houlsby et al. 2012). CollabGP contains parameters for each pairwise label and each user, so has a larger memory footprint than our SVI scheme, which stores only the moments at the inducing points.

We test *crowdBT* (Chen et al. 2013) as part of a method for predicting consensus utilities from crowdsourced pairwise preferences. CrowdBT models each worker's accuracy, assuming that the differences between workers' labels are due to random errors rather than subjective preferences. Since crowdBT does not account for the item features, it cannot predict utilities for items that were not part of the training set. We therefore treat the posterior mean utilities produced by crowdBT as training labels for Gaussian process regression using SVI. We set the observation noise variance of the GP equal to the crowdBT posterior variance of the utilities to propagate uncertainty from crowdBT to the GP. This pipeline method, *crowdBT–GP*, tests whether it is sufficient to treat annotator differences as noise, in contrast to the crowdGPPL approach of modelling individual preferences.

We evaluate the methods using the following metrics: *accuracy (acc)*, which is the fraction of correct pairwise labels; *cross entropy error (CEE)* between the posterior probabilities over pairwise labels and the true labels, which captures the quality of the pairwise posterior; and *Kendall's* τ, which evaluates the ranking obtained by sorting items by predicted utility.

5.1 Simulated noisy data

First, we evaluate whether crowdGPPL is able to model individual preferences with varying amounts of labelling noise. We set the number of latent components to $C = 20$ and all Gamma hyperparameters for crowdGPPL, GPPL and GPPL-per-user to $\alpha_0 = 1$, $\beta_0 = 100$. We use Matérn 3/2 kernels with the length-scale for each dimension of the feature vector, d, chosen by a median heuristic:

$$l_{d,\mathrm{MH}} = \mathrm{median}(\{||x_{i,d} - x_{j,d}||, \forall i = 1, \ldots, N, \forall j = 1, \ldots, N\}). \tag{26}$$

This is a computationally frugal way to choose the length-scales, that has been extensively used in various kernel methods (e.g., Bors and Pitas (1996); Gretton et al. (2012)). The SVI hyperparameters were set to $\rho = 0.9$, $P_i = 1000$ and $\epsilon = 1$. Hoffman et al. (2013) found that higher values of ρ gave better final results but slightly slower convergence, recommending 0.9 as a good balance across several datasets, and did not find any effect from changing ϵ. We follow their recommendations and do not find it necessary to perform further tuning in our experiments. Both M and P_i are constrained in practice by the computational resources available—we investigate these further in Sect. 5.3.

In simulation (a), to test consensus prediction, we generate a 20×20 grid of points and split them into 50% training and test sets. For each gridpoint, we generate pairwise labels by drawing from the generative model of crowdGPPL with $U = 20$ users, $C = 5$, each $s_c^{(v)}$ set to random values between 0.1 and 10, and $s_c^{(w)} = 1, \forall c$. We vary $s^{(t)}$ to control the noise in the consensus function. We train and test crowdGPPL with $C = U$ and repeat the complete experiment 25 times, including generating new data.

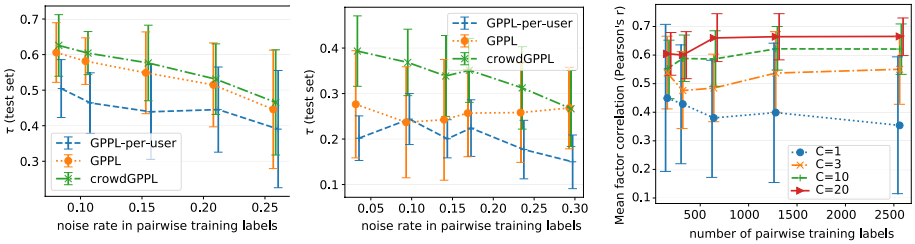
Figure 1a shows that crowdGPPL better recovers the consensus ranking than the baselines, even as noise increases, as GPPL's predictions are worsened by biased users who deviate consistently from the consensus. For GPPL-per-user, the consensus is simply the mean of all users' predicted utilities, so does not benefit from sharing information between users when training. For simulation (b), we modify the previous setup by fixing $s^{(t)} = 5$ and varying $s_c^{(v)}, \forall c$ to evaluate the methods' ability to recover the personal preferences of simulated users. The results in Fig. 1b show that crowdGPPL is able to make better predictions when noise is below 0.3.

We hypothesise that crowdGPPL can recover latent components given sufficient training data. In simulation (c), we generate data using the same setup as before, but fix $s^{(t)} = s_c^{(v)} = s^{(w)} = 1, \forall c$ and vary the number of pairwise training labels and the number of true components through $C_{\mathrm{true}} \in \{1, 3, 10, 20\}$. We match inferred components to the true components as follows: compute Pearson correlations between each unmatched true component and each unmatched inferred component; select the pair with the highest correlation as a match; repeat until all true components are matched. In Fig. 1c we plot the mean correlation between matched pairs of components. For all values of C_{true}, increasing the number of training labels beyond 700 brings little improvement. Performance is highest when $C_{\mathrm{true}} = 20$, possibly because the predictive model has $C = 20$, so is a closer match to the generating model. However, crowdGPPL is able to recover latent components reasonably well for all values of C_{true} given > 500 labels, despite mismatches between C and C_{true}.

5.2 Sushi preferences

The sushi datasets contain, for each user, a gold standard preference ranking of 10 types of sushi, from which we generate gold-standard pairwise labels. To test performance with very few training pairs, we obtain *Sushi-A-small* by selecting 100 users at random from the complete *Sushi-A* dataset, then selecting 5 pairs for training and 25 for testing per user. For *Sushi-A*, we select 100 users at random from the complete dataset, then split the data into training and test sets by randomly selecting 20 training and 25 test pairs per user. For *Sushi-B*, we use all 5000 workers, and subsample 10 training and 1 test pair per user.

We compare standard crowdGPPL with four other variants:



(a) Consensus **(b)** Personal preferences **(c)** Latent factors

Fig. 1 Simulations: rank correlation between true and inferred utilities. **a** and **b** vary the level of noise in pairwise training labels, **c** varies the number of pairwise training labels

- *crowdGPPL\inducing*: does not use the sparse inducing point approximation and instead uses all the original points in the training set;
- *crowdGPPL**$\boldsymbol{u}$: ignores the user features;
- *crowdGPPL**$\boldsymbol{u}\backslash\boldsymbol{x}$: ignores both user and item features;
- *crowdGPPL**$\boldsymbol{u}\backslash\boldsymbol{t}$: excludes the consensus function $\boldsymbol{t}$ from the model as well as the user features.

