Noname manuscript No.
(will be inserted by the editor)

Predicting Preferences for Crowds of Users: a Scalable, Bayesian Approach

Edwin Simpson · Iryna Gurevych
Ubiquitous Knowledge Processing Lab, Dept. of Computer Science, Technische Universität Darmstadt, Germany
E-mail: {simpson,gurevych}@ukp.informatik.tu-darmstadt.de

Received: date

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 if the amount of data for each user is small. These personal biases also make it harder to infer the consensus of the whole crowd when there are few labels per item. We address these challenges by combining matrix factorization with Gaussian processes, using a Bayesian approach to account for uncertainty arising from sparse data and annotation noise. Our method exploiting input features, such as text embeddings, image features or metadata, to predict preferences for new items and users with limited or no training data. As previous methods for Gaussian process preference learning 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 benchmark recommendation task show that our method is competitive with previous approaches despite our scalable inference approximation. We demonstrate the method's scalability empirically on a natural language processing task with thousands of users and items. On this task, our method improves consensus prediction over the state of the art by modelling the preferences of individual members of the crowd. We make our software publicly available for future work [1].

1 Introduction

Preference learning is the task of learning to compare the values of a set of alternatives according to a particular quality (Fürnkranz and Hüllermeier 2010). Here, we focus on ranking items by preference and selecting the preferred item from a pair. For many preference learning tasks, people have divergent opinions on the correct rankings, which means that data from different annotators may

Address(es) of author(s) should be given

[1] `https://github.com/UKPLab/tacl2018-preference-convincing/tree/crowdGPPL`

not agree. 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 recommendations to a specific user (Resnick and Varian 1997). Disagreements also occur when training data is acquired from multiple annotators, for example, using crowdsourcing, and are often mitigated by redundant labelling, which increases costs (Snow et al. 2008; Banerji et al. 2010). Therefore, we require preference learning methods that can account for differences of opinion to solve two tasks: (1) predicting *personal* preferences for members of a crowd and (2) inferring a *consensus* given observations from multiple, disparate users.

As data from a single user is often sparse, recommender systems often predict predict a user's preferences by via *collaborative filtering*, which combines the observed preferences of similar users (Resnick and Varian 1997). A typical approach is to represent observed ratings in a user-item matrix, then apply *matrix factorization* (Koren et al. 2009) to decompose a user-item matrix into two low-dimensional matrices. Users and items with similar observed ratings have similar row vectors in the low-dimensional matrices. By multiplying the low-dimensional representations, we can predict ratings for unseen user-item pairs. However, traditional approaches are not able to extrapolate to new users or items, as they do not exploit the *input features* of items or users that can be extracted from their content or metadata. For example, in text it is common to count occurrences of each unique token or, more recently, to obtain an *embedding* of a word or document that represents its content as a numerical vector (Mikolov et al. 2013; Devlin et al. 2018). Input features enable predictions for a test item or user based on observations of others with similar feature values, and can help to remedy labelling errors when learning from noisy, crowdsourced data (Felt et al. 2016; Simpson et al. 2015). Furthermore, many established methods for matrix factorisation require that training data is provided in the form of numerical ratings.

A drawback of numerical ratings is that each annotator may interpret the values differently and may label inconsistently depending on the order in which they annotate items (Ovadia 2004; Yannakakis and Hallam 2011). 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. Pairwise labelling can also be quicker for annotators than numerical rating, and facilitates the total sorting of items, since it avoids two items having the same value (Kendall 1948; Kingsley and Brown 2010; Yang and Chen 2011), and can be result in more accurate rankings(Kiritchenko and Mohammad 2017). Besides explicit annotations, pairwise labels can be extracted from user behaviour logs, such as when a user selects one item from a list in preference to others (Joachims 2002). We therefore focus on preference learning from pairwise labels, and henceforth refer to the annotators, users or implicit data sources simply as *users*. Likewise, *items* are any type of instance that users may express preferences over, and could be states or actions as well as objects.

Pairwise labels provided by a crowd are often noisy (Chen et al. 2013), as are labels extracted from user logs (Hu et al. 2008). The need to be robust to noise despite limited data per user and to aggregate data from multiple users motivates a Bayesian treatment. Bayesian approaches account for uncertainty in the model, which has been shown to benefit matrix factorization with sparse data (Salakhutdinov and Mnih 2008) and preference learning with noisy pairwise labels (Chen et al. 2013), although this previous work does not model individual preferences or make predictions for new users or items. A powerful Bayesian approach for prediction is the Gaussian process (GP), which models nonlinear functions of input features and has previously been applied to preference learning (Chu and Ghahramani 2005; Houlsby et al. 2012; Khan et al. 2014). However, existing GP-based methods do not scale to very 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 factorization 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 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 an objective gold-standard 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, illustrating the potential of the method for NLP applications.

The next section of the paper provides background on preference learning and discusses related work. We then develop our model for preference learning from crowds in Section 3, followed by our proposed inference method in Section 4. Then, in Section 5, we evaluate our approach empirically, showing its behaviour on synthetic data, its scalability and its predictive performance on real-world datasets.

2 Related Work

2.1 Pairwise Preference Learning

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), or the Thurstone-Mosteller model, also known as *Thurstone case V* (Thurstone 1927; Mosteller 2006). Examples include best-worst scaling Marley and Louviere (2005), which extends the BT model, and TrueSkill (Herbrich et al. 2007), a Thurstone case V model that learns the skill of game players by treating match outcomes as noisy pairwise labels. Recent work on the BT model has analysed bounds on error rates (Chen and Suh 2015), sample complexity (Shah et al. 2015) and developed computationally efficient active learning Li et al. (2018). However, these examples do not consider input features. Another commonly-used ranking method, SVM-rank (Joachims 2002), predicts pairwise labels from input features, but optimises pairwise label prediction directly without a random utility model. *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). Here, we build on GPPL to provide a scalable approach that can model the preferences of crowds of users.

2.2 Annotator Disagreements when Estimating a Consensus

Much of the 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, personal preferences also introduce labelling *bias*, as the differences between users are not random but depend on the items considered. With small numbers of labels, these biases may reduce the accuracy of the estimated consensus. Furthermore, previous aggregation methods for crowdsourced preferences do not consider the item features, so cannot predict the utility of test items with no or very few labels (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 item features into the model.

2.3 Inferring Personal Preferences for Members of a Crowd

A number of methods use matrix factorisation to predict personal preferences for members of a crowd 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 better handle sparse data. In matrix factorisation, each individual's preferences are represented by a mixture of latent functions. Kim et al. (2014) do not apply matrix factorisation, but instead assume that pairwise labels depend on one of several latent rankings. However, these techniques do not account for any input features for prediction.

A number of other methods use Gaussian processes for personal preference prediction, such as by Guo et al. (2010), who propose a joint Gaussian process over the space of users and features. Since this scales cubically in the number of users, Abbasnejad et al. (2013) propose to cluster the users into behavioural groups. However, distinct clusters do not allow for collaborative learning between users with partially overlapping preferences, e.g. two users may both like one genre of music, while having different preferences over other genres. Khan et al.

(2014) instead learn a GP for each user, then add a matrix factorization term that performs collaborative filtering. However, this approach does not model the relationship between input features and the latent factors, unlike Lawrence and Urtasun (2009) who place GP priors over latent item components. Neither of these last two methods give a fully Bayesian treatment to the model, as the the users' latent features are optimised rather than marginalised. An alternative is *Collaborative GP (collabGP)* (Houlsby et al. 2012), which uses a latent factor model, where each latent factor has a Gaussian process prior. This allows the model to take advantage of the input features of both users and items when learning the latent factors. However, unlike the approach we propose here, none of the methods to date jointly model both consensus and personal preferences in a fully-Bayesian manner, which is needed to handle small amounts of data for both users and items and account for personal biases when inferring the consensus. Furthermore, the existing GP-based approaches suffer from scalability issues due to limitations of their inference methods.

2.4 Scalable Approximate Bayesian Inference

Many of the approaches for modelling individual user preferences use matrix factorization to share information between users and items, which benefits from a Bayesian treatment to reduce the effects of overfitting or noisy data (Saha et al. 2015). Recent work on scalable Bayesian matrix factorization focuses on distributing and parallelising inference but are not directly applicable when Gaussian processes are used to integrate input features (Ahn et al. 2015; Saha et al. 2015; Vander Aa et al. 2017; Chen et al. 2018).

Models that combine Gaussian processes with non-Gaussian likelihoods require approximate inference methods that often scale poorly with the amount of training data available. Established methods such as the Laplace approximation and expectation propagation (Rasmussen and Williams 2006) have computational complexity $\mathcal{O}(N^3)$ with N data points and memory complexity $\mathcal{O}(N^2)$. For collaborative GPPL, Houlsby et al. (2012) propose a kernel for pairwise preference learning and use a sparse *generalized fully independent training conditional* (GFITC) approximation (Snelson and Ghahramani 2006) to reduce the computational 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 run in parallel on subsets of the data, since the objective function is a not a sum over data points. The GP over pairs also means that it does not output posteriors for the utilities of individual items, which are useful for ranking, and can only be trained using pairwise labels, even if observations of the utilities are available.

To handle large numbers of pairwise labels, Khan et al. (2014) develop a variational EM algorithm and sub-sample pairwise data rather than learning from the complete training set. An alternative is *stochastic variational inference (SVI)* (Hoffman et al. 2013), which updates an approximation to the posterior using a different random sample of the training data at each iteration. This allows the approximation to make use of all training data over a number of iterations, while limiting training costs per iteration. SVI has been successfully applied to Gaussian process regression (Hensman et al. 2013) and classification (Hensman et al. 2015), further improving scalability over existing sparse approximations. For multi-output GPs, Nguyen and Bonilla (2014) introduce an SVI approach where each output is a weighted combination of shared latent functions plus a latent function specific to that output. They apply their method to capture dependencies between regression tasks, treating the weights for the shared latent functions as hyperparameters. In this paper, we also use shared latent functions to capture dependencies between different users' preferences, but introduce a Bayesian treatment of the weights using a GP over latent user features, and integrate a non-Gaussian likelihood into the SVI framework to enable learning from pairwise labels. An SVI method for preference learning that places a GP over items, rather than pairs, was previously applied by Simpson and Gurevych (2018). However, this method, a variant of GPPL, did not consider the different preferences of individual users. Here, we propose a new SVI method, crowdGPPL, aimed at a different task: jointly modelling personal preferences of users and the crowd consensus using a combination of Gaussian processes and Bayesian matrix factorisation. In Section 4, we provide the full details of the SVI method for both GPPL, as these were not previously published, then extend the scheme to crowdGPPL.

