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

Scalable Bayesian Preference Learning with Crowds

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 inferring a consensus from crowdsourced pairwise labels and predicting the preferences of individual annotators. Annotators in a crowd may have divergent opinions, making it difficult to identify consensus rankings or ratings from pairwise labels. Limited, noisy data for each user also presents a challenge when predicting the preferences of individuals. We address these challenges by combining matrix factorization with Gaussian processes in a Bayesian approach that accounts for uncertainty arising from sparse data and annotation noise. Previous methods for Gaussian process preference learning (GPPL) do not scale to datasets with large numbers of users, items or pairwise labels, so we propose an inference method using stochastic variational inference (SVI) that can handle constraints on memory and computational costs. Our experiments on a computational argumentation task demonstrate the method's scalability and show that modeling the preferences of individual annotators in a crowd improves the quality of the inferred consensus. On a benchmark recommendation task, our method is competitive with previous approaches, while the model is able to scale to far larger datasets. We make our software and documentation publicly available for use in 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). The goal may be to rank items by preference, or given a pair of items, to select the item with the highest utility. For many preference learning tasks, annotators have divergent opinions on the correct rankings or utilities, which impedes the acquisition of training data. For example, in the field of argument mining, one goal is to identify convincing arguments from a corpus of documents (Habernal and Gurevych 2016). Whether a particular argument is convincing or not depends

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

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

on the reader's point of view and prior knowledge (Lukin et al. 2017). Similarly, recommender systems can perform better if they make recommendations tailored to a specific user (Resnick and Varian 1997). Crowdsourcing is frequently used as a cost-effective source of labeled data, yet disagreements between annotators must be resolved to obtain a gold-stanard training set, typically requiring redundant labeling and increased annotation costs (Snow et al. 2008; Banerji et al. 2010; Gaunt et al. 2016). Therefore, we require solutions for predicting individual preferences or producing a gold standard from crowdsourced data that take into account the differences of opinion between annotators and users.

A preference model can be trained using numerical ratings, yet 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 -1, say, from one annotator may be equivalent to a -20 from another. An alternative is *pairwise labeling*, in which the annotator compares pairs of items and selects the preferred one in each pair. Making pairwise choices places lower cognitive load on annotators than numerical ratings (Yang and Chen 2011), and facilitates the total sorting of items, since it avoids cases where two items have the same value (Kendall 1948; Kingsley and Brown 2010). Besides explicit annotations, pairwise preference labels can be extracted from user behavior logs, such as when a user selects one item from a list in preference to others (Joachims 2002). However, thess implicit annotations can be noisy, as are crowdsourced pairwise labels (Habernal and Gurevych 2016). Therefore, while pairwise labels provide a valuable form of training data, preference learning methods must be able to aggregate noisy sources of pairwise data, such as that obtained from crowds. We henceforth refer to the annotators, users or implicit data sources simply as *users* whose preferences we wish to model. Likewise, the term *items* refers to any type of instance that users may express preferences over, and could be states or actions as well as physical objects.

In recommender systems, information from different users is aggregated using *collaborative filtering*, which predicts a user's preferences for an item they have not annotated using the observed preferences of similar users for that item (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. Multiplying the low-dimensional representations predicts ratings for unseen user-item pairs. However, traditional approaches are not able to handle pairwise labels, nor extrapolate effectively to new users or items.

The limited amount of noisy data for each user, as well as the desire to aggregate data from multiple users, motivates a Bayesian approach that can account for uncertainty in the model. Matrix factorization, has been shown to benefit from a Bayesian treatment when data is sparse (Salakhutdinov and Mnih 2008). Previous work has also introduced a Bayesian approach for combining crowdsourced preferences, but this models only ground truth rather than individual preferences and cannot make predictions for new users or items (Chen et al. 2013). For crowdsourced classification tasks, Simpson et al. (2017) show that modeling item features using a Gaussian process can improve performance, but this has not yet been adapted for preference learning. Gaussian processes have also been used in previous work to provide a Bayesian framework for preference learning (Chu and

Ghahramani 2005; Houlsby et al. 2012; Khan et al. 2014), but these methods do not scale to large numbers of items, users, or pairwise labels as it is unclear how to parallelize them or constrain their memory requirements.

In this paper, we propose a scalable Bayesian approach to pairwise preference learning with crowds, whose members may be annotators or users. Our approach is designed for preference learning over items, rather than over labels (see discussion in Fürnkranz and Hüllermeier (2010)), and jointly models personal preferences and the consensus of a crowd by combining matrix factorization and Gaussian processes. To enable inference at the scale of large, real-world datasets, we derive a stochastic variational inference scheme (Hoffman et al. 2013) for our approach. By providing a scalable Bayesian approach we open preference learning up to novel applications for which annotations are sparse, noisy and biased, and where the number of users, items and pairwise labels is large. Our empirical evaluation demonstrates the scalability of our approach, and its ability to predict both personal preferences and an objective gold-standard given crowdsourced data. We also improve performance over the previous state-of-the-art Simpson and Gurevych (2018) on a crowdsoured argumentation dataset.

The next section of the paper discusses related work. We then we 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 several real-world datasets.

2 Related Work

2.1 Aggregating Pairwise Labels from a Crowd

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 (Thurstone 1927). An example is the method of Herbrich et al. (2007), which learns the skill of chess players from match outcomes by treating them as noisy pairwise labels. Recent work on this type of approach has analyzed bounds on error rates (Chen and Suh 2015), sample complexity (Shah et al. 2015), and joint models for ranking and clustering from pairwise comparisons (Li et al. 2018).

The problem of disagreement between annotators in a crowd was addressed by Chen et al. (2013) and Wang et al. (2016), using Bayesian approaches that learn the individual accuracy of each worker. However, they do not exploit item features to mitigate data sparsity. In contrast, Gaussian processes preference learning (*GPPL*) uses item features to make predictions for unseen items and share information between similar items (Chu and Ghahramani 2005). GPPL was used by Simpson and Gurevych (2018) to aggregate crowdsourced pairwise labels, but assumes the same level of noise for all annotators and ignores the effect of divergent opinions. To crowdsource sentiment annotations, Kiritchenko and Mohammad (2016) propose to use an extension of pairwise comparison known as *best-worst scaling*, in which the annotator selects best and worst items from a set. They apply a simple counting technique to infer a ranking over the items, which requires each item to have a sufficient number of comparisons.