For methods with $\backslash\boldsymbol{u}$, the user covariance matrix, $\boldsymbol{L}$, is replaced by the identity matrix, and for crowdGPPL$\backslash\boldsymbol{u}\backslash\boldsymbol{x}$, $\boldsymbol{K}$ is also replaced by the identity matrix. As the user features do not contain detailed, personal information (only region, age group, gender, etc.), they are not expected to be sufficiently informative to predict personal preferences on their own. Therefore, for crowdGPPL and crowdGPPL\inducing, we compute $\boldsymbol{L}$ for 10 latent components using the Matérn 3/2 kernel function and use the identity matrix for the remaining 10. CollabGP is also tested with and without user features. We set hyperparameters $C = 20$, $\epsilon = 1$, $\rho = 0.9$, $P_i = 200$ for *Sushi-A-small* and *Sushi-A*, and $P_i = 2000$ for *Sushi-B*, without optimisation. For the gamma hyperparameters, a grid search over $\{10^{-1}, \ldots, 10^3\}$ on withheld user data from *Sushi-A* resulted in $\alpha_0 = 1, \beta_0 = 100$ for GPPL variants, and $\alpha_0^{(t)} = 1, \beta_0^{(t)} = 100$, $\alpha_0^{(v)} = 1, \beta_0^{(v)} = 10$ and $\alpha_0^{(w)} = 1, \beta_0^{(w)} = 10$ for crowdGPPL variants. The complete process of subsampling, training and testing, was repeated 25 times for each dataset.

The results in Table 2 illustrate the benefit of personalised models over single-user GPPL. The inducing point approximation does not appear to harm performance of crowdGPPL, but including the user features tends to decrease its performance compared to crowdGPPL$\backslash\boldsymbol{u}$ and crowdGPPL$\backslash\boldsymbol{u}\backslash\boldsymbol{x}$, except on Sushi-A-small, where they may help with the small amount of training data. Comparing crowdGPPL$\backslash\boldsymbol{u}$ with crowdGPPL$\backslash\boldsymbol{u}\backslash\boldsymbol{t}$, including the consensus function improves performance modestly. The strong performance of GPPL-per-user suggests that even 10 pairs per person were enough to learn a reasonable model for *Sushi-B*. As expected, the more memory-intensive collabGP performs comparably well to crowdGPPL on accuracy and CEE but does not provide a ranking function for computing Kendall's τ. GPVU does not perform as well as other personalised methods on Sushi-A and Sushi-B, potentially due to its maximum likelihood inference steps. The results show that crowdGPPL is competitive despite the approximate SVI method, so in the next experiment, we test the approach on a larger crowdsourced dataset where low memory consumption is required.

Table 2 Predicting personal preferences on *Sushi* datasets, means over 25 repeats

Method	**Sushi-A-small**			**Sushi-A**			**Sushi-B**		
	Acc	CEE	τ	Acc	CEE	τ	Acc	CEE	τ
crowdGPPL	**.71**	**.56**	.48	.84	.33	.79	.76	.50	. 54
crowdGPPL \inducing	.70	.60	.45	.84	.34	.78	–	–	–
crowdGPPL $\backslash \boldsymbol{u}$	.70	.58	.46	**.85**	**.31**	**.80**	**.78**	.50	.57
crowdGPPL $\backslash \boldsymbol{u} \backslash \boldsymbol{x}$	**.71**	.57	**.49**	**.85**	.33	**.80**	.77	**.49**	.56
crowdGPPL $\backslash \boldsymbol{u}, \backslash \boldsymbol{t}$	.68	.60	.43	.84	.33	**.80**	.76	.51	.58
GPPL	.65	.62	.31	.65	.62	.31	.65	.62	.31
GPPL-per-user	.67	.64	.42	.83	.40	.79	.75	.60	**.60**
collabGP	.69	.58	n/a	.83	.35	n/a	.76	**.49**	n/a
collabGP$\backslash \boldsymbol{u}$	.69	.59	n/a	.84	.33	n/a	.76	.50	n/a
GPVU	.70	.67	.43	.72	.67	.42	.73	.59	.52

The standard deviations are ≤ 0.02 for all accuracies, ≤ 0.08 for all CEE, and ≤ 0.03 for all τ. For Sushi-B, crowdGPPL, GPPL-per-user and collabGP had runtimes of 30 min on a 12 core, 2.6 GHz CPU server; GPPL required only 1 min

Bold values indicate the best performance in each column

5.3 Argument convincingness

We evaluate consensus learning, personal preference learning and scalability on an NLP task, namely, ranking arguments by *convincingness*. The task requires learning from crowdsourced data, but is not simply an aggregation task as it requires learning a predictor for test documents that were not compared by the crowd. The dataset, *UKPConvArgCrowdSample*, was subsampled by Simpson and Gurevych (2018) from raw data provided by Habernal and Gurevych (2016), and contains arguments written by users of online debating forums, with crowdsourced judgements of pairs of arguments indicating the most convincing argument. The data is divided into 32 folds (16 topics, each with 2 opposing stances). For each fold, we train on 31 folds and test on the remaining fold. We extend the task to predicting both the consensus and personal preferences of individual crowd workers. GPPL previously outperformed SVM and Bi-LSTM methods at consensus prediction for *UKPConvArgCrowdSample* (Simpson and Gurevych 2018). We hypothesise that a worker's view of convincingness depends on their personal view of the subject discussed, so crowdGPPL may outperform GPPL and crowdBT-GP on both consensus and personal preference prediction.

The dataset contains 32, 310 linguistic and embedding features for each document (we use mean GloVe embeddings for the words in each document, see Simpson and Gurevych (2018)). The high-dimensionality of the input feature vectors requires us to modify the length-scale heuristic for all GP methods, as the distance between items grows with the number of dimensions, which causes the covariance to shrink to very small values. We therefore use $l_{d,\text{scaledMH}} = 20\sqrt{D} \times l_{d,\text{MH}}$, where D is the dimension of the input feature vectors, and the scale was chosen by comparing the training set accuracy with scales in $\{\sqrt{D}, 10\sqrt{D}, 20\sqrt{D}, 100\sqrt{D}\}$. The hyperparameters are the same as Sect. 5.1 except GPPL uses $\alpha_0 = 2$, $\beta_0 = 200$ and crowdGPPL uses $\alpha_0^{(t)} = \alpha_0^{(v)} = 2$, $\beta_0^{(t)} = \beta_0^{(t)} = 200$, $\alpha_0^{(w)} = 1$, $\beta_0^{(w)} = 10$. We do not optimise α_0, but choose β_0 by comparing training set accuracy for GPPL with $\beta_0 \in \{2, 200, 20000\}$. The best value of β_0 is

Table 3 UKPConvArgCrowdSample: predicting consensus, personal preferences for all workers, and personal preferences for workers with >50 pairs in the training set

Method	Consensus			Personal: all workers			>50 training pairs		
	Acc	CEE	τ	Acc	CEE	τ	Acc	CEE	τ
GPPL	.77	**.51**	.50	.71	**.56**	.31	.72	**.55**	.25
crowdGPPL	**.79**	.52	**.53**	**.72**	.58	**.33**	**.74**	**.55**	**.27**
crowdGPPL$\backslash \boldsymbol{t}$	–	–	–	.68	.63	.23	**.74**	.57	**.27**
crowdBT-GP	.75	.53	.45	.69	.58	.30	.71	.56	.23

Bold values indicate the best performance in each column

also used for $\beta_0^{(t)}$ and $\beta_0^{(v)}$, then training set accuracy of crowdGPPL is used to select $\beta_0^{(w)} \in \{1, 10, 100\}$. We set $C = 50$, $M = 500$, $P_i = 200$, $\epsilon = 10$, and $\rho = 0.9$ without optimisation.