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 the item's 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 as $a \succ b$). We record the outcome of each comparison between a and b as 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. Hence, given knowledge of f, we can predict 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. However, since $0 < p(y(a,b) = 1) < 1$, the model is uncertain about the value of $y(a,b)$, which accommodates labelling errors or inconsistency in a user's choices at different points in time. The uncertainty is lower if the values f_a and f_b are further apart, which reflects greater consistency in a user's choices when their preferences are stronger.

The random utility model is defined by a likelihood function that maps the utilities to $p(y(a,b))$. Two important random utility models for pairwise labels are the Bradley-Terry model (Bradley and Terry 1952; Plackett 1975; Luce 1959), which assumes a logistic likelihood, and the Thurstone-Mosteller model, also known as *Thurstone case V* (Thurstone 1927; Mosteller 2006), which assumes a probit likelihood.

In the Thurstone case V model, noise in the observations is explained by a Gaussian-distributed noise term, $\delta \sim \mathcal{N}(0, \sigma^2)$:

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

Integrating out the unknown values of δ_a and δ_b gives:

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

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. Please note that a full list of symbols is provided for reference in Appendix D.

In practice, $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$ must be inferred from pairwise training labels, $\boldsymbol{y}$, so that we obtain a posterior distribution over their values. If we assume that 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:

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

where $\hat{f}_a$ and $\hat{f}_b$ are the means and $\boldsymbol{C}$ is the 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 or numerical integration, hence we assume this pairwise label likelihood for our proposed preference learning model. This likelihood is also used for Gaussian process preference learning (GPPL) by Chu and Ghahramani (2005), but here we simplify the formulation by assuming that $\sigma^2 = 0.5$, which leads to z having a denominator of $\sqrt{2 \times 0.5} = 1$. Instead, we model varying degrees of noise in the pairwise labels by scaling f itself, as we describe in the next section.

3.1 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. Typical kernel functions include the *squared exponential* and the *Matérn* (Rasmussen and Williams 2006), which both make minimal assumptions and are effective in a wide range of tasks. These functions assign higher covariance to items with similar feature values. We use the kernel function k_θ to compute a covariance matrix K_θ, between a set of N observed items with features $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$.

The model extends the original definition of GPPL (Chu and Ghahramani 2005), by introducing the inverse scale, s. We also assume that s 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 Equation 2.

Given a set of P pairwise labels, $\boldsymbol{y} = \{y_1, ..., y_P\}$, 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:

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

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)\}$ is a vector containing the utilities of the N items referred to by $\boldsymbol{y}$, and $p(y_p|\boldsymbol{f}) = \Phi(z_p)$ is the pairwise likelihood (Equation 2). We henceforth refer to this model simply as *GPPL*.

3.2 Crowd Preference Learning

To predict the individual preferences of users in a crowd, we could assume an independent GPPL model for each user. However, by modelling all users' preferences jointly, we can exploit the correlations between different users' interests to improve predictions when preference data is sparse, and reduce the cost of storing separate models for all users. Groups of users may share common interests over certain subsets of items, for example, in a book recommendation task, some users may share an interest in one particular genre but otherwise prefer different categories of books. Identifying such correlations between users' interests helps to predict the preferences of users for whom we have only observed a small number of preferences. This is the core idea of recommendation techniques such as collaborative filtering (Resnick and Varian 1997) and matrix factorisation (Koren et al. 2009).

As well as predicting individual preferences, we wish to predict the consensus by aggregating preference labels from multiple users. If we do not account for the individual biases of different users when predicting the consensus, these biases may affect the consensus predictions, particularly when data for certain items is only available from a small subset of users. We address this problem by proposing *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 individuals as simply the result of labelling errors, but instead assume that differences result from the users' subjective biases towards particular items.

For crowdGPPL, we represent utilities in a matrix, $\boldsymbol{F} \in \mathbb{R}^{N \times U}$, with N rows corresponding to items and U columns corresponding to users. We assume that $\boldsymbol{F}$ is the product of two low-dimensional matrices plus a vector of consensus utilities, $\boldsymbol{t}$, of the N items, as follows:

$$\boldsymbol{F} = \boldsymbol{V}^T \boldsymbol{W} + \boldsymbol{t}, \qquad (5)$$

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, and C is the number of latent *components*, i.e., the dimension of the latent representations. 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 latent component corresponds to a utility function for certain items, which is shared by a subset of users. For example, in the case of book recommendation, one component could represent the membership of certain books to a particular genre and the interests of certain users in that genre.

For crowdGPPL, we assume that each row of $\boldsymbol{V}$, $\boldsymbol{v}_c = \{v_c(\boldsymbol{x}_1), ..., v_c(\boldsymbol{x}_N)\}$, 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. We also assume that $\boldsymbol{t} = \{t(\boldsymbol{x}_1), ..., t(\boldsymbol{x}_N)\}$ contains evaluations 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 $\boldsymbol{W}$, $\boldsymbol{w}_c = \{w_c(\boldsymbol{u}_1), ..., w_c(\boldsymbol{u}_U)\}$, 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, the utilities in $\boldsymbol{F}$ are evaluations of a joint preference function, f, which is a weighted sum over the latent functions:

$$f(\boldsymbol{x}_a, \boldsymbol{u}_j) = \sum_{c=1}^{C} v_c(\boldsymbol{x}_a) w_c(\boldsymbol{u}_j) + t(\boldsymbol{x}_a), \tag{6}$$

where $\boldsymbol{u}_j$ are the features of user j and $\boldsymbol{x}_a$ are the features of item a.

CrowdGPPL therefore combines latent features of items and users – represented by the latent components – with the utilities of the items according to an underlying consensus across users. Given the consensus utility for item a, $t(\boldsymbol{x}_a)$, the individual preferences of user j then deviate from the consensus according to $\sum_{c=1}^{C} v_c(\boldsymbol{x}_a) w_c(\boldsymbol{u}_j)$. Hence, when inferring the consensus utilities, we can subtract the effect of individual biases. The consensus can also help when inferring personal preferences for new users, new items or new combinations of users and items that are very different to the combinations in the training data, by taking into account any objective quality 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 $\boldsymbol{v}_c$ have a zero mean, which act as *shrinkage* or *ARD priors* that favour values close to zero (MacKay 1995; Psorakis et al. 2011). Those components that are not required to explain the data will have posterior expectations and expected scales $1/s^{(v)}$ and $1/s^{(w)}$ approaching zero. Therefore, due to our choice of prior, it is not necessary to optimise the value of C, providing a sufficiently large number is chosen.

The form of Equation 6 is also used in the *cross-task crowdsourcing* model of Mo et al. (2013), which uses matrix factorisation to model annotator performance in different tasks. In their model, $\boldsymbol{t}$ corresponds to the objective difficulty of a task. However, unlike crowdGPPL, they do not use Gaussian processes to model the factors, nor apply the approach to preference learning. A similar model for preference learning is proposed by Houlsby et al. (2012), but this does not include a consensus, and the values in $\boldsymbol{v}_c$ directly correspond to pairs, rather than individual items, which means we cannot easily infer the ranking function over items. Their model also omits the scale parameters for the GPs that are used to encourage shrinkage when C is larger than required.

We combine the matrix factorization method with the preference likelihood of Equation 2 to obtain a joint preference model for multiple users, *crowdGPPL*:

$$p\left(\boldsymbol{y}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{t}, s_1^{(v)}, .., s_C^{(v)}, s_1^{(w)}, .., s_C^{(w)}, s^{(t)} | k_\theta, \boldsymbol{X}, k_\eta, \boldsymbol{U}, \alpha_0^{(t)}, \beta_0^{(t)}, \alpha_0^{(v)}, \beta_0^{(v)}, \alpha_0^{(w)}, \beta_0^{(w)}\right)$$
$$= \prod_{p=1}^{P} \Phi\left(z_p\right) \mathcal{N}(\boldsymbol{t}; \boldsymbol{0}, \boldsymbol{K}_\theta / s^{(t)}) \mathcal{G}(s^{(t)}; \alpha_0^{(t)}, \beta_0^{(t)}) \prod_{c=1}^{C} \Big\{ \mathcal{N}(\boldsymbol{v}_c; \boldsymbol{0}, \boldsymbol{K}_\theta / s_c^{(v)})$$
$$\mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{L}_\eta / s_c^{(w)}) \mathcal{G}(s_c^{(v)}; \alpha_0^{(v)}, \beta_0^{(v)}) \mathcal{G}(s_c^{(w)}; \alpha_0^{(w)}, \beta_0^{(w)}) \Big\}, \tag{7}$$

where $z_p = \boldsymbol{v}_{.,a_p}^T \boldsymbol{w}_{.,u_p} + t_{a_p} - \boldsymbol{v}_{.,b_p}^T \boldsymbol{w}_{.,u_p} - t_{b_p}$, $s^{(t)}$ is the inverse scale of t, index p refers to a tuple, $\{u_p, a_p, b_p\}$, which identifies the user and a pair of items, $\boldsymbol{U}$ is the set of all feature vectors for all users, and L_η is the prior covariance between user feature vectors computed using the kernel function k_η.