A popular method for predicting pairwise labels of new items given their features is *SVM-rank* Joachims (2002). For crowdsourced data, Fu et al. (2016) show that performance is improved by identifying outliers in crowdsourced data that correspond to probable errors. Uchida et al. (2017) extend SVM-rank to account for different levels of confidence in each pairwise annotation expressed by the annotators. However, besides modeling labeling noise, the previous work on crowdsourced preference learning does not account for the effect of divergence of opinion on the inferred preferences.

2.2 Bayesian Methods for Inferring Individual Preferences

As well as aggregating preferences from a crowd to identify a consensus, we also wish to predict the preferences of individual users. Yi et al. (2013) and Kim et al. (2014) address this task by learning multiple latent rankings and inferring the preference of each user toward those rankings, while Salimans et al. (2012) use Bayesian matrix factorization to identify multiple latent rankings. However, none of these approaches exploit item features to remedy labeling errors or generalize to new test items. In contrast, Guo et al. (2010) 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, and combine them with matrix factorization to perform collaborative filtering. However, this approach does not model the relationship between input features and the latent factors, does not place a prior over item factors, and does not scale to very large sets of users. An alternative is *Collaborative GPPL* (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 users and items when learning the latent factors. Each individual's preferences are then represented by a mixture of latent functions. Using matrix factorization in combination with GP priors is therefore an effective way to model the individual preferences of users while making use of input features to generalize to test cases. However, none of these approaches considers a consensus preference function, and more scalable inference is needed for datasets containing thousands of items or users.

2.3 Scalable Approximate Bayesian Inference

Recent work on scalable Bayesian matrix factorization focuses on distributing and parallelizing inference but is not directly applicable to Gaussian processes (Ahn 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 items, users or pairs, as there is no clear way to parallelize it or to limit memory consumption. The GP over pairs also makes it difficult to extract posteriors for latent function values for individual items, and prevents mixing pairwise training labels with observed ratings in future data aggregation settings.

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 using a different random sample 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), and provides a convenient framework for sparse approximation. An SVI method was also developed for preference learning, which places a GP over items rather than pairs, but does not model multiple users' preferences (Simpson and Gurevych 2018). This paper provides the first full derivation of this approach, as well as providing the first application of SVI to Bayesian matrix factorization as part of a new model that accounts for individual user preferences.

add in table of notations (maybe in the appendix)

3 Bayesian Preference Learning for Crowds

For a pair of items, a and b, we write $a \succ b$ to symbolize that a is preferred to b. A pairwise label has value $y(a,b) = 1$ if $a \succ b$ and 0 otherwise. Thurstone (1927) proposed that the likelihood of pairwise label $y(a,b) = 1$ increases as the difference between the utilities of a and b increases. The uncertainty in the likelihood $p(y(a,b) = 1)$ accommodates labeling errors as well as variability in a user's judgments. Here, we assume the utility to be a function of the items' features, $f(\boldsymbol{x}_a) : \mathbb{R}^D \mapsto \mathbb{R}$, where $\boldsymbol{x}_a$ is a vector representation of the features of item a of length D. Pairwise labels can be modeled using a logistic likelihood defined by the Bradley-Terry model (Bradley and Terry 1952; Plackett 1975; Luce 1959), or a probit likelihood given by the Thurstone-Mosteller model, also known as *Thurstone case V* (Thurstone 1927; Mosteller 2006). We use the latter for our model as it enables f to be marginalized analytically to compute a posterior $p(y(a,b) = 1|\boldsymbol{y})$, where $\boldsymbol{y}$ is a set of pairwise training labels. Noise in the observations is explained by a Gaussian-distributed noise term, $\delta \sim \mathcal{N}(0, 0.5)$:

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

Integrating out the unknown values of δ_a and δ_b gives:

$$\begin{aligned} & p(y(a,b)|f(\boldsymbol{x}_a), f(\boldsymbol{x}_b),) \\ & = \iint p(y(a,b)|f(\boldsymbol{x}_a)+\delta_a, f(\boldsymbol{x}_b)+\delta_b)\mathcal{N}(\delta_a;0,0.5)\mathcal{N}(\delta_b;0,0.5)d\delta_a d\delta_b \\ & = \Phi\left(f(\boldsymbol{x}_a)-f(\boldsymbol{x}_b)\right) = \Phi\left(z\right), \end{aligned} \tag{2}$$

where $z = f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b)$, and Φ is the cumulative distribution function of the standard normal distribution. This likelihood is the basis of Gaussian process preference learning (GPPL) (Chu and Ghahramani 2005). Our formulation differs in that instead of learning the variance of δ, we fix it to 0.5 and scale f to vary the uncertainty in the pairwise labels, which conveniently leads to z having a denominator of 1. If we have distributions over $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$, they can be marginalized in a simple manner by modifying z:

$$\hat{z} = \frac{\mu_a - \mu_b}{\sqrt{1+\sigma_a+\sigma_b-\sigma_{a,b}}} \tag{3}$$

where μ_a and μ_b are the expected values of $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$ respectively, σ_a and σ_b are the corresponding variances, and $\sigma_{a,b}$ is the covariance between $f(\boldsymbol{x}_a)$ and $f(\boldsymbol{x}_b)$. The next section introduces a model for the utility function, f.

3.1 Single User Preference Learning

We assume a Gaussian process prior over the utility function, $f \sim \mathcal{GP}(0, k_\theta/s)$, where k_θ is a kernel function with hyper-parameters θ, and s is an inverse scale drawn from a gamma prior, $s \sim \mathcal{G}(\alpha_0, \beta_0)$, with shape α_0 and scale β_0. The value of s determines the variance of f and therefore its magnitude, which affects the level of certainty in the pairwise label likelihood (Equation 2). 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 choices of kernel function include the *squared exponential* or *Matérn* functions (Rasmussen and Williams 2006), which produce higher covariance between items with similar feature values. These kernel functions make minimal assumptions and are effective in a wide range of tasks. We use the kernel function k_θ to compute the covariance matrix K_θ, between a set of N observed items with features $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$.