Table 3 shows that crowdGPPL outperforms both GPPL and crowdBT–GP at predicting both the consensus and personal preferences (significant for Kendall's τ with $p < 0.05$, Wilcoxon signed-rank test), suggesting that there is a benefit to modelling individual workers in subjective, crowdsourced tasks. We also compare against crowdGPPL without the consensus (crowdGPPL$\backslash \boldsymbol{t}$) and find that including $\boldsymbol{t}$ in the model improves personalised predictions. This is likely because many workers have few training pairs, so the consensus helps to identify arguments that are commonly considered very poor or very convincing. Table 3 also shows that for workers with more than 50 pairs in the training set, accuracy and CEE improve for all methods but τ decreases, suggesting that some items may be ranked further away from their correct ranking for these workers. It is possible that workers who were willing to complete more annotations (on average 31 per fold) deviate further from the consensus, and crowdGPPL does not fully capture their preferences given the data available.

We examine the scalability of our SVI method by evaluating GPPL and crowd-GPPL with different numbers of inducing points, M, and different mini-batch sizes, P_i. Figure 2a shows the trade-off between runtime and training set accuracy as an effect of choosing M. Accuracy levels off as M increases, while runtime continues to increase rapidly in a polynomial fashion. Using inducing points can therefore give a large improvement in runtimes with a fairly small performance hit. Figure 2b demonstrates that smaller batch sizes do not negatively affect the accuracy, although they increase runtimes as more iterations are required for convergence. The runtimes flatten out as P_i increases, so we recommend choosing $P_i \geq 200$ but small enough to complete an iteration rapidly with the computational resources available. Figure 2c, d show runtimes as a function of the number of items in the training set, N, and the number of pairwise training labels, P, respectively (all other settings remain as in Fig. 2a). In both cases, the increases to runtime are small, despite the growing dataset size.

5.4 Posterior variance of item components

We investigate how many latent components were actively used by crowdGPPL on the *UKPConvArgCrowdSample* and *Sushi-A* datasets. Figure 3 plots the posterior expectations of the inferred scales, $1/\left(s_c^{(v)} s_c^{(w)}\right)$, for the latent item components. The plots show that many factors have a relatively small variance and therefore do not contribute to many of

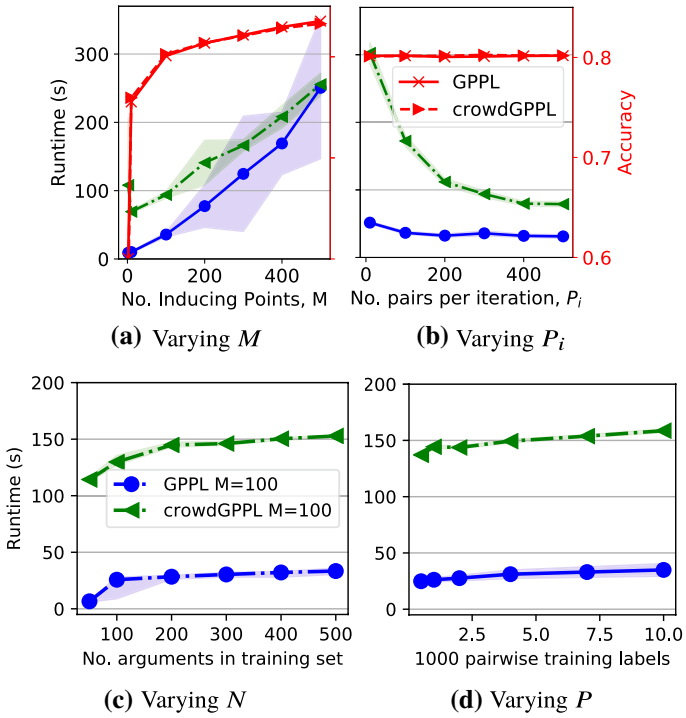


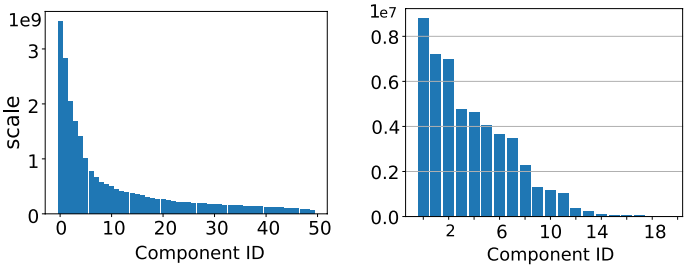
Fig. 2 Wall-clock times for training+prediction of consensus utilities for arguments in the training folds of UKPConvArgCrowdSample. CrowdGPPL was run with $C = 5$. In **b**, **c** and **d**, $M = 100$. Lines show means over 32 runs, bands indicate 1 standard deviation (mostly very little variation between folds)

the model's predictions. This indicates that our Bayesian approach will only make use of components that are supported by the data, even if C is larger than required.

6 Conclusions

We proposed a novel Bayesian preference learning approach for modelling both the preferences of individuals and the overall consensus of a crowd. Our model learns the latent utilities of items from pairwise comparisons using a combination of Gaussian processes and Bayesian matrix factorisation to capture differences in opinion. We introduce a stochastic variational inference (SVI) method, that, unlike previous work, can scale to arbitrarily large datasets, since its time and memory complexity do not grow with the dataset size. Our experiments confirm the method's scalability and show that jointly modelling the consensus and personal preferences can improve predictions of both. Our approach performs competitively against less scalable alternatives and improves on the previous state of the art for predicting argument convincingness from crowdsourced data (Simpson and Gurevych 2018).

Future work will investigate learning inducing point locations and optimising length-scale hyperparameters by maximising the variational lower bound, $\mathcal{L}$, as part of the



(a) *UKPConvArgCrowdSample* **(b)** *Sushi-A*

Fig. 3 Latent component variances, $1/\left(s_c^{(v)} s_c^{(w)}\right)$ in crowdGPPL, means over all runs

variational inference method. Another important direction will be to generalise the likelihood from pairwise comparisons to comparisons involving more than two items (Pan et al. 2018) or best–worst scaling (Kiritchenko and Mohammad 2017) to provide scalable Bayesian methods for other forms of comparative preference data.

Acknowledgements Open Access funding provided by Projekt DEAL. This work was supported by the German Federal Ministry of Education and Research (BMBF) under promotional references 01UG1416B (CEDIFOR), by the German Research Foundation through the the German-Israeli Project Cooperation (DIP, Grant DA1600/1-1 and Grant GU 798/17-1), and by the German Research Foundation EVIDENCE Project (Grant GU 798/27-1). We would like to thank the journal editors and reviewers for their valuable feedback.