4 Scalable Inference

In the single user case, the goal is to infer the posterior distribution over the utilities of test items, $\boldsymbol{f}^*$, given a set of pairwise training labels, $\boldsymbol{y}$. In the multi-user case, 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 makes exact inference intractable, hence previous work has used the Laplace approximation for the single user case (Chu and Ghahramani 2005) or a combination of expectation propagation (EP) with variational Bayes (VB) for a multi-user model (Houlsby et al. 2012). The Laplace approximation is a maximum a-posteriori (MAP) solution that takes the most probable values of parameters rather than integrating over their distributions, and has been shown to perform poorly for classification (Nickisch and Rasmussen 2008). EP and VB approximate the true posterior with a simpler, factorized distribution that can be learned using an iterative algorithm. For crowdGPPL, the true posterior is multi-modal, since the latent factors can be re-ordered arbitrarily without affecting $\boldsymbol{F}$, causing a *non-identifiability problem.* EP would average these modes and produce uninformative predictions over $\boldsymbol{F}$, so Houlsby et al. (2012) incorporate a VB step that approximates a single mode. A drawback of EP is that convergence is not guaranteed, whereas VB will always converge when distributions are conjugate (Minka 2001).

Exact inference for a Gaussian process 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 set of 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 $\mathcal{O}(NM)$. These points must be selected to give a good approximation using either heuristics or by optimizing their positions to maximize the approximate marginal likelihood.

One such sparse approximation is the *generalized fully independent training conditional* (GFITC) (Naish-guzman and Holden 2008), used by Houlsby et al. (2012) for the collaborative GP, which generalizes the fully independent training conditional (FITC) (Snelson and Ghahramani 2006) method to non-Gaussian likelihoods. FITC assumes that each training and test point is independent of all other points given the function values at the inducing points. However, this may be less appropriate for the pairwise likelihood (Equation 2) because it ignores the covariance between the utilities of the items in the training pairs. In practice, memory costs that grow linearly with $\mathcal{O}(N)$ start to become a problem with more than a few thousands points, which prohibits the use of GFITC in many large datasets. It is also unclear how to apply distributed computation to GFITC to tackle the growing computational costs (Hensman et al. 2015).

We now derive a more scalable approach for GPPL and crowdGPPL using stochastic variational inference (SVI), an iterative scheme that limits the computational and memory costs at each iteration (Hoffman et al. 2013). As we explain below, this allows us to reduce the computational cost of each iteration of the algorithm from $\mathcal{O}(NM^2)$ with GFITC to $\mathcal{O}(P_iM^2 + Pi^2M + M^3)$, where P_i is the mini-batch size, and memory complexity from $\mathcal{O}(NM)$ to $\mathcal{O}(P_iM + M^2 + Pi^2)$. Neither P_i nor M are dependent on the size of the dataset, meaning that SVI can be run with arbitrarily large datasets. 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 Equation 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:

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{y}, s) &\propto \prod_{p=1}^{P} p(y_p|z_p)p(\boldsymbol{f}|\boldsymbol{K}, s) = \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}/s) \\ &\approx \prod_{p=1}^{P} \mathcal{N}(y_p; \Phi(z_p), Q_{p,p})\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}/s) = \mathcal{N}(\boldsymbol{y}; \Phi(\boldsymbol{z}), \boldsymbol{Q})\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}/s), \end{aligned} \tag{8}$$

where $\boldsymbol{Q}$ is a diagonal noise covariance matrix. For crowdGPPL, we use the same approximation to the likelihood, but replace $\boldsymbol{f}$ with $\boldsymbol{F}$ throughout this section.

Since each $\Phi(z_p)$ term defines a Bernoulli distribution over y_p, we estimate the diagonals of $\boldsymbol{Q}$ by moment matching the variance of the Bernoulli distributions to give $Q_{p,p} = \Phi(z_p)(1 - \Phi(z_p))$. However, this means that the variance of the approximate likelihood depends on $\boldsymbol{z}$ and therefore on $\boldsymbol{f}$, which means the posterior remains intractable. We resolve this by approximating $Q_{p,p}$ independently for each pairwise label, p, thereby replacing intractable expectations with respect to $p(\boldsymbol{f}|\boldsymbol{y})$ with simple updates to the parameters of the conjugate priors over each $\Phi(z_p)$. The conjugate prior for the Bernoulli distribution with parameter $\Phi(z_p)$ is a beta distribution with parameters γ and λ. We find γ and λ by matching the moments of the beta distribution to the mean and variance of the prior defined by the Gaussian process, $p(\Phi(z_p)|\boldsymbol{K}_\theta, \alpha_0, \beta_0)$, found using numerical integration. Assuming a beta prior over $\Phi(z_p)$ means that y_p has a beta-Bernoulli distribution. Given the observed label y_p, we use the variance of the posterior beta-Bernoulli to estimate the diagonal variance terms in $\boldsymbol{Q}$ as:

$$Q_{p,p} \approx \frac{(\gamma + y_p)(\lambda + 1 - y_p)}{\gamma + \lambda + 1}. \tag{9}$$

This approximation has previously given good empirical performance when used 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 of $\boldsymbol{f}$:

$$\Phi(\boldsymbol{z}) \approx \tilde{\Phi}(\boldsymbol{z}) = \boldsymbol{G}(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}]) + \Phi(\hat{\boldsymbol{z}}), \tag{10}$$

$$G_{p,i} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\hat{z}_p))(2y_p - 1)([i = a_p] - [i = b_p]) \tag{11}$$

where $\boldsymbol{G}$ is a matrix containing elements $G_{p,i}$, which are the partial derivatives of the pairwise likelihood with respect to each of the latent function values, $\boldsymbol{f}$. This creates a circular dependency between the posterior mean of $\boldsymbol{f}$, which is needed to compute $\hat{\boldsymbol{z}}$, and $\boldsymbol{G}$. These terms can be estimated using a variational inference procedure that iterates between updating $\boldsymbol{f}$ and $\boldsymbol{G}$ (Steinberg and Bonilla 2014), and is described in more detail below.

The complete approximate posterior for GPPL is now as follows:

$$p(\boldsymbol{f}|\boldsymbol{y}, s) \approx \frac{1}{Z}\mathcal{N}(\boldsymbol{y}; \boldsymbol{G}(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}]) + \Phi(\hat{\boldsymbol{z}}), \boldsymbol{Q})\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}/s) = \mathcal{N}(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}), \tag{12}$$

where Z is a normalisation constant. Linearisation means that our approximate likelihood is now conjugate to the prior, so the approximate posterior is also Gaussian. Gaussian approximations to the posterior have shown strong empirical results for tasks such as classification (Nickisch and Rasmussen 2008), including the expectation propagation method (Rasmussen and Williams 2006). Linearisation using a Taylor expansion has also been widely tested in the extended Kalman filter (Haykin 2001), as well as applied previously to Gaussian processes with non-Gaussian likelihoods (Steinberg and Bonilla 2014; Bonilla et al. 2016). Given our approximate posterior, we can now derive an efficient inference scheme using SVI.

4.2 SVI for Single User GPPL

Our linear approximation in the previous section permits iterative approximate inference for GPPL given s. However, computing the posterior covariance and mean requires inverting $\boldsymbol{K}$, which has 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). First, we introduce a sparse approximation to the Gaussian process that allows us to limit the size of the covariance matrices we need to work with. This sparse approximation introduces a set of $M \ll N$ *inducing* items with inputs $\boldsymbol{X}_m$, utilities $\boldsymbol{f}_m$, covariance $\boldsymbol{K}_{mm}$, and covariance between the observed and inducing items, $\boldsymbol{K}_{nm}$. For clarity, we omit θ from this point on. We now assume a *mean-field* approximation to the posterior over the inducing and training items that factorises between different sets of latent variables:

$$p(\boldsymbol{f}, \boldsymbol{f}_m, s | \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_m, k_\theta, \alpha_0, \beta_0) \approx q(\boldsymbol{f}, \boldsymbol{f}_m, s) = q(s)q(\boldsymbol{f})q(\boldsymbol{f}_m), \tag{13}$$

where $q(.)$ are *variational factors*, which we define below. Each factor corresponds to a subset of latent variables, $\boldsymbol{z}$, and takes the form $\ln q(\boldsymbol{z}) = \mathbb{E}_{\not{\boldsymbol{z}}}[\ln p(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{y})]$. That is, the expectation with respect to all other latent variables, $\not{\boldsymbol{z}}$, of the log joint distribution of the observations and the current subset of latent variables, $\boldsymbol{z}$.

To obtain the factor for $\boldsymbol{f}_m$, we marginalise $\boldsymbol{f}$ and take expectations with respect to $q(s)$:

$$\begin{aligned}\ln q(\boldsymbol{f}_m) &= \ln \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}\right) + \ln \mathcal{N}\left(\boldsymbol{f}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}\left[s\right]\right) + \text{const}, \\ &= \ln \mathcal{N}(\boldsymbol{f}_m; \hat{\boldsymbol{f}}_m, \boldsymbol{S}),\end{aligned} \tag{14}$$

where the variational parameters $\hat{\boldsymbol{f}}_m$ and $\boldsymbol{S}$ are computed using the 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:

$$\ln q(\boldsymbol{f}) = \ln \mathcal{N}(\boldsymbol{f}; \boldsymbol{A}\hat{\boldsymbol{f}}_m, \boldsymbol{K} + \boldsymbol{A}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{A}^T), \tag{15}$$

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

$$\ln q(s) = \mathbb{E}_{q(\boldsymbol{f}_m)}[\ln \mathcal{N}(\boldsymbol{f}_m | \boldsymbol{0}, \boldsymbol{K}_{mm}/s)] + \ln \mathcal{G}(s; \alpha_0, \beta_0) + \text{const} = \ln \mathcal{G}(s; \alpha, \beta), \tag{16}$$

where $\alpha = \alpha_0 + \frac{M}{2}$ and $\beta = \beta_0 + \frac{\text{tr}(\boldsymbol{K}_{mm}^{-1}(S + \hat{\boldsymbol{f}}_m \hat{\boldsymbol{f}}_m^T))}{2}$. 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 (Equation 13) by maximizing a lower bound, $\mathcal{L}$, on the log marginal likelihood (see also the detailed equations in Appendix A), which is given by:

$$\begin{aligned}&\ln p(\boldsymbol{y}|\boldsymbol{K}, \alpha_0, \beta_0) = \text{KL}(q(\boldsymbol{f}, \boldsymbol{f}_m, s) || p(\boldsymbol{f}, \boldsymbol{f}_m, s | \boldsymbol{y}, \boldsymbol{K}, \alpha_0, \beta_0)) + \mathcal{L} \\ &\mathcal{L} = \mathbb{E}_{q(\boldsymbol{f})}[\ln p(\boldsymbol{y}|\boldsymbol{f})] + \mathbb{E}_{q(\boldsymbol{f}_m, s))}[\ln p(\boldsymbol{f}_m, s | \boldsymbol{K}, \alpha_0, \beta_0) - \ln q(\boldsymbol{f}_m) - \ln q(s)].\end{aligned} \tag{17}$$