Given a set of P pairwise labels, $\boldsymbol{y} = \{y_1, ..., y_P\}$, where $y_p = y(a_p \succ b_p)$, we can write the joint distribution over all variables as follows:

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

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

3.2 Crowd Preference Learning

When there are multiple users, jointly modeling their different preferences may improve consensus predictions and improve our predictions for each user when faced with sparse data. We represent utilities in a matrix, $\boldsymbol{F} \in \mathbb{R}^{N \times U}$, where N rows correspond to items and U columns correspond to users. If we factorize this matrix, we obtain two low-dimensional matrices, one for users, $\boldsymbol{W} \in \mathbb{R}^{C \times U}$, and one for the items, $\boldsymbol{V} \in \mathbb{R}^{C \times N}$, where C is the number of latent components: $\boldsymbol{F} = \boldsymbol{V}^T \boldsymbol{W}$. 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 a user j, respectively. Latent components correspond to utility functions for certain items shared by multiple users. These could represent, for example, in the case of book recommendation, interests in a particular genre of book. For the multi-user case, we assume that there are C latent functions of item features, $v_c \sim \mathcal{GP}(\mathbf{0}, k_\theta / s_c)$, and C latent functions of user features, $w_c \sim \mathcal{GP}(\mathbf{0}, k_\eta)$, where k is a kernel function, s_c is an inverse scale for the item features, θ are the kernel hyperparameters for the items and η are the kernel hyperparameters for the users. The matrices V and W are evaluations of these functions at the points corresponding to the users and items observed during training. Since our goal is to infer a consensus from a crowd as well as to model individual users' preferences, we introduce a consensus utility function over item features, $t \sim \mathcal{GP}(\mathbf{0}, k_\theta / \sigma_t)$, that is shared across all users, with values $\boldsymbol{t} = \{t(\boldsymbol{x}_1), ..., t(\boldsymbol{x}_N)\}$ for the training items. Therefore, the latent preference function, f, is a weighted sum over latent functions:

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

where $\boldsymbol{u}_j$ are the features of user j and $\boldsymbol{x}_a$ are the features of item a. For simplicity, we assume fixed values of C in this paper. The Bayesian approach avoids overfitting by inferring $s_c \approx 0$ with high probability for any dimensions that are not required to model the data.

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

$$p\left(\boldsymbol{y}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{t}, s_1, ..., s_C, \sigma_t | k_\theta, k_\eta, \alpha_0, \beta_0\right) = \prod_{p=1}^{P} \Phi\left(z_p\right) \mathcal{N}(\boldsymbol{t}; \mathbf{0}, \boldsymbol{K}_\theta / \sigma_t)$$
$$\mathcal{G}(\sigma_t; \alpha_0, \beta_0) \prod_{c=1}^{C} \left\{ \mathcal{N}(\boldsymbol{v}_c; \mathbf{0}, \boldsymbol{K}_\theta / s_c) \mathcal{N}(\boldsymbol{w}_c; \mathbf{0}, \boldsymbol{K}_{w,\eta}) \mathcal{G}(s_c; \alpha_0, \beta_0) \right\}, \tag{6}$$

where $z_p = \boldsymbol{v}_{.,a_p}^T \boldsymbol{w}_{.,u_p} + t_{a_p} - \boldsymbol{v}_{.,b_p}^T \boldsymbol{w}_{.,u_p} - t_{b_p}$, σ_t is the inverse scale of t, and index p refers to a tuple, $\{u_p, a_p, b_p\}$, which identifies the user and a pair of items.

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. 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 unlike VB, convergence is not guaranteed (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 so as to give a good approximation, using either heuristics or optimizing their positions to maximize the approximate marginal likelihood. The sparse approximation used by Houlsby et al. (2012) for the collaborative GP is the *generalized fully independent training conditional* (GFITC) (Snelson and Ghahramani 2006). In practice, GFITC is unsuitable for datasets with more than a few thousands points, as the computational and memory costs cannot be constrained and GFITC is not amenable to distributed computation (Hensman et al. 2015). We 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). First, we define an approximate likelihood that enables the SVI method.

4.1 Approximate Pairwise Likelihood

To obtain a tracatable approximate posterior, we begin by approximating the expected preference likelihood (Equation 2) with a Gaussian:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathbb{E}\left[\prod_{p=1}^{P} \Phi(z_p)\right] = \prod_{p=1}^{P} \Phi(\hat{z}_p) \approx \mathcal{N}(\boldsymbol{y}; \Phi(\hat{\boldsymbol{z}}), \boldsymbol{Q}), \tag{7}$$

where $\hat{\boldsymbol{z}} = \{\hat{z}_1, ..., \hat{z}_P\}$ and $\boldsymbol{Q}$ is a diagonal noise covariance matrix. Since $\Phi(\hat{z}_p)$ defines a bernoulli distribution, for which the conjugate prior is a beta distribution, we moment match the diagonal entries of $\boldsymbol{Q}$ to the expected variance of a bernoulli distribution as follows:

$$Q_{p,p} = \mathbb{E}_f[\Phi(z_p)(1 - \Phi(z_p))] \approx \frac{(y_p + \gamma)(1 - y_p + \lambda)}{(\gamma + \lambda + 1)} - \frac{(y_p + \gamma)(1 - y_p + \lambda)}{(\gamma + \lambda + 1)^2(\gamma + \lambda + 2)}, \tag{8}$$

where γ and λ are parameters of a beta distribution with the same variance as the prior, $p(\Phi(z_p)|\boldsymbol{K}_\theta, \alpha_0, \beta_0)$, and are estimated using numerical integration.

Unfortunately, the nonlinear term, $\Phi(\boldsymbol{z})$ means that the posterior is still intractable, so we linearize $\Phi(\boldsymbol{z})$ by taking its first-order Taylor series expansion about the expectation of $\boldsymbol{f}$ for GPPL (for crowdGPPL, replace $\boldsymbol{f}$ with $\boldsymbol{F}$ in the following):

$$\Phi(\boldsymbol{z}) \approx \tilde{\Phi}(\boldsymbol{z}) = \boldsymbol{G}(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}]) + \Phi(\hat{\boldsymbol{z}}), \tag{9}$$

$$G_{p,i} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\hat{z}_p))(2y_p - 1)([i = a_p] - [i = b_p]) \tag{10}$$

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 $\boldsymbol{f}$ and $\boldsymbol{G}$, which can be estimated iteratively using variational inference (Steinberg and Bonilla 2014), as we describe below. Linearization makes the approximate likelihood conjugate to the prior, $\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)$, so that the approximate posterior over $\boldsymbol{f}$ is also Gaussian. The Gaussian likelihood approximation and linearization also appear in GP inference methods based on expectation propagation (Rasmussen and Williams 2006) and the extended Kalman filter (Reece et al. 2011; Steinberg and Bonilla 2014). We can now use the approximate likelihood to derive equations for SVI.