Appendix 1: Variational lower bound for GPPL

Due to the non-Gaussian likelihood, Eq. 2, the posterior distribution over $\boldsymbol{f}$ contains intractable integrals:

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \alpha_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f};\boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s;\alpha_0, \beta_0)ds}{\int\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}';\boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s;\alpha_0, \beta_0)dsdf'}. \tag{27}$$

We can derive a variational lower bound as follows, beginning with an approximation that does not use inducing points:

$$\begin{aligned}\mathcal{L} = \sum_{p=1}^{P} \mathbb{E}_{q(f)}\left[\ln p\left(y_p|f(\boldsymbol{x}_{a_p}), f(\boldsymbol{x}_{b_p})\right)\right] + \mathbb{E}_{q(\boldsymbol{f}),q(s)}\left[\ln \frac{p\left(\boldsymbol{f}|\boldsymbol{0}, \frac{\boldsymbol{K}}{s}\right)}{q(\boldsymbol{f})}\right] \\ + \mathbb{E}_{q(s)}\left[\ln \frac{p\left(s|\alpha_0, \beta_0\right)}{q(s)}\right]\end{aligned} \tag{28}$$

Writing out the expectations in terms of the variational parameters, we get:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f})}\left[\sum_{p=1}^{P} y_p \ln \Phi(z_p) + (1-y_p)\left(1 - \ln \Phi(z_p)\right)\right] + \mathbb{E}_{q(\boldsymbol{f})}\left[\ln \mathcal{N}\left(\hat{\boldsymbol{f}};\boldsymbol{\mu}, \boldsymbol{K}/\mathbb{E}[s]\right)\right] \\ &\quad - \mathbb{E}_{q\boldsymbol{f}}\left[\ln \mathcal{N}\left(\boldsymbol{f};\hat{\boldsymbol{f}}, \boldsymbol{C}\right)\right] + \mathbb{E}_{q(s)}\left[\ln \mathcal{G}\left(s;\alpha_0, \beta_0\right) - \ln \mathcal{G}(s;\alpha, \beta)\right] \\ &= \sum_{p=1}^{P} y_p \mathbb{E}_{q(\boldsymbol{f})}[\ln \Phi(z_p)] + (1-y_p)\left(1 - \mathbb{E}_{q(\boldsymbol{f})}[\ln \Phi(z_p)]\right)\Bigg] \\ &\quad - \frac{1}{2}\left\{\ln |\boldsymbol{K}| - \mathbb{E}[\ln s] + \mathrm{tr}\left(\left(\hat{\boldsymbol{f}}^T\hat{\boldsymbol{f}} + \boldsymbol{C}\right)\boldsymbol{K}^{-1}\right) - \ln |\boldsymbol{C}| - N\right\} \\ &\quad - \Gamma(\alpha_0) + \alpha_0(\ln \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\ln s] + \Gamma(\alpha) + (\beta - \beta_0)\mathbb{E}[s] - \alpha \ln \beta.\end{aligned} \tag{29}$$

The expectation over the likelihood can be computed using numerical integration. Now we can introduce the sparse approximation to obtain the bound in Eq. 16:

$$\begin{aligned}\mathcal{L} &\approx \mathbb{E}_{q(\boldsymbol{f})}[\ln p(\boldsymbol{y}|\boldsymbol{f})] + \mathbb{E}_{q(\boldsymbol{f}_m),q(s)}[\ln p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0)] \\ &\quad - \mathbb{E}_{q(\boldsymbol{f}_m)}[\ln q(\boldsymbol{f}_m)] - \mathbb{E}_{q(s)}[\ln q(s)] \\ &= \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\ln p(y_p|f(\boldsymbol{x}_{a_p}), f(\boldsymbol{x}_{b_p}))] - \frac{1}{2}\Bigg\{\ln |\boldsymbol{K}_{mm}| - \mathbb{E}[\ln s] - \ln |\boldsymbol{S}| - M \\ &\quad + \hat{\boldsymbol{f}}_m^T\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\Bigg\} + \ln \Gamma(\alpha) - \ln \Gamma(\alpha_0) + \alpha_0(\ln \beta_0) \\ &\quad + (\alpha_0 - \alpha)\mathbb{E}[\ln s] + (\beta - \beta_0)\mathbb{E}[s] - \alpha \ln \beta,\end{aligned} \tag{30}$$

where the terms relating to $\mathbb{E}\left[p(\boldsymbol{f}|\boldsymbol{f}_m) - q(\boldsymbol{f})\right]$ cancel.

Appendix 2: Variational lower bound for crowdGPPL

For crowdGPPL, our approximate variational lower bound is:

$$
\begin{aligned}
\mathcal{L}_{cr} = & \sum_{p=1}^{P} \ln p(y_p | \hat{\boldsymbol{v}}_{.,a_p}^T \hat{\boldsymbol{w}}_{.,j_p} + \hat{t}_{a_p}, \hat{\boldsymbol{v}}_{.,b_p}^T \hat{\boldsymbol{w}}_{.,j_p} + \hat{t}_{b_p}) \\
& - \frac{1}{2} \Bigg\{ \sum_{c=1}^{C} \Bigg\{ \ln |\boldsymbol{K}_{mm}| - \mathbb{E}\left[\ln s_c^{(v)}\right] - \ln |\boldsymbol{S}_c^{(v)}| \\
& - M_{\text{items}} + \hat{\boldsymbol{v}}_{m,c}^T \mathbb{E}\left[s_c^{(v)}\right] \boldsymbol{K}_{mm}^{-1} \hat{\boldsymbol{v}}_{m,c} + \text{tr}\left(\mathbb{E}\left[s_c^{(v)}\right] \boldsymbol{K}_{mm}^{-1} \boldsymbol{S}_{v,c}\right) + \ln |\boldsymbol{L}_{mm}| - \mathbb{E}\left[\ln s_c^{(w)}\right] \\
& - \ln |\boldsymbol{\Sigma}_c| - M_{\text{users}} + \hat{\boldsymbol{w}}_{m,c}^T \mathbb{E}\left[s_c^{(w)}\right] \boldsymbol{L}_{mm}^{-1} \hat{\boldsymbol{w}}_{m,c} + \text{tr}\left(\mathbb{E}\left[s_c^{(w)}\right] \boldsymbol{L}_{mm}^{-1} \boldsymbol{\Sigma}_c\right) + \ln |\boldsymbol{K}_{mm}| \Bigg\} \\
& - \mathbb{E}\left[\ln s^{(t)}\right] - \ln |\boldsymbol{S}^{(t)}| - M_{\text{items}} + \hat{\boldsymbol{t}}^T \mathbb{E}\left[s^{(t)}\right] \boldsymbol{K}_{mm}^{-1} \hat{\boldsymbol{t}} + \text{tr}\left(\mathbb{E}\left[s^{(t)}\right] \boldsymbol{K}_{mm}^{-1} \boldsymbol{S}^{(t)}\right) \Bigg\} \\
& + \sum_{c=1}^{C} \Bigg\{ \ln \Gamma\left(\alpha_0^{(v)}\right) + \alpha_0^{(v)} \left(\ln \beta_0^{(v)}\right) + \ln \Gamma\left(\alpha_c^{(v)}\right) + \left(\alpha_0^{(v)} - \alpha_c^{(v)}\right) \mathbb{E}\left[\ln s_c^{(v)}\right] \\
& + \left(\beta_c^{(v)} - \beta_0^{(v)}\right) \mathbb{E}[s_c^{(v)}] - \alpha_c^{(v)} \ln \beta_c^{(v)} + \ln \Gamma\left(\alpha_0^{(w)}\right) + \alpha_0^{(w)} \left(\ln \beta_0^{(w)}\right) + \ln \Gamma\left(\alpha_c^{(w)}\right) \\
& + \left(\alpha_0^{(w)} - \alpha_c^{(w)}\right) \mathbb{E}\left[\ln s_c^{(w)}\right] + \left(\beta_c^{(w)} - \beta_0^{(w)}\right) \mathbb{E}[s_c^{(w)}] - \alpha_c^{(w)} \ln \beta_c^{(w)} \Bigg\} + \ln \Gamma\left(\alpha_0^{(t)}\right) \\
& + \alpha_0^{(t)} \left(\ln \beta_0^{(t)}\right) + \ln \Gamma\left(\alpha^{(t)}\right) + \left(\alpha_0^{(t)} - \alpha^{(t)}\right) \mathbb{E}\left[\ln s^{(t)}\right] + \left(\beta^{(t)} - \beta_0^{(t)}\right) \mathbb{E}\left[s^{(t)}\right] \\
& - \alpha^{(t)} \ln \beta^{(t)}.
\end{aligned}
\tag{31}
$$