To optimize $\mathcal{L}$, we initialize 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 maximized iteratively 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. We then perform updates using $\boldsymbol{Q}_i$ (rows and columns of $\boldsymbol{Q}$ for observations in $\boldsymbol{P}_i$), $\boldsymbol{K}_{im}$ and $\boldsymbol{A}_i$, (rows of $\boldsymbol{K}_{nm}$ and $\boldsymbol{A}$ referred to by $\{y_p \forall p \in \boldsymbol{P}_i\}$), $\boldsymbol{G}_i$ (rows of $\boldsymbol{G}$ in $\boldsymbol{P}_i$ and columns referred to by any items

$\{a_p \forall p \in \boldsymbol{P}_i\} \cup \{b_p \forall p \in \boldsymbol{P}_i\}$), and $\hat{\boldsymbol{z}}_i = \{\mathbb{E}[\boldsymbol{z}_p] \forall p \in P_i\}$. The update equations optimize the natural parameters of the Gaussian distribution by following the natural gradient (Hensman et al. 2015):

$$\boldsymbol{S}_i^{-1} = (1-\rho_i)\boldsymbol{S}_{i-1}^{-1} + \rho_i \left(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1} + \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i \right) \tag{18}$$

$$\hat{\boldsymbol{f}}_{m,i} = \boldsymbol{S}_i \left((1-\rho_i)\boldsymbol{S}_{i-1}^{-1}\hat{\boldsymbol{f}}_{m,i-1} + \rho_i \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{f}}_{m,i} \right) \right) \tag{19}$$

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 computational complexity of Equations 18 and 19 has order $\mathcal{O}(M^3)$, and the second term has order $\mathcal{O}(P_i M^2 + P_i^2 M + M^3)$. The P_i^2 term arises due to G_i, which is an $N_i \times P_i$ matrix, where $N_i \leq 2P_i$ is the number of items referred to by the pairwise labels in the mini-batch. Memory complexity is now bounded by $\mathcal{O}(M^2 + P_i^2 + MP_i)$, where each complexity term is due to the sizes of K_{mm}, G_i and K_{im}. Note that 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.