4.2 SVI for Single User GPPL

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. To do this, we introduce 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. The posterior over the inducing and training items is approximated as:

$$p(\boldsymbol{f}, \boldsymbol{f}_m, s|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}_m, k_\theta, \alpha_0, \beta_0) \approx q(\boldsymbol{f}, \boldsymbol{f}_m, s) = q(s)q(\boldsymbol{f})q(\boldsymbol{f}_m). \tag{11}$$

We marginalize $\boldsymbol{f}$ to obtain the factor for $\boldsymbol{f}_m$:

$$\begin{aligned}\log q(\boldsymbol{f}_m) &= \log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}\right)] + \log \mathcal{N}\left(\boldsymbol{f}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}\left[s\right]\right) + \text{const}, \\ &= \log \mathcal{N}(\boldsymbol{f}_m; \hat{\boldsymbol{f}}_m, \boldsymbol{S}),\end{aligned} \tag{12}$$

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:

$$\log q(\boldsymbol{f}) = \log \mathcal{N}(\boldsymbol{f}; \boldsymbol{A}\hat{\boldsymbol{f}}_m, \boldsymbol{K} + \boldsymbol{A}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{A}^T), \tag{13}$$

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:

$$\log q(s) = \log \mathcal{N}(\mathbb{E}[\boldsymbol{f}_m|\boldsymbol{0}, \boldsymbol{K}_{mm}/s] + \log \mathcal{G}(s; \alpha_0, \beta_0) + \text{const} = \log \mathcal{G}(s; \alpha, \beta), \tag{14}$$

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 location of inducing points 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 $K = M$ to cluster the feature vectors, then taking the cluster centers as inducing points that represent the spread of observations across feature space.

We can apply variational inference to iteratively reduce the KL-divergence between our approximate posterior and the true posterior (both stated in Equation 11) by maximizing a lower bound, $\mathcal{L}$, on the log marginal likelihood (see also the detailed equations in Appendix A):

$$\log p(\boldsymbol{y}|\boldsymbol{K}, \alpha_0, \beta_0) = \text{KL}(q(\boldsymbol{f}, \boldsymbol{f}_m, s)||p(\boldsymbol{f}, \boldsymbol{f}_m, s|\boldsymbol{y}, \boldsymbol{K}, \alpha_0, \beta_0)) + \mathcal{L} \tag{15}$$
$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{f}, \boldsymbol{f}_m, s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)].$$

We optimize $\mathcal{L}$ by initializing the q factors randomly, then updating 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 observations $\boldsymbol{y}_i = \{y_p \forall p \in \boldsymbol{P}_i\}$, where $\boldsymbol{P}_i$ is random subset of indexes of observations. 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):

$$\boldsymbol{S}_i^{-1} = (1-\rho_i)\boldsymbol{S}_{i-1}^{-1} + \rho_i \left(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1} + w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i\right) \tag{16}$$

$$\hat{\boldsymbol{f}}_{m,i} = \boldsymbol{S}_i \left((1-\rho_i)\boldsymbol{S}_{i-1}^{-1}\hat{\boldsymbol{f}}_{m,i-1} + \rho_i w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{f}}_{m,i}\right)\right) \tag{17}$$

where $\rho_i = (i+\epsilon)^{-r}$ is a mixing coefficient that controls the update rate, $w_i = \frac{P}{|P_i|}$ weights each update according to sample size, ϵ is a delay hyperparameter and r is a forgetting rate (Hoffman et al. 2013). For the inverse scale, $q(s)$ is updated using Equation 14, then its expected value is computed as $\mathbb{E}[s] = \frac{\alpha}{\beta}$.

The complete SVI algorithm is summarized in Algorithm 1. The use of an inner loop to learn $\boldsymbol{G}_i$ avoids the need to store the complete matrix, $\boldsymbol{G}$. The inferred distribution over the inducing points can be used for predicting the values of test items, $f(\boldsymbol{x}^*)$:

$$\boldsymbol{f}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m, \qquad \boldsymbol{C}^* = \boldsymbol{K}_{**} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{K}_{*m}^T\boldsymbol{K}_{mm}^{-1}, \tag{18}$$

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.

Input: Pairwise labels, $\boldsymbol{y}$, training item features, $\boldsymbol{x}$, test item features $\boldsymbol{x}^*$
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**
  Select random sample, $\boldsymbol{P}_i$, of P observations **while** $\boldsymbol{G}_i$ *not converged* **do**
    Compute $\boldsymbol{G}_i$ given $\mathbb{E}[\boldsymbol{f}_i]$;
    Compute $\hat{\boldsymbol{f}}_{m,i}$ and $\boldsymbol{S}_i$;
    Compute $\mathbb{E}[\boldsymbol{f}_i]$;
  end
  Update $q(s)$ and compute $\mathbb{E}[s]$ and $\mathbb{E}[\log s]$;
end
Compute kernel matrices for test items, $\boldsymbol{K}_{**}$ and $\boldsymbol{K}_{*m}$, given $\boldsymbol{x}^*$;
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 preference learning with a single user.

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 in Equation 11 to approximate the crowd model defined in Equation 6:

$$p(\boldsymbol{V}, \boldsymbol{V}_m, \boldsymbol{W}, \boldsymbol{W}_m, \boldsymbol{t}, \boldsymbol{t}_m, s_1, ..., s_C, \sigma | \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{u}_m, k, \alpha_0, \beta_0) \approx$$
$$q(\boldsymbol{V})q(\boldsymbol{W})q(\boldsymbol{t})q(\boldsymbol{V}_m)q(\boldsymbol{W}_m)q(\boldsymbol{t}_m)\prod_{c=1}^{C} q(s_c)q(\sigma), \tag{19}$$

where $\boldsymbol{u}_m$ are the feature vectors of inducing users. By updating each of the q factors in turn, the SVI procedure optimizes the lower bound on the marginal likelihood (see also Equation 32 in the Appendix):

$$\mathcal{L}_{cr} = \mathbb{E}_q \Bigg[\log p(\boldsymbol{y}|\boldsymbol{F}) + \sum_{c=1}^{C} \big\{ \log p(\boldsymbol{v}_{m,c}, s_c | \boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \log q(\boldsymbol{v}_{m,c}) \tag{20}$$
$$+ \log p(\boldsymbol{w}_{m,c} | \boldsymbol{K}_{w,mm}) - \log q(\boldsymbol{w}_{m,c}) \big\} + \log p(\boldsymbol{t}_m, \sigma | \boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \log q(\boldsymbol{t}_m) \Big] .$$