Appendix 3: Posterior parameters for variational factors in crowdGPPL

For the latent item components, the posterior precision estimate for $\boldsymbol{S}_{v,c}^{-1}$ at iteration i is given by:

$$
\begin{aligned}
\left(\boldsymbol{S}_{c,i}^{(v)}\right)^{-1} = & (1 - \rho_i) \left(\boldsymbol{S}_{c,i-1}^{(v)}\right)^{-1} + \rho_i \boldsymbol{K}_{mm}^{-1} \mathbb{E}\left[s_c^{(v)}\right] \\
& + \rho_i \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \text{diag}\left(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}}\right) \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i,
\end{aligned}
\tag{32}
$$

where $\boldsymbol{A}_i = \boldsymbol{K}_{im}\boldsymbol{K}_{mm}^{-1}$, $\hat{\boldsymbol{w}}_c$ and $\boldsymbol{\Sigma}_c$ are the variational mean and covariance of the cth latent user component (defined below in Eqs. 37 and 36), and $\boldsymbol{u} = \{u_p \forall p \in \boldsymbol{P}_i\}$ is the vector of user indexes in the sample of observations. We use $\boldsymbol{S}_{v,c}^{-1}$ to compute the means for each row of $\boldsymbol{V}_m$:

$$\hat{\boldsymbol{v}}_{m,c,i} = \boldsymbol{S}_{c,i}^{(v)}\Bigg((1-\rho_i)\Big(\boldsymbol{S}_{c,i-1}^{(v)}\Big)^{-1}\hat{\boldsymbol{v}}_{m,c,i-1} + \rho_i\pi_i\boldsymbol{S}_{c,i}^{(v)}\boldsymbol{A}_i^T\boldsymbol{G}_i^T\mathrm{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}})\boldsymbol{Q}_i^{-1}\Big(\boldsymbol{y}_i - \Phi(\hat{\boldsymbol{z}}_i) + \mathrm{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}})\boldsymbol{G}_i\boldsymbol{A}_i\hat{\boldsymbol{v}}_{c,m,i-1}^T\Big)\Bigg). \tag{33}$$

For the consensus, the precision and mean are updated according to the following:

$$\Big(\boldsymbol{S}_i^{(t)}\Big)^{-1} = (1-\rho_i)\Big(\boldsymbol{S}_{i-1}^{(t)}\Big) + \rho_i\boldsymbol{K}_{mm}^{-1}\mathbb{E}\big[s^{(t)}\big] + \rho_i\pi_i\boldsymbol{A}_i^T\boldsymbol{G}_i^T\boldsymbol{Q}_i^{-1}\boldsymbol{G}_i\boldsymbol{A}_i \tag{34}$$

$$\hat{\boldsymbol{t}}_{m,i} = \boldsymbol{S}_i^{(t)}\Bigg((1-\rho_i)\Big(\boldsymbol{S}_{i-1}^{(t)}\Big)^{-1}\hat{\boldsymbol{t}}_{m,i-1} + \rho_i\pi_i\boldsymbol{A}_i^T\boldsymbol{G}_i^T\boldsymbol{Q}_i^{-1}\big(\boldsymbol{y}_i - \Phi(\hat{\boldsymbol{z}}_i) + \boldsymbol{G}_i\boldsymbol{A}_i\hat{\boldsymbol{t}}_i\big)\Bigg). \tag{35}$$

For the latent user components, the SVI updates for the parameters are:

$$\boldsymbol{\Sigma}_{c,i}^{-1} = (1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}^{-1} + \rho_i\boldsymbol{L}_{mm}^{-1}\mathbb{E}\big[s_c^{(w)}\big] + \rho_i\pi_i\boldsymbol{A}_{w,i}^T \Bigg(\boldsymbol{H}_i^T\mathrm{diag}\Big(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 + \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}}^{(v)} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}}^{(v)} - 2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}\hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}}^{(v)}\Big)\boldsymbol{Q}_i^{-1}\boldsymbol{H}_i\Bigg)\boldsymbol{A}_{w,i} \tag{36}$$

$$\hat{\boldsymbol{w}}_{m,c,i} = \boldsymbol{\Sigma}_{c,i}\Bigg((1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}\hat{\boldsymbol{w}}_{m,c,i-1} + \rho_i\pi_i\boldsymbol{A}_{w,i}^T\boldsymbol{H}_i^T\mathrm{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}} - \hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\boldsymbol{Q}_i^{-1} \Big(\boldsymbol{y}_i - \Phi(\hat{\boldsymbol{z}}_i) + \mathrm{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}} - \hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\boldsymbol{H}_u^{(i)}\hat{\boldsymbol{w}}_{c,m,i-1}^T\Big)\Bigg), \tag{37}$$

where the subscripts $\boldsymbol{a} = \{a_p \forall p \in P_i\}$ and $\boldsymbol{b} = \{b_p \forall p \in P_i\}$ are lists of indices to the first and second items in the pairs, respectively, $\boldsymbol{A}_{w,i} = \boldsymbol{L}_{im}\boldsymbol{L}_{mm}^{-1}$, and $\boldsymbol{H}_i \in U_i \times P_i$ contains partial derivatives of the likelihood corresponding to each user (U_i is the number of users referred to by pairs in $\boldsymbol{P}_i$), with elements given by:

$$H_{p,j} = \Phi(\mathbb{E}[z_p])(1-\Phi(\mathbb{E}[z_p]))(2y_p - 1)[j = u_p]. \tag{38}$$

Input: Pairwise labels, $\boldsymbol{y}$, training item features, $\boldsymbol{x}$, training user features $\boldsymbol{u}$, test item features $\boldsymbol{x}^*$, test user features $\boldsymbol{u}^*$