The complete SVI algorithm is summarized in Algorithm 1. Using an inner

```
  Input: Pairwise labels, $\boldsymbol{y}$, training item features, $\boldsymbol{x}$, test item features $\boldsymbol{x}^*$
1 Compute kernel matrices $\boldsymbol{K}$, $\boldsymbol{K}_{mm}$ and $\boldsymbol{K}_{nm}$ given $\boldsymbol{x}$ Initialise $\mathbb{E}[s]$, $\mathbb{E}[\boldsymbol{f}]$and $\hat{\boldsymbol{f}}_m$ to
    prior means and $\boldsymbol{S}$ to prior covariance $\boldsymbol{K}_m m$;
  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.

loop to learn $\boldsymbol{G}_i$ avoids the need to store the complete matrix, $\boldsymbol{G}$. It is possible to distribute computation in lines 3-6 by selecting multiple random samples to be processed 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 the mean over the values computed by each node, weighted by the size of each batch. This combination step is permitted without modifying Equations 18 and 19, since they already contain a sum weighted by π_i.

Inducing point locations can be learned as part of the variational inference procedure (Hensman et al. 2015), or by optimizing a bound on the log marginal likelihood. However, the former breaks the convergence guarantees, and both approaches may add substantial computational cost. We find that we are able to 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 spread of observations across feature space.

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

$$\boldsymbol{f}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m, \qquad \boldsymbol{C}^* = \boldsymbol{K}_{**} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{K}_{*m}^T\boldsymbol{K}_{mm}^{-1}, \tag{20}$$

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 extend the SVI method to the crowd preference learning model proposed in Section 3.2. To begin with, we extend the variational posterior for GPPL (Equation 13) to the crowdGPPL model defined in Equation 7:

$$\begin{aligned} &p(\boldsymbol{V}, \boldsymbol{V}_m, \boldsymbol{W}, \boldsymbol{W}_m, \boldsymbol{t}, \boldsymbol{t}_m, s_1^{(v)}, .., s_C^{(v)}, s_1^{(w)}, .., s_C^{(w)}, s^{(t)} | \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_m, \boldsymbol{U}, \boldsymbol{U}_m, k, \alpha_0, \beta_0) \\ &\approx q(\boldsymbol{t})q(\boldsymbol{t}_m)q\left(s^{(t)}\right)\prod_{c=1}^{C} q(\boldsymbol{v}_c)q(\boldsymbol{w}_c)q(\boldsymbol{v}_{c,m})q(\boldsymbol{w}_{c,m})q\left(s_c^{(v)}\right)q\left(s_c^{(w)}\right) \end{aligned} \tag{21}$$

where $\boldsymbol{U}_m$ are the feature vectors of inducing users and the variational q factors are defined below. SVI can then be used to optimize the lower bound on the marginal likelihood (see also Equation 33 in the Appendix), given by:

$$\begin{aligned} \mathcal{L}_{cr} = &\mathbb{E}_{q(\boldsymbol{F})}[\ln p(\boldsymbol{y}|\boldsymbol{F})] + \mathbb{E}_{q(\boldsymbol{t}_m),q(s^{(t)})}[\ln p(\boldsymbol{t}_m, s^{(t)}|\boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \ln q(\boldsymbol{t}_m)] \\ &\sum_{c=1}^{C}\Big\{\mathbb{E}_{q(\boldsymbol{v}_{m,c}),q(s_c^{(v)})}[\ln p(\boldsymbol{v}_{m,c}, s_c^{(v)}|\boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \ln q(\boldsymbol{v}_{m,c})] \\ &+ \mathbb{E}_{q(\boldsymbol{w}_{m,c})}[\ln p(\boldsymbol{w}_{m,c}|\boldsymbol{L}_{mm}/s_c^{(w)}) - \ln q(\boldsymbol{w}_{m,c})]\Big\}. \end{aligned} \tag{22}$$

The algorithm (detailed in Algorithm 2 in Appendix) follows the same pattern as Algorithm 1, updating each of the q factors in turn by computing means and covariances for $\boldsymbol{V}_m$, $\boldsymbol{W}_m$ and $\boldsymbol{t}_m$ instead of $\boldsymbol{f}_m$.

The variational factor for the cth inducing item component is:

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

where the posterior mean $\hat{\boldsymbol{v}}_{m,c}$ and covariance $\boldsymbol{S}_c^{(v)}$ are computed using update equations of the same form as those of the single user GPPL in Equations 18 and 19. The noise precision, $\boldsymbol{Q}^{-1}$, in Equation 18 is scaled by expectation terms over $\boldsymbol{w}_{m,c}$, and $\hat{\boldsymbol{f}}_{m,i}$ is replaced by $\hat{\boldsymbol{v}}_{m,c,i}$. For reasons of space, we provide the complete equations for $\hat{\boldsymbol{v}}_{m,c}$ and $\boldsymbol{S}_c^{(v)}$ in Appendix B, Equations 34 and 35.

The factor for the inducing points of $\boldsymbol{t}$ follows a similar pattern to $\boldsymbol{v}_{m,c}$:

$$\begin{aligned}\ln q(\boldsymbol{t}_m) = &\ \ln \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right) + \ln \mathcal{N}(\boldsymbol{t}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}[s^{(t)}]) + \text{const} \\ = &\ \ln \mathcal{N}\left(\boldsymbol{t}_m; \hat{\boldsymbol{t}}_m, \boldsymbol{S}^{(t)}\right), \end{aligned} \tag{24}$$

where the posterior mean $\hat{\boldsymbol{t}}$ and covariance $\boldsymbol{S}^{(t)}$ uses the same updates Equations 18 and 19, except $\hat{\boldsymbol{f}}_{m,i}$ is replaced by $\hat{\boldsymbol{t}}_{m,i}$ (see also Equations 37 and 38).

Finally, the variational distribution for each inducing users component is:

$$\begin{aligned}\ln q(\boldsymbol{w}_{m,c}) = &\ \ln \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right) + \ln \mathcal{N}(\boldsymbol{w}_{m,c}; \boldsymbol{0}, \boldsymbol{L}_{mm}/s_c^{(w)}) + \text{const} \\ = &\ \ln \mathcal{N}\left(\boldsymbol{w}_{m,c}; \hat{\boldsymbol{w}}_{m,c}, \boldsymbol{\Sigma}_c\right), \end{aligned} \tag{25}$$

where $\hat{\boldsymbol{w}}_c$ and $\boldsymbol{\Sigma}_c$ also follow the pattern of Equations 18 and 19, with the noise precision, $\boldsymbol{Q}^{-1}$, scaled by expectations terms involving $\boldsymbol{w}_{c,m}$, and $\hat{\boldsymbol{f}}_{m,i}$ replaced by $\hat{\boldsymbol{w}}_{m,c,i}$. The full definitions are given in Appendix B, Equations 37 and 38. The computational and memory complexity are the same as for single-user GPPL, except that we now have C updates, and the number of inducing points for items and users may be different: $\mathcal{O}(CM_{\text{items}}^3 + CM_{\text{items}}^2 P_i + CM_{\text{items}} P_i^2 + CM_{\text{users}}^3 + CM_{\text{users}}^2 P_i + CM_{\text{users}} P_i^2)$. Memory complexity for crowdGPPL is $\mathcal{O}(CM_{\text{items}}^2 + P_i^2 + M_{\text{items}} P_i + CM_{\text{users}}^2 + M_{\text{users}} P_i$.

The expectations for the inverse scales, $s_1^{(v)}, .., s_c^{(v)}$, $s_1^{(w)}, .., s_c^{(w)}$ and $s^{(t)}$ can be computed using Equation 16 by substituting in the corresponding terms for each $\boldsymbol{v}_c$, $\boldsymbol{w}_c$ or $\boldsymbol{t}$ instead of $\boldsymbol{f}$. As with GPPL, the stochastic updates are amenable to parallel computation within one iteration of the variational inference algorithm, by performing computations for mini-batches of training data in parallel.

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:

$$\boldsymbol{F}^* = \hat{\boldsymbol{t}}^* + \sum_{c=1}^{C} \hat{\boldsymbol{v}}_c^{*T} \hat{\boldsymbol{w}}_c^*, \qquad \boldsymbol{\Lambda}_u^* = \boldsymbol{C}_t^* + \sum_{c=1}^{C} \omega_{c,u}^* \boldsymbol{C}_{v,c}^* + \hat{w}_{c,u}^2 \boldsymbol{C}_{v,c}^* + \omega_{c,u}^* \hat{\boldsymbol{v}}_c \hat{\boldsymbol{v}}_c^T, \tag{26}$$

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 for the user components for the test users. These terms are defined in Appendix C, Equations 41 to 43. The mean $\boldsymbol{F}^*$ and covariances Λ_u^* can be inserted into Equation 2 to predict pairwise labels. In practice, the full covariance terms are needed only for Equation 2, so need only be computed between items for which we wish to predict pairwise labels.

Dataset	#folds/ samples	#users	#items	train #pairs	test #pairs	#scores	#features items	#features users
Simulation a	25	25	100	900	0	100	2	2
Simulation 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	1000	18	123
Sushi A	25	100	10	2000	2500	1000	18	123
Sushi B	25	5000	100	50000	5000	500000	18	123
UKPConvArg-CrowdSample	32	1442	1052	16398	529	33	32310	0

Table 1 Summary of datasets showing counts per subsample. For simulations, we generate the subsamples of data independently, for Sushi we select subsamples independently from the dataset. Values for UKPConvArgCrowdSample are means per fold, where the test data in each fold corresponds to a single topic and stance. Numbers of features are given after categorical labels have been converted to one-hot encoding, counting each category as a separate feature.

5 Experiments

Our experiments test the key aspects of *crowdGPPL*: modelling personal preferences from pairwise labels; predicting consensus utilities from noisy crowdsourced preferences in a subjective task; and the scalability of our proposed SVI method. In Section 5.2, we use simulated data to test the robustness of crowdGPPL to noise and unknown numbers of latent components. Section 5.3, compares different configurations of the model against alternative approaches using the *Sushi* datasets[2] (Kamishima 2003). Section 5.4 evaluates the ability of crowdGPPL to predict both personal and consensus utilities in a high-dimensional natural language processing (NLP) task with sparse, noisy crowdsourced preferences (*UKP-ConvArgCrowdSample*[3], Simpson and Gurevych (2018)). Using the same dataset, Section 5.4 analyses the scalability of our SVI approach. Finally, Section 5.5 evaluates whether crowdGPPL ignores redundant components when the number of components, C, is larger than necessary. The datasets are summarised in Table 1.

5.1 Method Comparison

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.

The *collaborative Gaussian process* (*collabGP*, Houlsby et al. (2012)), is a similar model that, like crowdGPPL, assumes Gaussian processes over latent user and item components, but does not learn a consensus or the scales of the components to identify active and unneeded components. The inference method maintains a set of parameters for each pairwise label and for each user, but uses an inducing point method, *GFITC*, to reduce computational complexity, as described in Section 4. As collabGP requires more parameters, it has higher memory complexity ($\mathcal{O}(M_{\text{items}}^2 + M_{\text{items}}P + P^2 + M_{\text{users}}^2 + M_{\text{users}}P + U^2$) in contrast to

[2] `http://www.kamishima.net/sushi/`
[3] `https://github.com/ukplab/tacl2018-preference-convincing`

$\mathcal{O}(CM_{\text{items}}^2 + P_i^2 + M_{\text{items}}P_i + CM_{\text{users}}^2 + M_{\text{users}}P_i)$. We therefore treat the performance of collabGP as a target for crowdGPPL.

CrowdBT (Chen et al. 2013) has been shown to be an effective method for learning from crowdsourced preferences. CrowdBT learns a model of each worker's accuracy and infers consensus utilities, therefore 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 and is hence purely an aggregation method. However, the aggregated utilities produced by crowdBT can be treated as training labels for regression. Here, we use the posterior mean utilities from crowdBT as target values for training a Gaussian process regressor, and set the observation noise variance of the GP equal to the posterior variance of the utilities estimated by crowdBT. This pipeline method, *crowdBT–GP*, allows us to compare predictive performance when using crowdBT, which treats annotator differences as noise, against crowdGPPL, which models the individual preferences of annotators.

5.2 Simulated Noisy Data

First, we test how well crowdGPPL is able to recover an underlying consensus function from pairwise labels with varying amounts of noise. The consensus for GPPL-per-user is the mean of all users' predicted utilities. For crowdGPPL, we set $C = 20$ and for all models, we set $\alpha_0 = 1$, $\beta_0 = 100$, and use Matérn 3/2 kernels with length-scales chosen by a median heuristic:

$$l_{d,MH} = \text{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1,..,N, \forall j = 1,...,N\})D. \tag{27}$$

This is a computationally frugal heuristic for choosing the length-scales, which normalizes features and prevents the average covariance between items or users from shrinking as the number of features grows. The SVI hyperparameters were set to $r = 0.9$, $P_i = 1000$ and $\epsilon = 1$.

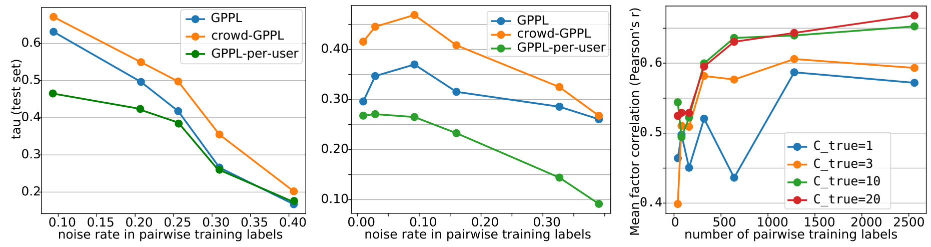
is D actually defined? I think the comment below stems from confusion that this is not just a median function. But anyway, why not add the refs for using a median? Page 13, equation (25): please provide reference for 'Dmedian' function.

For the first test, we generate data by creating a 20×20 grid of points and choosing 400 pairs of these points at random, which are split into 50% training and test sets. For each point, we generate pairwise labels by drawing from crowdGPPL with 20 users, 5 latent components, and $s = 0.001$. We vary the precision of the consensus function, σ, to control the noise in the consensus function. The complete experiment was repeated 25 times, including generating new data for each value of σ. Figure 1a shows that the crowdGPPL model is better able to recover the latent consensus function than the other methods, even when noise levels are high. GPPL's predictions may be worsened by biased users whose preferences deviate consistently from the consensus. GPPL-per-user relies on separate instances of GPPL, so does not benefit from sharing information between users when training.

The second simulation modifies the previous setup by fixing $\sigma = 10$ and varying s to evaluate the methods' ability to recover the personal preferences of simulated users. Results in Figure 1b show that crowdGPPL is able to make better predictions when noise is below 0.3 but its benefit disappears when the noise level increases further.

We hypothesize that learning a model with a larger number of latent components requires more training data. In the third simulation, we generate data using the same setup as before, but fix $s = 0.2$ and $\sigma = 1$ and vary the number of pairwise

Page 13, line 41: Can you provide some intuition as to



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

Fig. 1 Simulations: Rank correlation between true and inferred preference values for different inference tasks. (a) & (b) varying level of noise in pairwise training labels, (c) varying number of pairwise training labels.