The algorithm follows the same pattern as Algorithm 1, 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 inducing item factors is:

$$\log q(\boldsymbol{V}_m) = \mathbb{E}\left[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{v}_{m,c}; \boldsymbol{0}, \frac{\boldsymbol{K}_{v,mm}}{\mathbb{E}[s_c]}\right) + \text{const}$$
$$= \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{v}_{m,c}; \hat{\boldsymbol{v}}_{m,c}, \boldsymbol{S}_{v,c}), \tag{21}$$

where the posterior mean $\hat{\boldsymbol{v}}_{m,c}$ and covariance $\boldsymbol{S}_{v,c}$ are computed using update equations of the same form as those of the single user GPPL in Equations 16 and 17. For reasons of space, we provide the complete equations for $\hat{\boldsymbol{v}}_{m,c}$ and $\boldsymbol{S}_{v,c}$ in

Appendix B, Equations 33 and 34. The variational component for the inducing points of the common item mean follows a similar pattern:

$$\begin{aligned}\log q(\boldsymbol{t}_m) = & \ \mathbb{E}[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \mathbb{E}[\log \mathcal{N}(\boldsymbol{t}_m; \boldsymbol{0}, \boldsymbol{K}_{t,mm}/s)] + \text{const} \\ = & \ \log \mathcal{N}\left(\boldsymbol{t}; \hat{\boldsymbol{t}}, \boldsymbol{S}_t\right), \end{aligned} \tag{22}$$

where again, the posterior mean $\hat{\boldsymbol{t}}$ and covariance $\boldsymbol{S}_t$ use similar updates to Equation Equations 16 and 17, and their full definitions are given in Appendix B, Equations 36 and 37. Finally, the variational distribution for the inducing users factors is:

$$\begin{aligned}\log q(\boldsymbol{W}_m) = & \ \mathbb{E}\left[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \sum_{c=1}^{C} \mathbb{E}[\log \mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{K}_{w,mm})] + \text{const} \\ = & \ \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{w}_c; \hat{\boldsymbol{w}}_c, \boldsymbol{\Sigma}_c\right), \end{aligned} \tag{23}$$

where $\hat{\boldsymbol{w}}_c$ and $\boldsymbol{\Sigma}_c$ also follow the pattern of Equations 16 and 17, and their full definitions are given in Appendix B, Equations 36 and 37. The expectations for the inverse scales, $s_1, ..., s_c$ and σ, can be computed using Equation 14 by substituting in the corresponding terms for each $\boldsymbol{v}_c$ or $\boldsymbol{t}$ instead of $\boldsymbol{f}$.

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

$$\boldsymbol{F}^* = \hat{\boldsymbol{t}}^* + \sum_{c=1}^{C} \hat{\boldsymbol{v}}_c^{*T} \hat{\boldsymbol{w}}_c^*, \qquad \boldsymbol{\Lambda}_u^* = \boldsymbol{C}_t^* + \sum_{c=1}^{C} \omega_{c,u}^* \boldsymbol{C}_{v,c}^* + \hat{w}_{c,u}^2 \boldsymbol{C}_{v,c}^* + \omega_{c,u}^* \hat{\boldsymbol{v}}_c \hat{\boldsymbol{v}}_c^T, \tag{24}$$

where $\hat{\boldsymbol{t}}^*$, $\hat{\boldsymbol{v}}_c^*$ and $\hat{\boldsymbol{w}}_c^*$ are posterior means of the predictions, $\boldsymbol{C}_t^*$ and $\boldsymbol{C}_{v,c}^*$ are posterior item covariances of the predictions, and $\omega_{c,u}^*$ is the posterior variance for the user components for the predicted users. These terms are defined in Appendix C, Equations 40 to 42. The mean $\boldsymbol{F}^*$ and covariances Λ_u^* can be inserted into Equations 3 and 2 to predict pairwise labels. In practice, the full covariance terms are needed only for Equations 3, so need only be computed between items for which we wish to predict pairwise labels.

5 Experiments

We use the datasets summarized in Table 1 to test the key aspects of our proposed methods: recovering an underlying consensus from noisy pairwise labels; modeling personal preferences from pairwise labels; and the scalability of our proposed Bayesian preference learning methods, GPPL and crowdGPPL using SVI. In Section 5.1, we use simulated data to test the robustness of our method to noise, small training sets and unknown numbers of latent components. Section 5.2 evaluates the ability of our approach to recover personal and consensus utilities on an NLP task with high-dimensional feature vectors. GPPL previously established the best performance on this dataset in Simpson and Gurevych (2018). Using the same dataset, we then analyze the scalability of our SVI approach. In Section 5.3, we

Dataset	Folds /subsamples	Users	Items	Pairs, train	Pairs, test	Pref vals, test	Item features	User features
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 mean counts per subsample or per fold. For the simulation datasets, we generate the subsamples of data independently, for Sushi we select subsamples independently from the dataset. UKPConvArgCrowdSample is divided into folds, where the test data in each fold corresponds to a single topic and stance. The numbers of features are given after categorical labels have been converted to one-hot encoding, counting each category as a separate feature.

compare our method against previous approaches for predicting the personal preferences of thousands of users on the *Sushi* datasets (Kamishima 2003). Finally, Section 5.4 evaluates whether the Bayesian approach was able to ignore redundant components when C is set larger than necessary.

5.1 Simulated Noisy Data

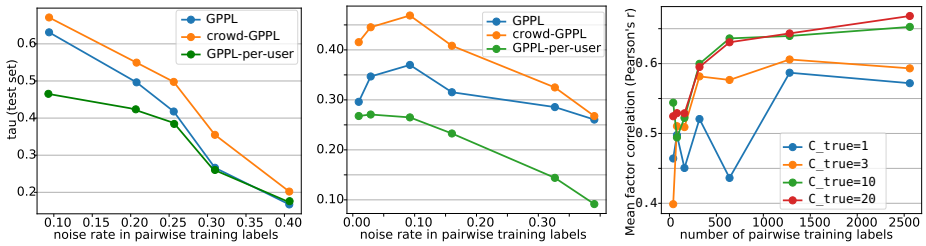
First, we test how well crowdGPPL is able to recover an underlying consensus function from pairwise labels with varying amounts of noise. We compare crowdGPPL against *GPPL* trained on all users' preference labels to learn a single function, and separate GPPL instances per user with no collaborative learning (*GPPL-per-user*). 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:

$$l_{d,MH} = D\text{median}(\{|x_{i,d} - x_{j,d}|\forall i = 1, .., N, \forall j = 1, ..., N\}). \quad (25)$$

is D actually defined?