Compute kernel matrices $\boldsymbol{K}$, $\boldsymbol{K}_{mm}$ and $\boldsymbol{K}_{nm}$ given $\boldsymbol{x}$;
Compute kernel matrices $\boldsymbol{L}$, $\boldsymbol{L}_{mm}$ and $\boldsymbol{L}_{nm}$ given $\boldsymbol{u}$;
Initialise $\mathbb{E}\left[s^{(t)}\right]$, $\mathbb{E}\left[s_c^{(v)}\right]\ \forall c$, $\mathbb{E}\left[s_c^{(w)}\right]\ \forall c$, $\mathbb{E}[\boldsymbol{V}]$, $\hat{\boldsymbol{V}}_m$, $\mathbb{E}[\boldsymbol{W}]$, $\hat{\boldsymbol{W}}_m$, $\mathbb{E}[\boldsymbol{t}]$, $\hat{\boldsymbol{t}}_m$ to prior means;
Initialise $\boldsymbol{S}_{v,c}\forall c$ and $\boldsymbol{S}_t$ to prior covariance $\boldsymbol{K}_{mm}$;
Initialise $\boldsymbol{S}_{w,c}\forall c$ to prior covariance $\boldsymbol{L}_{mm}$;
while $\mathcal{L}$ *not converged* **do**
  Select random sample, $\boldsymbol{P}_i$, of P observations;
  while $\boldsymbol{G}_i$ *not converged* **do**
    Compute $\boldsymbol{G}_i$ given $\mathbb{E}[\boldsymbol{F}_i]$;
    Compute $\hat{\boldsymbol{t}}_{m,i}$ and $\boldsymbol{S}_i^{(t)}$;
    for *c in 1,...,C* **do**
      Update $\mathbb{E}[\boldsymbol{F}_i]$;
      Compute $\hat{\boldsymbol{v}}_{m,c,i}$ and $\boldsymbol{S}_{i,c}^{(v)}$;
      Update $q\left(s_c^{(v)}\right)$, compute $\mathbb{E}\left[s_c^{(v)}\right]$ and $\mathbb{E}\left[\ln s_c^{(v)}\right]$;
      Update $\mathbb{E}[\boldsymbol{F}_i]$;
      Compute $\hat{\boldsymbol{W}}_{m,c,i}$ and $\boldsymbol{\Sigma}_{i,c}$;
      Update $q\left(s_c^{(w)}\right)$, compute $\mathbb{E}\left[s_c^{(w)}\right]$ and $\mathbb{E}\left[\ln s_c^{(w)}\right]$;
    end
    Update $\mathbb{E}[\boldsymbol{F}_i]$;
  end
  Update $q\left(s^{(t)}\right)$, compute $\mathbb{E}\left[s^{(t)}\right]$ and $\mathbb{E}\left[\ln s^{(t)}\right]$;
end
Compute kernel matrices for test items, $\boldsymbol{K}_{**}$ and $\boldsymbol{K}_{*m}$, given $\boldsymbol{x}^*$;
Compute kernel matrices for test users, $\boldsymbol{L}_{**}$ and $\boldsymbol{L}_{*m}$, given $\boldsymbol{u}^*$;
Use converged values of $\mathbb{E}[\boldsymbol{F}]$ and $\hat{\boldsymbol{F}}_m$ to estimate posterior over $\boldsymbol{F}^*$ at test points ;
Output: Posterior mean of the test values, $\mathbb{E}[\boldsymbol{F}^*]$ and covariance, $\boldsymbol{C}^*$

Algorithm 2: The SVI algorithm for crowdGPPL.

Appendix 4: Predictions with crowdGPPL

The means, item covariances and user variance required for predictions with crowdGPPL (Eq. 25) are defined as follows:

$$\hat{\boldsymbol{t}}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{t}}_m, \qquad \boldsymbol{C}^{(t)*} = \frac{\boldsymbol{K}_{**}}{\mathbb{E}\left[s^{(t)}\right]} + \boldsymbol{A}_{*m}\left(\boldsymbol{S}^{(t)} - \boldsymbol{K}_{mm}\right)\boldsymbol{A}_{*m}^T, \tag{39}$$

$$\hat{\boldsymbol{v}}_c^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{v}}_{m,c}, \qquad \boldsymbol{C}_c^{(v)*} = \frac{\boldsymbol{K}_{**}}{\mathbb{E}\left[s_c^{(v)}\right]} + \boldsymbol{A}_{*m}\left(\boldsymbol{S}_c^{(v)} - \boldsymbol{K}_{mm}\right)\boldsymbol{A}_{*m}^T \tag{40}$$

$$\hat{\boldsymbol{w}}_c^* = \boldsymbol{L}_{*m}\boldsymbol{L}_{mm}^{-1}\hat{\boldsymbol{w}}_{m,c}, \qquad \omega_{c,u}^* = 1/\mathbb{E}\left[s_c^{(w)}\right] + \boldsymbol{A}_{um}^{(w)}(\boldsymbol{\Sigma}_{w,c} - \boldsymbol{L}_{mm})\boldsymbol{A}_{um}^{(w)T} \tag{41}$$

where $\boldsymbol{A}_{*m} = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}$, $\boldsymbol{A}_{um}^{(w)} = \boldsymbol{L}_{um}\boldsymbol{L}_{mm}^{-1}$ and $\boldsymbol{L}_{um}$ is the covariance between user u and the inducing users.

Appendix 5: Mathematical notation

A list of symbols is provided in Table 4.

Table 4 Table of symbols used to represent variables in this paper

Symbol	Meaning
General symbols used with multiple variables	
$\hat{}$	An expectation over a variable
$\tilde{}$	An approximation to the variable
upper case, bold letter	A matrix
lower case, bold letter	A vector
lower case, normal letter	A function or scalar
$*$	Indicates that the variable refers to the test set, rather than the training set
Pairwise preference labels	
$y(a, b)$	A binary label indicating whether item a is preferred to item b
y_p	The pth pairwise label in a set of observations
$\boldsymbol{y}$	The set of observed values of pairwise labels
Φ	Cumulative density function of the standard Gaussian (normal) distribution
$\boldsymbol{x}_a$	The features of item a (a numerical vector)
$\boldsymbol{X}$	The features of all items in the training set
D	The size of the feature vector
N	Number of items in the training set
P	Number of pairwise labels in the training set
$\boldsymbol{x}^*$	The features of all items in the test set
δ_a	Observation noise in the utility of item a
σ^2	Variance of the observation noise in the utilities
z_p	The difference in utilities of items in pair p, normalised by its total variance
$\boldsymbol{z}$	set of z_p Values for training pairs
GPPL (some terms also appear in crowdGPPL)	
f	Latent utility function over items in single-user GPPL
$\boldsymbol{f}$	Utilities, i.e., values of the latent utility function for a given set of items
$\boldsymbol{C}$	Posterior covariance in $\boldsymbol{f}$; in crowdGPPL, superscripts indicate whether this is the covariance of consensus values or latent item components
s	An inverse function scale; in crowdGPPL, superscripts indicate which function this variable scales
k	Kernel function
θ	Kernel hyperparameters for the items
$\boldsymbol{K}$	Prior covariance matrix over items
α_0	Shape hyperparameter of the inverse function scale prior
β_0	Scale hyperparameters of the inverse function scale prior

Table 4 (continued)