training labels and the number of true components through $C_{true} \in \{1, 3, 10, 20\}$. To evaluate the correlation between inferred and true user components, we compute Pearson correlations between each true component and each inferred component, then repeatedly select pairs of components with the highest correlations until every true component is matched to an inferred component. In Figure 1c we plot the mean of the correlations between matched pairs of components. For all values of C_{true}, increasing the number of training labels beyond 1000 leads to only minor increases in correlation at best. Performance is highest when $C_{true} = 20$, possibly because the predictive model has $C = 25$ and hence is a closer match to the generating model. However, for all values of C_{true}, performance with > 500 labels remains above 0.5, showing the model is reasonably robust to mismatches between C and C_{true}.

5.3 Sushi Preferences

We use sushi preference data (shown in Table 1), to compare GPPL and crowdGPPL with different sets of features against the previous benchmark method, *collaborative GP (collabGP)* (Houlsby et al. 2012). Collaborative GP uses a simpler model than crowdGPPL, with no consensus function, and no learning of the output scales of the utility functions, which makes it more important to select the correct number of factors in advance. It also uses an inference scheme based on expectation propagation (EP) and variational Bayes, which has a couple of important practical drawbacks: firstly, it does not infer the utilities, but learns pairwise labels directly, so its predictions cannot directly be used to rank or rate large sets of items; secondly, the number of parameters in the model is proportional to the dataset size, as it represents every combination of (user, item-pair) with a separate set of parameters, rather than working with inducing points, which may prevent the use of the model with large datasets due to memory constraints. However, as the EP approach of collabGP has been consistently shown to perform well for non-Gaussian likelihoods(**?**), we consider its performance as a target for our SVI approach for crowdGPPL.

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

	Sushi-A-small			**Sushi-A**			**Sushi-B**		
Method	Acc	CEE	τ	Acc	CEE	τ	Acc	CEE	τ
crowdGPPL	**.71**	**.56**	.48	.84	.33	.79	.74	.58	. 53
crowdGPPL \inducing	.70	.60	.45	.84	.34	.78	-	-	-
crowdGPPL $\backslash u$	.70	.58	.46	**.85**	**.31**	**.80**	**.78**	.52	.57
crowdGPPL $\backslash u \backslash x$	**.71**	.57	**.49**	**.85**	.33	.80	.77	.50	.58
crowdGPPL $\backslash u, \backslash t$	.68	.60	.43	.84	.33	.80	.76	.50	.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$\backslash u$	.69	.59	n/a	.84	.33	n/a	.76	.50	n/a
GPVU	.70	.67	.43	.72	.67	.42	.73	.59	.52

Table 2 Predicting personal preferences: performance on *Sushi-A* dataset and *Sushi-B* datasets, means over 25 repeats. CrowdGPPL uses $C = 20$ unless otherwise specified. The standard deviations of all metrics shown are ≤ 0.02, for CEE, ≤ 0.08, for τ, ≤ 0.03. For Sushi-B, crowdGPPL, GPPL-per-user and collabGP had runtimes of approximately 30 minutes on a 12 core, 2.6GHz CPU server, while GPPL required only one minute.

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 a subset of 100 users at random from the complete dataset, then split the data into training and test sets by randomly selecting 20 pairs for each user for training and 25 for testing. For *Sushi-B*, we use all 5000 workers, and subsample 10 training pairs and 1 test pair per user. Beside GPPL, crowdGPPL and GPPL-per-user, we introduce four further baselines: *crowdGPPL*$\backslash\boldsymbol{u}$, which ignores the user features; *crowdGPPL*$\backslash\boldsymbol{u}\backslash\boldsymbol{x}$, which ignores both user and item features; *crowdGPPL*$\backslash\boldsymbol{u}\backslash\boldsymbol{t}$, which excludes the consensus function $\boldsymbol{t}$ from the model as well as the user features; and *crowdGPPL*$\backslash$*induc*, which uses all features but does not use inducing points for a sparse approximation. For $\backslash\boldsymbol{u}$ methods, the user covariance matrix, $\boldsymbol{K}_w$, in the crowdGPPL model is replaced by the identity matrix, and for crowdGPPL$\backslash\boldsymbol{u}\backslash\boldsymbol{x}$, the item covariance matrices, $\boldsymbol{K}_v$ and $\boldsymbol{K}_t$ are also replaced by the identity matrix. We set hyperparameters $C = 20$ without optimization, $\alpha_0 = 1, \beta_0 = 100$ using a grid search over values $10^{\{-1,\ldots,3\}}$ on withheld user data from the *Sushi-A* dataset, $\epsilon = 5$, $P_i = 200$ for *Sushi-A* and $P_i = 2000$ for *Sushi-B*. All other hyperparameters are the same as for Section 5.2. The complete process of subsampling, training and testing, was repeated 25 times for each dataset.

clarify the parameters for sushi and for conv

The results in Table 2 illustrate the performance benefit of crowd models over single-user GPPL, and the runtimes show the speedup of the sparse approximation of crowdGPPL over crowdGPPL$\backslash$induc. GPPL is substantially quicker to train than crowdGPPL, and GPPL-per-user does not scale well to larger numbers of users. When using inducing points, the user features decrease the performance of crowdGPPL: crowdGPPL$\backslash$induc and crowdGPPL$\backslash\boldsymbol{u}$ both outperform the full crowdGPPL. To use inducing points, there must be a strong relationship between neighbouring points, which may not to be the case given the features in this dataset, since they describe only very general characteristics, such as the user's region and age group. Comparing crowdGPPL$\backslash\boldsymbol{u}$ with crowdGPPL$\backslash\boldsymbol{ut}$, including the consensus function appears to improve performance by a modest amount.

	Consensus			Personal: all workers			>40 training pairs		
Method	Acc	CEE	τ	Acc	CEE	τ	Acc	CEE	τ
GPPL	.76	.51	.48	.71	**.56**	.32	.72	**.55**	.26
crowdGPPL	**.78**	**.50**	**.53**	**.73**	.58	**.33**	**.74**	.58	**.29**
crowdBT-GP	.77	.52	.50	.69	.58	.30	.70	.57	.24

Table 3 UKPConvArgCrowdSample, using ling+GloVe features: predicting consensus values, personal preferences for all workers, and personal preferences for workers who had >40 pairs in the training set. Classification accuracy (*Acc*) and cross entropy error (or log-loss, *CEE*) evaluate pairwise predictions, Kendall's τ evaluates the predicted utilities. Runtimes on a 12 core, 2.6GHz CPU server were approximately 3 minutes for GPPL and crowdBT-GP, 60 minutes for crowdGPPL.

The strong performance of GPPL-per-user suggests that even ten preferences per person were enough to learn a reasonable model for *Sushi-B*.

Like crowdGPPL, collaborative GP (collabGP) considers user features as inputs but requires many more model parameters. This allows it to outperform crowdGPPL on accuracy and CEE but does not provide a ranking function for computing Kendall's τ. The inference method of collabGP did not present practical problems on the Sushi dataset on computer with 16GB RAM, and in fact produces relatively fast run times due to its more efficient implementation using a C library in place of Python. However, the results show that crowdGPPL is able to obtain competitive performance despite the approximations required for our proposed SVI method. In the next experiment, we test the approach on a larger dataset with many more pairs, users and items, for which the memory requirements of collabGP become problematic.

some comment on the strength of GPPL-per-user?

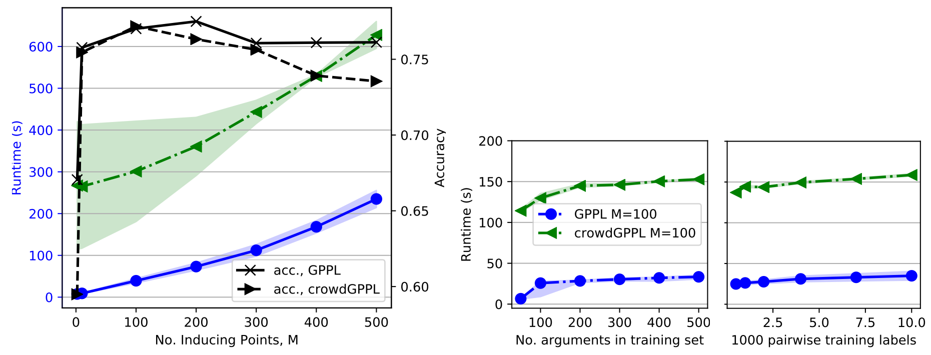
does houlsby learn the output scales of the factors? If not, then they are not learning to discard the unused ones. Make sure to mention this when presenting the output scale plot and model differences.

5.4 Argument Convincingness

We evaluate consensus learning, personal preference learning and scalability on an NLP task, namely identifying convincing arguments. The dataset, *UKPConvArgCrowdSample*, was subsampled by Simpson and Gurevych (2018) from the crowdsourced data provided by Habernal and Gurevych (2016), and contains arguments written by users of online debating forums, along with crowdsourced judgments of pairs of arguments indicating the most convincing argument. Each argument is represented by $32,310$ numerical features and the dataset is divided into 32 folds (16 topics, each of which has two opposing stances). For each fold, we train on 31 folds and test on the remaining fold. We extend the task described in Simpson and Gurevych (2018) to predict not just the consensus, but also the personal preferences of individual crowd workers. GPPL was previously shown to outperform SVM, Bi-LSTM and Gaussian process classifier methods at consensus prediction for *UKPConvArgCrowdSample* (Simpson and Gurevych 2018). We hypothesize that a worker's view of convincingness depends on their prior beliefs and understanding of the subject discussed, and that crowdGPPL may therefore predict unseen pairwise labels or rankings for individual workers or the consensus more accurately than GPPL, by accounting for the biases of individual workers.

Table 3 shows performance for GPPL and crowdGPPL. The hyperparameters were kept the same as in Section 5.2 except for: GPPL uses $\alpha_0 = 2$, $\beta_0 = 200$ and crowdGPPL uses $\alpha_0 = 2$, $\beta_0 = 2000$, set by comparing training set accuracy

New strategy == add so In the Houlsby paper, it appears that they use their own EP implementation for the models they



(a) Varying M with complete training set (b) Varying N. (c) Varying P

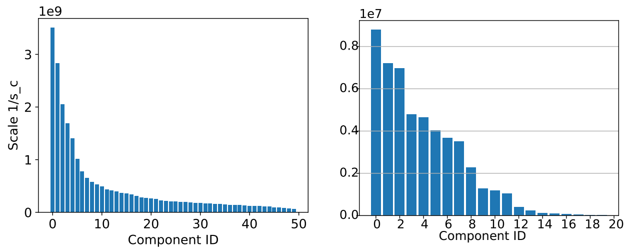
Fig. 2 Wall-clock times for training+prediction of the consensus utilities on UKPConvArgCrowdSample. Lines show means over 32 runs, with bands indicating one standard deviation. CrowdGPPL with $C = 5$ components.

against $\alpha_0 = 2, \beta_0 = 20000$ and $\alpha_0 = 2, \beta_0 = 2$; $C = 50$ and $\epsilon = 2$, which were not optimized. CrowdGPPL outperforms GPPL at predicting the consensus pairwise labels shown by classification accuracy (acc) and cross entropy error (CEE), and the consensus ranking (significant with $p < 0.05$, Wilcoxon signed-rank test), shown by Kendall's τ rank correlation. For the personal preference predictions, crowdGPPL also outperforms GPPL at ranking pairwise label accuracy, suggesting that there is a benefit to modeling individual workers when predicting the consensus. The benefits of crowdGPPL on this task may be restricted because the pairwise comparisons in the test folds are noisy and contain numerous contradictions (Habernal and Gurevych 2016). For workers with more training data, the more complex crowdGPPL model is able to improve further over GPPL. Since the CEE scores of crowdGPPL are lower than those of GPPL, while accuracy is higher, crowdGPPPL may be slightly under-confident.

We examine the scalability of our SVI implementation by evaluating GPPL and crowdGPPL with different numbers of inducing points, M. Here, we fix $C = 5$ and keep other model hyperparameters and experimental setup the same as for Table 3. Figure 2a shows the trade-off between runtime and accuracy as an effect of choosing M. Accuracy peaks using $M = 200$ while the runtime continues to increase rapidly in a polynomial fashion. Since there are $33,210$ features, the runtime includes large overheads due to the computation of the covariance matrices, which is linear in the number of features.

Figures 2b and 2c show runtimes as a function of the number of items in the training set, N_{tr}, and the number of pairwise training labels, P, respectively (all other settings remain as in Figure 2a). The runtime increase with N_{tr} for GPPL is almost imperceptible, while for crowdGPPL it increases almost linearly as more computations over the set of items are required than for GPPL, and the method takes more iterations to converge with more items. However, many of the additional computations can be parallelised in future implementations. Increasing the number of pairwise labels, P, above 1000, however, does not visibly affect the runtimes.

same as figure for inducing points but for pairwise labels in each mini batch?



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

Fig. 3 Distribution of latent component variances, $1/s_c$ in crowdGPPL, means over all runs.

5.5 Posterior Variance of Item Components

We investigate how many latent components were actively used by crowdGPPL on the *UKPConvArgCrowdSample* and *Sushi-A* datasets using the median heuristic. Figure 3 plots the posterior expectations of the inferred scales, $1/s_c$, for the latent item components. The plots show that many factors have a very small variance and therefore do not contribute strongly to many of the model's predictions. This indicates that our Bayesian approach, in which the priors of the latent factors have mean zero, has inferred a simpler model than the number of latent components would permit.

6 Conclusions

We proposed a novel Bayesian preference learning approach for modelling both the preferences of individual users 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 factorization to capture divergences in opinion. Unlike previous work, our method can scale to arbitrarily large datasets, since its computational and memory complexity are bounded independently of dataset size through the use of stochastic variational inference. Our empirical results confirm that our approach scales well with the amount of training data, and that jointly modelling the consensus and personal preferences of users can improve the predictions of both. On a benchmark recommendation dataset, our approach performs competitively with rival methods with inferior computational and memory complexity, despite the approximations required for our stochastic inference method. When predicting a consensus from crowdsourced data, our model, CrowdGPPL, improves on the previous state of the art (Simpson and Gurevych 2018), which did not account for personal preferences.

Future work will evaluate the benefit of learning inducing point locations from data and optimizing length-scale hyperparameters by maximising $\mathcal{L}$ as part of the variational inference method. Another important direction is to generalise the likelihood from pairwise comparisons to comparisons involving k items Pan et al. (2018) or best–worst scaling (Kiritchenko and Mohammad 2017), to make it possible to exploit Gaussian processes and Bayesian matrix factorisation when given other forms of comparative preference data.