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 $\epsilon = 1$.

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.



(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.

	Consensus			Personal			Personal, workers with >40 pairs in training set		
Method	Acc	CEE	τ	Acc	CEE	τ	Acc	CEE	τ
GPPL	.77	.50	.40	.71	.56	.32	.72	.55	.26
crowdGPPL	.78	.48	.51	.73	.61	.33	.74	.59	.29

Table 2 Performance comparison on UKPConvArgCrowdSample using ling+GloVe features. *Acc* and *CEE* show classification accuracy and cross entropy error (or log-loss) for pairwise predictions, while τ is Kendall's τ for the predicted preference function.

The second simulation modifies the previous setup by fixing $\sigma = 10$ and varying s to evaluate the methods' ability to 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 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}.

5.2 Argument Convincingness

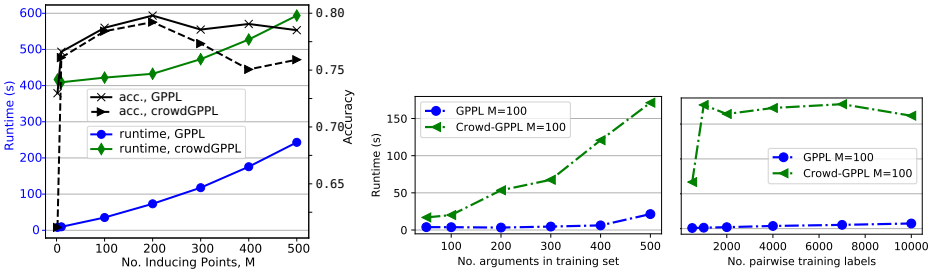
We evaluate consensus learning, personal preference learning and scalability on an NLP task, namely identifying convincing arguments. The dataset, *UKPCon-*

vArgCrowdSample, 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 2 shows performance for GPPL and crowdGPPL. The hyperparameters were kept the same as in Section 5.1 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 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 2. 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 parallelized in future implementations. Increasing the number of pairwise labels, P, above 1000, however, does not visibly affect the runtimes.



(a) Accuracy and runtime with varying M

(b) Varying number of items, N.

(c) Varying no. pairwise training labels, P.

Fig. 2 Runtimes for training+prediction on UKPConvArgCrowdSample. 300 GloVe features. Means over 15 runs. CrowdGPPL with 5 components.

5.3 Sushi Preferences

We use Sushi preference data (shown in Table 1), to benchmark the classification and ranking performance of GPPL and crowdGPPL, against previous approaches and investigate the use of user features for predicting preferences. The datasets contain, for each user, a gold standard preference ranking of 10 types of sushi, from which we generate gold-standard pairwise labels. To test performance with very few training pairs, we obtain *Sushi-A-small* by selecting 100 users at random from the complete *Sushi-A* dataset, then selecting 5 pairs for training and 25 for testing per user. For *Sushi-A*, we select 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 5.1. The complete process of subsampling, training and testing, was repeated 25 times for each dataset.

The results in Table 3 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\induc. GPPL is substantially quicker to train than crowd-GPPL, 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 on *Sushi-A* and *Sushi-A-small*: crowdGPPL\induc and crowdGPPL$\backslash \boldsymbol{u}$ both outperform the full crowdGPPL. To use inducing points,

Method	Sushi-A-small				Sushi-A				Sushi-B			
	Acc	CE E	τ	Run-time	Acc	CE E	τ	Run-time	Acc	CE E	τ	Run-time
crowd-GPPL	.67	.63	.39	38	.81	.39	.66	21	**.79**	**.57**	.50	393
\induc	.70	**.57**	.46	51	.84	.33	.79	39	-	-	-	-
$\backslash u$	.70	.58	.46	9	**.85**	**.31**	**.81**	21	.78	**.57**	.49	303
$\backslash u \backslash x$	**.71**	**.57**	**.49**	5	**.85**	.33	.80	14	.77	.58	.48	269
$\backslash u, \backslash t$	.68	.60	.43	291	.84	.33	.80	230	.76	.61	.54	1001
GPPL	.65	.62	.31	**2**	.65	.62	.31	**2**	.65	.62	.31	**51**
GPPL-per-user	.67	.64	.42	183	.83	.40	.79	44	.75	.60	**.60**	3172

Table 3 Predicting personal preferences: performance on *Sushi-A* dataset and *Sushi-B* datasets. Runtimes are means in seconds over 25 repeats given. For accuracy, the standard deviations of all metrics shown are ≤ 0.02, for CEE, ≤ 0.08, for τ, ≤ 0.03.

there must be a strong relationship between neighbouring points, which may not to be the case given the features in this dataset. However, on *Sushi-B*, the user features provide a small improvement, so may be weakly informative given enough users when we have only 10 pairwise labels per user, rather than 15 as for *Sushi-A*. Comparing crowdGPPL$\backslash \boldsymbol{u}$ with crowdGPPL$\backslash \boldsymbol{ut}$, including the consensus function appears to improve performance by a modest amount. GPPL-per-user performs well on *Sushi-A*, but does not perform as well as other methods on *Sushi-B*, as there are only 10 pairwise training labels per user and no collaborative learning.

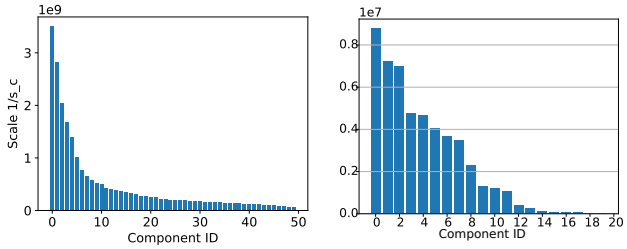
CrowdGPPL produces similar classification scores to the earlier method of Houlsby et al. (2012) (0.83 on *Sushi-A* with user features, and 0.84 without), while using a more scalable inference method. Performance is also improved over Khan et al. (2014) , whose model comprised a GP for each user and matrix factorization (CEE=0.69 on *Sushi-B*), as well as Salimans et al. (2012), using matrix factorization with no user or item features (Kendall's $\tau \approx 0.39$ on *Sushi-B*).