Symbol	Meaning
CrowdGPPL	
$\boldsymbol{F}$	Matrix of utilities, where rows correspond to items and columns to users
$\boldsymbol{t}$	Consensus utilities
C	Number of latent components
c	Index of a component
$\boldsymbol{V}$	Matrix of latent item components, where rows correspond to components
$\boldsymbol{v}_c$	A row of $\boldsymbol{V}$ for the cth component
$\boldsymbol{W}$	Matrix of latent user components, where rows correspond to components
$\boldsymbol{w}_c$	A row of $\boldsymbol{W}$ for the cth component
$\boldsymbol{\omega}_c$	Posterior variance for the cth user component
η	Kernel hyperparameters for the users
$\boldsymbol{L}$	Prior covariance matrix over users
$\boldsymbol{u}_j$	User features for user j
U	Number of users in the training set
$\boldsymbol{U}$	Matrix of features for all users in the training set
Probability distributions	
$\mathcal{N}$	(multivariate) Gaussian or normal distribution
$\mathcal{G}$	Gamma distribution
Stochastic variational inference (SVI)	
M	Number of inducing items
$\boldsymbol{Q}$	Estimated observation noise variance for the approximate posterior
γ, λ	Estimated hyperparameters of a Beta prior distribution over $\Phi(z_p)$
i	Iteration counter for stochastic variational inference
$\boldsymbol{f}_m$	Utilities of inducing items
$\boldsymbol{K}_{mm}$	Prior covariance of the inducing items
$\boldsymbol{K}_{nm}$	Prior covariance between training and inducing items
$\boldsymbol{S}$	Posterior covariance of the inducing items; in crowdGPPL, a superscript and subscript indicate which variable this is the posterior covariance for
$\boldsymbol{\Sigma}$	Posterior covariance over the latent user components
$\boldsymbol{A}$	$\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}$
$\boldsymbol{G}$	Linearisation term used to approximate the likelihood
a	Posterior shape parameter for the Gamma distribution over s
b	Posterior scale parameter for the Gamma distribution over s
ρ_i	A mixing coefficient, i.e., a weight given to the ith update when combining with current values of variational parameters
ϵ	Delay
r	Forgetting rate

Table 4 (continued)

Symbol	Meaning
π_i	Weight given to the update at the ith iteration
$\boldsymbol{P}_i$	Subset of pairwise labels used in the ith iteration
P_i	Number of pairwise labels in the ith iteration subsample
U_i	Number of users referred to in the ith subsample
$\boldsymbol{u}$	Users in the ith subsample
$\boldsymbol{a}$	Indexes of first items in the pairs in the ith subsample
$\boldsymbol{b}$	Indexes of first items in the pairs in the ith subsample

References

Abbasnejad, E., Sanner, S., Bonilla, E. V., & Poupart, P., et al. (2013). Learning community-based preferences via dirichlet process mixtures of Gaussian processes. In *Twenty-third international joint conference on artificial intelligence* (pp. 1213–1219). Retrieved January 17, 2020 from https://www.ijcai.org/Proceedings/13/Papers/183.pdf.

Adams, R. P., Dahl, G. E., & Murray, I. (2010). Incorporating side information in probabilistic matrix factorization with Gaussian processes. In *Proceedings of the twenty-sixth conference on uncertainty in artificial intelligence* (pp. 1–9). AUAI Press.

Ahn, S., Korattikara, A., Liu, N., Rajan, S., & Welling, M. (2015). Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 9–18). ACM.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Banerji, M., Lahav, O., Lintott, C. J., Abdalla, F. B., Schawinski, K., Bamford, S. P., et al. (2010). Galaxy zoo: Reproducing galaxy morphologies via machine learning. *Monthly Notices of the Royal Astronomical Society*, *406*(1), 342–353.

Bonilla, E., Steinberg, D., & Reid, A. (2016). Extended and unscented kitchen sinks. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning, PMLR, New York, New York, USA, proceedings of machine learning research* (Vol. 48, pp. 1651–1659). Retrieved January 17, 2020 from http://proceedings.mlr.press/v48/bonilla16.html.

Bors, A. G., & Pitas, I. (1996). Median radial basis function neural network. *IEEE Transactions on Neural Networks*, *7*(6), 1351–1364.

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, *39*(3/4), 324–345.

Chen, X., Bennett, P. N., Collins-Thompson, K., & Horvitz, E. (2013). Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on web search and data mining* (pp. 193–202). ACM.

Chen, G., Zhu, F., & Heng, P. A. (2018). Large-scale Bayesian probabilistic matrix factorization with memo-free distributed variational inference. *ACM Transactions on Knowledge Discovery from Data*, *12*(3), 31:1–31:24.

Chu, W., & Ghahramani, Z. (2005). Preference learning with Gaussian processes. In *Proceedings of the 22nd international conference on machine learning* (pp. 137–144). ACM.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies (long and short papers), association for computational linguistics, Minneapolis, Minnesota* (Vol. 1, pp. 4171–4186). https://doi.org/10.18653/v1/N19-1423.

Felt, P., Ringger, E., & Seppi, K. (2016). Semantic annotation aggregation with conditional crowdsourcing models and word embeddings. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 1787–1796). Osaka, Japan: The COLING 2016 Organizing Committee.

Fu, Y., Hospedales, T. M., Xiang, T., Xiong, J., Gong, S., Wang, Y., et al. (2016). Robust subjective visual property prediction from crowdsourced pairwise labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(3), 563–577.

Fürnkranz, J., & Hüllermeier, E. (2010). Preference learning and ranking by pairwise comparison. In *Preference learning* (pp. 65–82). Springer.

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., & Sriperumbudur, B. K. (2012). Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems* (pp. 1205–1213). Retrieved January 17, 2020 from https://papers.nips.cc/paper/4727-optimal-kernel-choice-for-large-scale-two-sample-tests.

Guo, S., Sanner, S., & Bonilla, E. V. (2010). Gaussian process preference elicitation. In *Advances in neural information processing systems* (pp. 262–270). Retrieved January 17, 2020 from https://papers.nips.cc/paper/4141-gaussian-process-preference-elicitation.

Habernal, I., & Gurevych, I. (2016). Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional LSTM. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Vol. 1: long papers, pp. 1589–1599). Berlin, Germany: Association for Computational Linguistics.

Han, B., Pan, Y., & Tsang, I. W. (2018). Robust Plackett-Luce model for k-ary crowdsourced preferences. *Machine Learning*, *107*(4), 675–702.

Haykin, S. (2001). *Kalman filtering and neural networks*. Wiley.

Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the twenty-ninth conference on uncertainty in artificial intelligence* (pp. 282–290). AUAI Press.

Hensman, J., Matthews, A. G. D. G., & Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *Proceedings of the 18th international conference on artificial intelligence and statistics* (pp. 351–360). Retrieved January 17, 2020 from http://proceedings.mlr.press/v38/hensman15.

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. W. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, *14*(1), 1303–1347.

Houlsby, N., Huszar, F., Ghahramani, Z., & Hernández-Lobato, J. M. (2012). Collaborative Gaussian processes for preference learning. In *Advances in neural information processing systems* (pp. 2096–2104). Retrieved January 17, 2020 from http://papers.nips.cc/paper/4700-collaborative-gaussian-processes-for-preference-learning.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 133–142). ACM.

Kamishima, T. (2003). Nantonac collaborative filtering: Recommendation based on order responses. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 583–588). ACM.

Kendall, M. G. (1948). *Rank correlation methods*. New York City: Griffin.