Acknowledgments

References

Abbasnejad E, Sanner S, Bonilla EV, 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

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, ACM, pp 9–18

Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp 1027–1035

Banerji M, Lahav O, Lintott CJ, Abdalla FB, Schawinski K, Bamford SP, Andreescu D, Murray P, Raddick MJ, Slosar A, 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: International Conference on Machine Learning, pp 1651–1659

Bradley RA, Terry ME (1952) Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 39(3/4):324–345

Chen G, Zhu F, Heng PA (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

Chen X, Bennett PN, 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, ACM, pp 193–202

Chen Y, Suh C (2015) Spectral mle: Top-k rank aggregation from pairwise comparisons. In: International Conference on Machine Learning, pp 371–380

Chu W, Ghahramani Z (2005) Preference learning with Gaussian processes. In: Proceedings of the 22nd International Conference on Machine learning, ACM, pp 137–144

Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805

Felt P, Ringger EK, Seppi KD (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

Fu Y, Hospedales TM, Xiang T, Xiong J, Gong S, Wang Y, Yao Y (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, Springer, pp 65–82

Guo S, Sanner S, Bonilla EV (2010) Gaussian process preference elicitation. In: Advances in neural information processing systems, pp 262–270

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 (Volume 1: Long Papers), Association for Computational Linguistics, pp 1589–1599

Han B, Pan Y, Tsang IW (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 Online Library

Hensman J, Fusi N, Lawrence ND (2013) Gaussian processes for big data. In: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, AUAI Press, pp 282–290

Hensman J, Matthews AGdG, Ghahramani Z (2015) Scalable variational Gaussian process classification. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pp 351–360

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

Hoffman MD, Blei DM, Wang C, Paisley JW (2013) Stochastic variational inference. Journal of Machine Learning Research 14(1):1303–1347

Houlsby N, Huszar F, Ghahramani Z, Hernández-Lobato JM (2012) Collaborative Gaussian processes for preference learning. In: Advances in Neural Information Processing Systems, pp 2096–2104

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: In IEEE International Conference on Data Mining (ICDM 2008, Citeseer

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

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, ACM, pp 583–588

Kendall MG (1948) Rank correlation methods. Griffin

Khan ME, Ko YJ, Seeger M (2014) Scalable Collaborative Bayesian Preference Learning. In: Kaski S, Corander J (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, URL `http://proceedings.mlr.press/v33/khan14.html`

Kim Y, Kim W, Shim K (2014) Latent ranking analysis using pairwise comparisons. In: Data Mining (ICDM), 2014 IEEE International Conference on, IEEE, pp 869–874

Kingsley DC, Brown TC (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 (Volume 2: Short Papers), vol 2, pp 465–470

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

Lawrence ND, Urtasun R (2009) Non-linear matrix factorization with gaussian processes. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 601–608

Lee SM, Roberts SJ (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

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

Luce RD (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 DJ (1995) Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks. Network: computation in neural systems 6(3):469–505

Marley AAJ, Louviere JJ (2005) Some probabilistic models of best, worst, and best–worst choices. Journal of Mathematical Psychology 49(6):464–480, DOI 10.1016/j.jmp.2005.05.003

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119

Minka TP (2001) Expectation propagation for approximate Bayesian inference. In: Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., pp 362–369

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, ACM, pp 677–685

Mosteller F (2006) Remarks on the method of paired comparisons: I. The least squares solution assuming equal standard deviations and equal correlations. In: Selected Papers of Frederick Mosteller, Springer, pp 157–162

Naish-guzman A, Holden S (2008) The generalized fitc approximation. In: Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in Neural Information Processing Systems 20, Curran Associates, Inc., pp 1057–1064, URL `http://papers.nips.cc/paper/`

`3351-the-generalized-fitc-approximation.pdf`

Nguyen TV, Bonilla EV (2014) Collaborative multi-output gaussian processes. In: Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, AUAI Press, pp 643–652

Nickisch H, Rasmussen CE (2008) Approximations for binary Gaussian process classification. Journal of Machine Learning Research 9(Oct):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 IW (2018) Stagewise learning for noisy k-ary preferences. Machine Learning 107(8-10):1333–1361

Plackett RL (1975) The analysis of permutations. Applied Statistics pp 193–202

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 CE, Williams CKI (2006) Gaussian processes for machine learning. The MIT Press, Cambridge, MA, USA 38:715–719

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, IEEE, pp 1–8

Resnick P, Varian HR (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-Volume 1521, CEUR-WS. org, pp 43–54

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

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

Shah N, Balakrishnan S, Bradley J, Parekh A, Ramchandran K, Wainwright M (2015) Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. In: Artificial Intelligence and Statistics, pp 856–865

Simpson E, Gurevych I (2018) Finding convincing arguments using scalable bayesian preference learning. Transactions of the Association for Computational Linguistics 6:357–371

Simpson E, Reece S, Roberts SJ (2017) Bayesian heatmaps: probabilistic classification with multiple unreliable information sources. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 109–125

Simpson ED, Venanzi M, Reece S, Kohli P, Guiver J, Roberts SJ, Jennings NR (2015) Language understanding in the wild: Combining crowdsourcing and machine learning. In: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp 992–1002

Snelson E, Ghahramani Z (2006) Sparse Gaussian processes using pseudo-inputs. In: Advances in neural information processing systems, pp 1257–1264

Snow R, O'Connor B, Jurafsky D, Ng AY (2008) Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics, pp 254–263

Steinberg DM, Bonilla EV (2014) Extended and unscented Gaussian processes. In: Advances in Neural Information Processing Systems, pp 1251–1259

Thurstone LL (1927) A law of comparative judgment. Psychological review 34(4):273

Uchida S, Yamamoto T, Kato MP, Ohshima H, Tanaka K (2017) Entity ranking by learning and inferring pairwise preferences from user reviews. In: Asia Information Retrieval Symposium, Springer, pp 141–153

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

Wang X, Wang J, Jie L, Zhai C, Chang Y (2016) Blind men and the elephant: Thurstonian pairwise preference for ranking in crowdsourcing. In: Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, pp 509–518

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

Yannakakis GN, Hallam J (2011) Ranking vs. preference: a comparative study of self-reporting. In: International Conference on Affective Computing and Intelligent Interaction, Springer, pp 437–446

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

A Variational Lower Bound for GPPL

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

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \alpha_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)ds}{\int\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}'; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)dsdf'}. \tag{28}$$

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

$$\mathcal{L} = \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}\left[\ln p\left(y_p | f(\boldsymbol{x}_{a_p}), f(\boldsymbol{x}_{b_p})\right)\right] + \mathbb{E}_{q(\boldsymbol{f}),q(s)}\left[\ln \frac{p\left(\boldsymbol{f}|\boldsymbol{0}, \frac{\boldsymbol{K}}{s}\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_{q(s)}\left[\ln \frac{p\left(s|\alpha_0, \beta_0\right)}{q\left(s\right)}\right] \tag{29}$$

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

$$\begin{aligned}
\mathcal{L} = & \, \mathbb{E}_{q(\boldsymbol{f})}\left[\sum_{p=1}^{P} y_p \ln \Phi(z_p) + (1-y_p)\left(1 - \ln \Phi(z_p)\right)\right] + \mathbb{E}_{q(\boldsymbol{f})}\left[\ln \mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\mathbb{E}[s]\right)\right] \\
& - \mathbb{E}_{q(\boldsymbol{f}}\left[\ln \mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right)\right] + \mathbb{E}_{q(s)}\left[\ln \mathcal{G}\left(s; \alpha_0, \beta_0\right) - \ln \mathcal{G}\left(s; \alpha, \beta\right)\right] \\
= & \sum_{p=1}^{P} y_p \mathbb{E}_{q(\boldsymbol{f})}[\ln \Phi(z_p)] + (1-y_p)\left(1 - \mathbb{E}_{q(\boldsymbol{f})}[\ln \Phi(z_p)]\right)\Bigg] \\
& - \frac{1}{2}\left\{\ln |\boldsymbol{K}| - \mathbb{E}[\ln s] + \mathrm{tr}\left(\left(\hat{\boldsymbol{f}}^T \hat{\boldsymbol{f}} + \boldsymbol{C}\right)\boldsymbol{K}^{-1}\right) - \ln|\boldsymbol{C}| - N\right\} \\
& - \Gamma(\alpha_0) + \alpha_0(\ln \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\ln s] + \Gamma(\alpha) + (\beta - \beta_0)\mathbb{E}[s] - \alpha \ln \beta. \qquad (30) \\
& \qquad (31)
\end{aligned}$$

The expectation over the likelihood can be computed using numerical integration. Now we can introduce the sparse approximation to obtain the bound in Equation 17:

$$\begin{aligned}
\mathcal{L} \approx & \, \mathbb{E}_{q(\boldsymbol{f})}[\ln p(\boldsymbol{y}|\boldsymbol{f})] + \mathbb{E}_{q(\boldsymbol{f}_m),q(s)}[\ln p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0)] - \mathbb{E}_{q(\boldsymbol{f}_m)}[\ln q(\boldsymbol{f}_m)] - \mathbb{E}_{q(s)}[\ln q(s)] \\
= & \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\ln p(y_p | f(\boldsymbol{x}_{a_p}), f(\boldsymbol{x}_{b_p}))] - \frac{1}{2}\Bigg\{ \ln|\boldsymbol{K}_{mm}| - \mathbb{E}[\ln s] - \ln|\boldsymbol{S}| - M \\
& + \hat{\boldsymbol{f}}_m^T \mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\Bigg\} + \ln \Gamma(\alpha) - \ln \Gamma(\alpha_0) + \alpha_0(\ln \beta_0) \\
& + (\alpha_0 - \alpha)\mathbb{E}[\ln s] + (\beta - \beta_0)\mathbb{E}[s] - \alpha \ln \beta, \qquad (32)
\end{aligned}$$

where the terms relating to $\mathbb{E}[p(\boldsymbol{f}|\boldsymbol{f}_m) - q(\boldsymbol{f})]$ cancel. 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} \quad (33)$$

B 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:

$$\left(\boldsymbol{S}_{c,i}^{(v)}\right)^{-1} = (1-\rho_i)\left(\boldsymbol{S}_{c,i-1}^{(v)}\right)^{-1} + \rho_i\left(\boldsymbol{K}_{mm}^{-1}\mathbb{E}\left[s_c^{(v)}\right] + \pi_i \boldsymbol{A}_{v,i}^T \boldsymbol{G}_i^T \text{diag}\left(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}}\right) \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_{v,i}\right), \quad (34)$$

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 Equations 40 and 39), and $\boldsymbol{u} = \{u_p \forall p \in P_i\}$ 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 $\boldsymbol{V}_m$:

$$\begin{aligned}\hat{\boldsymbol{v}}_{m,c,i} = \boldsymbol{S}_{c,i}^{(v)} \Bigg((1-\rho_i)\left(\boldsymbol{S}_{c,i-1}^{(v)}\right)^{-1} \hat{\boldsymbol{v}}_{m,c,i-1} + \rho_i \pi_i \Bigg(\boldsymbol{S}_{c,i}^{(v)} \boldsymbol{A}_i^T \boldsymbol{G}_i^T \text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}) \boldsymbol{Q}_i^{-1} \\
\Big(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_j^{(i)} (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j})\Big)\Bigg)\Bigg),\end{aligned} \quad (35)$$

where $\boldsymbol{H}_j^{(i)} \in |P_i| \times N$ contains partial derivatives of the pairwise likelihood with respect to $F_{n,j} = \hat{v}_{c,n}\hat{w}_{c,j}$, with elements given by:

$$H_{j,p,n}^{(i)} = \Phi(\mathbb{E}[z_p])(1-\Phi(\mathbb{E}[z_p]))(2y_p - 1)([n = a_p] - [n = b_p])[j = u_p]. \quad (36)$$

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

$$\left(\boldsymbol{S}_i^{(t)}\right)^{-1} = (1-\rho_i)\left(\boldsymbol{S}_{i-1}^{(t)}\right) + \rho_i \boldsymbol{K}_{mm}^{-1}\mathbb{E}\left[s^{(t)}\right] + \rho_i \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i \quad (37)$$

$$\hat{\boldsymbol{t}}_{m,i} = \boldsymbol{S}_i^{(t)} \left((1-\rho_i)\left(\boldsymbol{S}_{i-1}^{(t)}\right)^{-1} \hat{\boldsymbol{t}}_{m,i-1} + \rho_i \pi_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{t}}_i\right)\right). \quad (38)$$

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

$$\boldsymbol{\Sigma}_{c,i}^{-1} = (1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}^{-1} + \rho_i \boldsymbol{L}_{mm}^{-1}\mathbb{E}\left[s_c^{(w)}\right] + \rho_i \pi_i \boldsymbol{A}_{w,i}^T \Bigg(\sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)T} \mathrm{diag}\left(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 \right.$$
$$\left. + \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}}^{(v)} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}}^{(v)} - 2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}\hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}}^{(v)}\right) \boldsymbol{Q}_i^{-1} \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \Bigg) \boldsymbol{A}_{w,i} \tag{39}$$

$$\hat{\boldsymbol{w}}_{m,c,i} = \boldsymbol{\Sigma}_{c,i} \Bigg((1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}\hat{\boldsymbol{w}}_{m,c,i-1} + \rho_i \pi_i \boldsymbol{A}_{w,i}^T \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \left(\mathrm{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}})\right.$$
$$\left. - \mathrm{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\right) \boldsymbol{Q}_i^{-1} \Bigg(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_u^{(i)} (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \Bigg) \Bigg), \tag{40}$$

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, and $\boldsymbol{A}_{w,i} = \boldsymbol{L}_{im}\boldsymbol{L}_{mm}^{-1}$.