5.4 Posterior Variance of Item Components

We investigate how many latent components were actively used by crowdGPPL on the *UKPConvArgCrowdSample* and *Sushi-A* datasets 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 jointly modeling the individual utilities of members of a crowd and forming a consensus from unreliable annotations, such as those provided by a crowd. Unlike previous methods, our model learns the latent utilities of items from pairwise comparisons using a



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

Fig. 3 Distribution of latent factor variances, $1/s_c$, for crowdGPPL on UKPConvArgCrowdSample, *Sushi-A* and Sushi-B, averaged over all runs.

combination of Gaussian processes and Bayesian matrix factorization to capture divergences in opinion while inferring a consensus. To enable inference at the scale of real-world datasets in fields such as NLP, we derived a stochastic variational inference scheme. Our empirical results confirm that our approach scales well with the amount of training data, and that jointly modeling the consensus and personal preferences of users can improve the predictions of both. When predicting a consensus from crowdsourced data, our model, CrowdGPPL, improves on the results of Simpson and Gurevych (2018), who used a GPPL method that did not account for personal preferences. On a benchmark recommendation dataset, our approach also produces competitive results and showed that modeling the consensus function can boost predictions of individual preferences. Future work will evaluate the benefit of learning inducing point locations from data and optimizing length-scale hyperparameters by maximizing $\mathcal{L}$.

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

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

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

Gaunt A, Borsa D, Bachrach Y (2016) Training deep neural nets to aggregate crowdsourced responses. In: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUAI Press, p 242251

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

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

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: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, vol 33, pp 475–483

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 SM (2016) Capturing reliable fine-grained sentiment associations by crowdsourcing and best–worst scaling. In: Proceedings of NAACL-HLT, pp 811–817

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

Li J, Baba Y, Kashima H (2018) Simultaneous clustering and ranking from pairwise comparisons. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, AAAI Press, pp 1554–1560

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

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

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

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

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

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

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

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 Deriving the Variational Lower Bound for GPPL

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

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \alpha_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)ds}{\int\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}'; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)dsdf'}. \tag{26}$$

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

$$\mathcal{L}_1 = \sum_{i=1}^{L} \mathbb{E}_q \left[\log p\left(v_i \succ u_i | f(v_i), f(u_i)\right)\right] + \mathbb{E}_q\left[\log \frac{p\left(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K}/s\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_q\left[\log \frac{p\left(s|\alpha_0, \beta_0\right)}{q\left(s\right)}\right] \tag{27}$$

Substituting the forms of the distributions with their variational parameters, we get:

$$\begin{aligned}\mathcal{L}_1 = \mathbb{E}_q\left[\sum_{i=1}^{L} [v_i \succ u_i] \log \Phi(z_i) + [v_i \prec u_i]\left(1 - \log \Phi(z_i)\right)\right] \\ + \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) + \mathbb{E}_q\left[\log \mathcal{G}\left(s; \alpha_0, \beta_0\right) - \log \mathcal{G}\left(s; \alpha, \beta\right)\right]\end{aligned} \tag{28}$$

We now replace the likelihood with a Gaussian approximation:

$$\begin{aligned}\mathcal{L}_1 \approx \mathcal{L}_2 &= \mathbb{E}_q\left[\mathcal{N}(\boldsymbol{y}|\Phi(\boldsymbol{z}), \boldsymbol{Q})\right] + \log \mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ &\quad + \mathbb{E}_q\left[\log \mathcal{G}\left(s; \alpha_0, \beta_0\right) - \log \mathcal{G}\left(s; \alpha, \beta\right)\right] \\ &= -\frac{1}{2}\Big\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}/s| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &\quad + \mathbb{E}_q\left[(\boldsymbol{y} - \Phi(\boldsymbol{z}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\boldsymbol{z}))\right]\Big\} - \Gamma(\alpha_0) + \alpha_0(\log \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\log s] \\ &\quad + \Gamma(\alpha) + (\beta - \beta_0)\hat{s} - \alpha \log \beta,\end{aligned} \tag{29}$$

where $\mathbb{E}[s] = \frac{\alpha}{\beta}$, $\mathbb{E}[\log s] = \Psi(2\alpha) - \log(2\beta)$, Ψ is the digamma function and $\Gamma()$ is the gamma function, Finally, we use a Taylor-series linearisation to make the remaining expectation tractable:

$$\begin{aligned}\mathcal{L}_2 \approx \mathcal{L}_3 = -\frac{1}{2}\{&L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}/\hat{s}| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &+ (\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))\Big\} - \Gamma(\alpha_0) + \alpha_0(\log \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\log s] \\ &+ \Gamma(\alpha) + (\beta - \beta_0)\hat{s} - \alpha \log \beta.\end{aligned} \tag{30}$$

Now, we can introduce the sparse approximation to obtain the bound in Equation 15:

$$\begin{aligned}\mathcal{L} \approx \mathcal{L}_3 &= \mathbb{E}_{q(\boldsymbol{f}, \boldsymbol{f}_m, s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)] \\ &= \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|f_{a_p}, f_{b_p})] - \frac{1}{2}\Big\{\log|\boldsymbol{K}_{mm}| - \mathbb{E}[\log s] - \log|\boldsymbol{S}| - M \\ &\quad + \hat{\boldsymbol{f}}_m^T \mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\Big\} + \log \Gamma(\alpha) - \log \Gamma(\alpha_0) + \alpha_0(\log \beta_0) \\ &\quad + (\alpha_0 - \alpha)\mathbb{E}[\log s] + (\beta - \beta_0)\mathbb{E}[s] - \alpha \log \beta,\end{aligned} \tag{31}$$