Khan, M. E., Ko, Y. J., & Seeger, M. (2014). Scalable collaborative Bayesian preference learning. In: S. Kaski & J. Corander (Eds.), *Proceedings of the seventeenth international conference on artificial intelligence and statistics, PMLR, Reykjavik, Iceland, proceedings of machine learning research* (Vol. 33, pp. 475–483). Retrieved January 17, 2020 from http://proceedings.mlr.press/v33/khan14.

Kim, Y., Kim, W., & Shim, K. (2014). Latent ranking analysis using pairwise comparisons. In *2014 IEEE international conference on data mining (ICDM), IEEE* (pp. 869–874). Retrieved January 17, 2020 from https://ieeexplore.ieee.org/abstract/document/7023415.

Kingsley, D. C., & Brown, T. C. (2010). Preference uncertainty, preference refinement and paired comparison experiments. *Land Economics*, *86*(3), 530–544.

Kiritchenko, S., & Mohammad, S. (2017). Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th annual meeting of the association for computational linguistics* (Vol. 2: short papers, pp. 465–470). Vancouver, Canada: Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-2074.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Lawrence, N. D., & Urtasun, R. (2009). Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th international conference on machine learning* (pp. 601–608). ACM.

Lee, S. M., & Roberts, S. J. (2010). Sequential dynamic classification using latent variable models. *The Computer Journal*, *53*(9), 1415–1429.

Li, J., Mantiuk, R., Wang, J., Ling, S., & Le Callet, P. (2018). Hybrid-MST: A hybrid active sampling strategy for pairwise preference aggregation. In *Advances in neural information processing systems* (pp. 3475–3485). Retrieved January 17, 2020 from https://papers.nips.cc/paper/7607-hybrid-mst-a-hybrid-active-sampling-strategy-for-pairwise-preference-aggregation.

Lowne, D., Roberts, S. J., & Garnett, R. (2010). Sequential non-stationary dynamic classification with sparse feedback. *Pattern Recognition*, *43*(3), 897–905.

Luce, R. D. (1959). On the possible psychophysical laws. *Psychological Review*, *66*(2), 81.

Lukin, S., Anand, P., Walker, M., & Whittaker, S. (2017). Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics* (pp. 742–753).

MacKay, D. J. (1995). Probable networks and plausible prediction–a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, *6*(3), 469–505.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119). Retrieved January 17, 2020 from https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence* (pp. 362–369). arXiv:1301.2294.

Mo, K., Zhong, E., & Yang, Q. (2013). Cross-task crowdsourcing. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 677–685). ACM.

Mosteller, F. (1951). Remarks on the method of paired comparisons: I. The least squares solution assuming equal standard deviations and equal correlations. *Psychometrika*, *16*, 3–9.

Naish-Guzman, A., & Holden, S. (2008). The generalized FITC approximation. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems 20* (pp. 1057–1064). Curran Associates, Inc.

Nguyen, T. V., & Bonilla, E. V. (2014). Collaborative multi-output Gaussian processes. In *Proceedings of the thirtieth conference on uncertainty in artificial intelligence* (pp. 643–652). AUAI Press.

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

Ovadia, S. (2004). Ratings and rankings: Reconsidering the structure of values and their measurement. *International Journal of Social Research Methodology*, *7*(5), 403–414.

Pan, Y., Han, B., & Tsang, I. W. (2018). Stagewise learning for noisy k-ary preferences. *Machine Learning*, *107*(8–10), 1333–1361.

Plackett, R. L. (1975). The analysis of permutations. *Applied Statistics*, *24*, 193–202.

Porteous, I., Asuncion, A., & Welling, M. (2010). Bayesian matrix factorization with side information and Dirichlet process mixtures. In *Proceedings of the twenty-fourth AAAI conference on artificial intelligence* (pp. 563–568). AAAI Press.

Psorakis, I., Roberts, S., Ebden, M., & Sheldon, B. (2011). Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, *83*(6), 066114.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning* (Vol. 38, pp. 715–719). Cambridge: The MIT Press.

Reece, S., Roberts, S., Nicholson, D., & Lloyd, C. (2011). Determining intent using hard/soft data and Gaussian process classifiers. In *Proceedings of the 14th international conference on information fusion* (pp. 1–8). IEEE.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Saha, A., Misra, R., & Ravindran, B. (2015). Scalable Bayesian matrix factorization. In *Proceedings of the 6th international conference on mining ubiquitous and social environments* (Vol. 1521, pp. 43–54). Retrieved January 17, 2020 from http://ceur-ws.org/Vol-1521/paper6.pdf.

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on machine learning* (pp. 880–887). ACM.

Salimans, T., Paquet, U., & Graepel, T. (2012). Collaborative learning of preference rankings. In *Proceedings of the sixth ACM conference on recommender systems* (pp. 261–264). ACM.

Simpson, E., Reece, S., & Roberts, S. J. (2017). Bayesian heatmaps: Probabilistic classification with multiple unreliable information sources. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 109–125). Springer.

Simpson, E., & Gurevych, I. (2018). Finding convincing arguments using scalable Bayesian preference learning. *Transactions of the Association for Computational Linguistics*, *6*, 357–371. https://doi.org/10.1162/tacl_a_00026.

Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems* (pp. 1257–1264). Retrieved January 17, 2020 from https://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-inputs.

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. (2008). Cheap and fast— but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 conference on empirical*

methods in natural language processing (pp. 254–263). Honolulu, Hawaii: Association for Computational Linguistics.

Steinberg, D. M., & Bonilla, E. V. (2014). Extended and unscented Gaussian processes. In *Advances in neural information processing systems* (pp. 1251–1259). Retrieved January 17, 2020 from https://papers.nips.cc/paper/5455-extended-and-unscented-gaussian-processes.

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological Review*, *34*(4), 273.

Uchida, S., Yamamoto, T., Kato, M. P., Ohshima, H., & Tanaka, K. (2017). Entity ranking by learning and inferring pairwise preferences from user reviews. In *Asia information retrieval symposium* (pp. 141–153). Springer.

Vander Aa, T., Chakroun, I., & Haber, T. (2017). Distributed Bayesian probabilistic matrix factorization. *Procedia Computer Science*, *108*, 1030–1039.

Volkovs, M., Yu, G., & Poutanen, T. (2017). Dropoutnet: Addressing cold start in recommender systems. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4957–4966). Curran Associates, Inc.

Wang, X., Wang, J., Jie, L., Zhai, C., & Chang, Y. (2016). Blind men and the elephant: Thurstonian pairwise preference for ranking in crowdsourcing. In *2016 IEEE 16th international conference on data mining (ICDM)* (pp 509–518). IEEE.

Yang, Y. H., & Chen, H. H. (2011). Ranking-based emotion recognition for music organization and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(4), 762–774.

Yannakakis, G. N., & Hallam, J. (2011). Ranking vs. preference: A comparative study of self-reporting. In *International conference on affective computing and intelligent interaction* (pp. 437–446). Springer.

Yi, J., Jin, R., Jain, S., & Jain, A. (2013). Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *First AAAI conference on human computation and crowdsourcing*. Retrieved January 17, 2020 from https://www.aaai.org/ocs/index.php/HCOMP/HCOMP13/paper/view/7536.