```
   Input: Pairwise labels, $y$, training item features, $x$, training user features $u$, test
          item features $x^*$, test user features $u^*$
 1 Compute kernel matrices $K$, $K_{mm}$ and $K_{nm}$ given $x$;
 2 Compute kernel matrices $L$, $L_{mm}$ and $L_{nm}$ given $u$;
 3 Initialise $\mathbb{E}[s^{(t)}]$, $\mathbb{E}[s_c^{(v)}]\ \forall c$, $\mathbb{E}[s_c^{(w)}]\ \forall c$, $\mathbb{E}[V]$, $\hat{V}_m$, $\mathbb{E}[W]$, $\hat{W}_m$, $\mathbb{E}[t]$, $\hat{t}_m$ to prior
    means;
 4 Initialise $S_{v,c} \forall c$ and $S_t$ to prior covariance $K_{mm}$;
 5 Initialise $S_{w,c} \forall c$ to prior covariance $L_{mm}$;
   while $\mathcal{L}$ not converged do
 6     Select random sample, $P_i$, of $P$ observations;
       while $G_i$ not converged do
 7         Compute $G_i$ given $\mathbb{E}[F_i]$ ;
 8         Compute $\hat{t}_{m,i}$ and $S_i^{(t)}$ ;
           for c in 1,...,C do
 9             Update $\mathbb{E}[F_i]$ ;
10             Compute $\hat{v}_{m,c,i}$ and $S_{i,c}^{(v)}$ ;
11             Update $q\left(s_c^{(v)}\right)$, compute $\mathbb{E}\left[s_c^{(v)}\right]$ and $\mathbb{E}\left[\ln s_c^{(v)}\right]$;
12             Update $\mathbb{E}[F_i]$ ;
13             Compute $\hat{W}_{m,c,i}$ and $\Sigma_{i,c}$ ;
14             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
15         Update $\mathbb{E}[F_i]$ ;
       end
16     Update $q\left(s^{(t)}\right)$, compute $\mathbb{E}\left[s^{(t)}\right]$ and $\mathbb{E}\left[\ln s^{(t)}\right]$ ;
   end
17 Compute kernel matrices for test items, $K_{**}$ and $K_{*m}$, given $x^*$ ;
18 Compute kernel matrices for test users, $L_{**}$ and $L_{*m}$, given $u^*$ ;
19 Use converged values of $\mathbb{E}[F]$ and $\hat{F}_m$ to estimate posterior over $F^*$ at test points ;
   Output: Posterior mean of the test values, $\mathbb{E}[F^*]$ and covariance, $C^*$
```

Algorithm 2: The SVI algorithm for crowdGPPL.

C Predictions with CrowdGPPL

The means, item covariances and user variance required for predictions with crowdGPPL (Equation 26) are defined as follows:

$$\hat{\boldsymbol{t}}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{t}}_m, \qquad \boldsymbol{C}^{(t)*} = \frac{\boldsymbol{K}_{**}}{\mathbb{E}\left[s^{(t)}\right]} + \boldsymbol{A}_{*m}\left(\boldsymbol{S}^{(t)} - \boldsymbol{K}_{mm}\right)\boldsymbol{A}_{*m}^T, \tag{41}$$

$$\hat{\boldsymbol{v}}_c^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{v}}_{m,c}, \qquad \boldsymbol{C}_c^{(v)*} = \frac{\boldsymbol{K}_{**}}{\mathbb{E}\left[s_c^{(v)}\right]} + \boldsymbol{A}_{*m}\left(\boldsymbol{S}_c^{(v)} - \boldsymbol{K}_{mm}\right)\boldsymbol{A}_{*m}^T \tag{42}$$

$$\hat{\boldsymbol{w}}_c^* = \boldsymbol{L}_{*m}\boldsymbol{L}_{mm}^{-1}\hat{\boldsymbol{w}}_{m,c}, \qquad \omega_{c,u}^* = 1/\mathbb{E}\left[s_c^{(w)}\right] + \boldsymbol{A}_{um}^{(w)}(\boldsymbol{\Sigma}_{w,c} - \boldsymbol{L}_{mm})\boldsymbol{A}_{um}^{(w)T} \tag{43}$$

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.

D Mathematical Notation

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
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

Table 4 Table of symbols used to represent variables in this paper (continued on next page in Table 5).

Symbol	Meaning
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
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
π_i	weight given to the update at the ith iteration using a subsample of the data
$\|P_i\|$	number of pairwise labels in the ith iteration subsample

Table 5 Table of symbols used to represent variables in this paper (continued from Table 4 on previous page).