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:

$$\begin{aligned}
\mathcal{L}_{cr} = &\sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|\boldsymbol{v}_{a_p}^T \boldsymbol{w}_{a_p} + t_{a_p}, \boldsymbol{v}_{b_p}^T \boldsymbol{w}_{b_p} + t_{b_p})] - \frac{1}{2}\Bigg\{ \sum_{c=1}^{C} \Bigg\{ -M_n - M_u \\
&+ \log|\boldsymbol{K}_{v,mm}| + \log|\boldsymbol{K}_{w,mm}| - \log|\boldsymbol{S}_{v,c}| - \mathbb{E}[\log s_c] + \hat{\boldsymbol{v}}_{m,c}^T \mathbb{E}[s_c] \boldsymbol{K}_{v,mm}^{-1} \hat{\boldsymbol{v}}_{m,c} \\
&+ \mathrm{tr}(\mathbb{E}[s_c]\boldsymbol{K}_{v,mm}^{-1}\boldsymbol{S}_{v,c}) - \log|\boldsymbol{\Sigma}_c| + \hat{\boldsymbol{w}}_{m,c}^T \boldsymbol{K}_{w,mm}^{-1} \hat{\boldsymbol{w}}_{m,c} + \mathrm{tr}(\boldsymbol{K}_{w,mm}^{-1}\boldsymbol{\Sigma}_c) \Bigg\} \\
&- M_n + \log|\boldsymbol{K}_{t,mm}| - \log|\boldsymbol{S}_t| - \mathbb{E}[\log\sigma] + \hat{\boldsymbol{t}}^T \mathbb{E}[\sigma] \boldsymbol{K}_{t,mm}^{-1} \hat{\boldsymbol{t}} \\
&+ \mathrm{tr}(\mathbb{E}[\sigma]\boldsymbol{K}_{t,mm}^{-1}\boldsymbol{S}_t) \Bigg\} - (C+1)(\log\Gamma(\alpha_0) + \alpha_0(\log\beta_0)) \\
&+ \sum_{c=1}^{C} \Bigg\{ \log\Gamma(\alpha_c) + (\alpha_0 - \alpha_c)\mathbb{E}[\log s_c] + (\beta_c - \beta_0)\mathbb{E}[s_c] - \alpha_c \log\beta_c \Bigg\} \\
&+ \log\Gamma(\alpha_\sigma) + (\alpha_0 - \alpha_\sigma)\mathbb{E}[\log\sigma] + (\beta_\sigma - \beta_0)\mathbb{E}[s_c] - \alpha_\sigma \log\beta_\sigma, \qquad (32)
\end{aligned}$$

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:

$$\begin{aligned}
\boldsymbol{S}_{v,c,i}^{-1} = &(1-\rho_i)\boldsymbol{S}_{v,c,i-1}^{-1} + \rho_i \left(\boldsymbol{K}_{v,mm}^{-1}\mathbb{E}[s_c] \right. \\
&\left. + w_i \boldsymbol{A}_{v,i}^T \boldsymbol{G}_i^T \mathrm{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}}) \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_{v,i} \right), \qquad (33)
\end{aligned}$$

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 39 and 38), 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 $\boldsymbol{V}_m$:

$$\begin{aligned}
\hat{\boldsymbol{v}}_{m,c,i} = \boldsymbol{S}_{v,c,i} \Bigg(&(1-\rho_i)\boldsymbol{S}_{v,c,i-1}^{-1}\hat{\boldsymbol{v}}_{m,c,i-1} + \rho_i w_i \Bigg(\boldsymbol{S}_{v,c,i} \boldsymbol{A}_i^T \boldsymbol{G}_i^T \mathrm{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}) \boldsymbol{Q}_i^{-1} \\
&\left(\boldsymbol{y}_i - \boldsymbol{\Phi}(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_j^{(i)} (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \right) \Bigg) \Bigg), \qquad (34)
\end{aligned}$$

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:

$$H_{j,p,n}^{(i)} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\mathbb{E}[z_p]))(2y_p - 1)([n = a_p] - [n = b_p])[j = u_p]. \qquad (35)$$

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

$$\boldsymbol{S}_{t,i}^{-1} = (1-\rho_i)\boldsymbol{S}_{t,i-1}^{-1} + \rho_i \boldsymbol{K}_{t,mm}^{-1}/\mathbb{E}[\sigma] + \rho_i w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i \qquad (36)$$

$$\begin{aligned}
\hat{\boldsymbol{t}}_{m,i} = &\boldsymbol{S}_{t,i} \left((1-\rho_i)\boldsymbol{S}_{t,i-1}^{-1}\hat{\boldsymbol{t}}_{m,i-1} \right. \\
&\left. + \rho_i w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \left(\boldsymbol{y}_i - \boldsymbol{\Phi}(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{t}}_{m,i} \right) \right). \qquad (37)
\end{aligned}$$

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

$$\boldsymbol{\Sigma}_{c,i}^{-1} = (1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}^{-1} + \rho_i \boldsymbol{K}_{w,mm}^{-1} + \rho_i w_i \boldsymbol{A}_{w,i}^T \Bigg(\sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)T} \operatorname{diag}\left(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 + \right.$$

$$\left. \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}} - 2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}\hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}}\right) \boldsymbol{Q}_i^{-1} \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \Bigg) \boldsymbol{A}_{w,i} \tag{38}$$

$$\hat{\boldsymbol{w}}_{m,c,i} = \boldsymbol{\Sigma}_{c,i} \Bigg((1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}\hat{\boldsymbol{w}}_{m,c,i-1} + \rho_i w_i \boldsymbol{A}_{w,i}^T \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \left(\operatorname{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}) \right.$$

$$\left. - \operatorname{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\right) \boldsymbol{Q}_i^{-1} \Bigg(\boldsymbol{y}_i - \boldsymbol{\Phi}(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_u^{(i)} (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \Bigg) \Bigg), \tag{39}$$

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

C Predictions with CrowdGPPL

The means, item covariances and user variance required for predictions with crowdGPPL (Equation 24) are defined as follows:

$$\hat{\boldsymbol{t}}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{t}}_m, \qquad \hat{\boldsymbol{v}}_c^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{v}}_{m,c}, \qquad \hat{\boldsymbol{w}}_c^* = \boldsymbol{K}_{w,*m}\boldsymbol{K}_{w,mm}^{-1}\hat{\boldsymbol{w}}_{m,c}, \tag{40}$$

$$\boldsymbol{C}_t^* = \frac{\boldsymbol{K}_{**}}{\mathbb{E}[\sigma]} + \boldsymbol{A}_{*m}(\boldsymbol{S}_t - \boldsymbol{K}_{mm})\boldsymbol{A}_{*m}^T, \qquad \boldsymbol{C}_{v,c}^* = \frac{\boldsymbol{K}_{**}}{\mathbb{E}[s_c]} + \boldsymbol{A}_{*m}(\boldsymbol{S}_{v,c} - \boldsymbol{K}_{mm})\boldsymbol{A}_{*m}^T \tag{41}$$

$$\omega_{c,u}^* = 1 + \boldsymbol{A}_{w,um}(\boldsymbol{\Sigma}_{w,c} - \boldsymbol{K}_{w,mm})\boldsymbol{A}_{w,um}^T \tag{42}$$

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