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

Scalable and Sparse: Bayesian Preference Learning with Crowds

Edwin Simpson · Iryna Gurevych

Received: date

Abstract We show how to make collaborative preference learning work at scale and how it can be used to learn a target preference function from crowdsourced data or other noisy preference labels. The collaborative model captures the reliability of each worker or data source and models their biases and error rates. It uses latent factors to share information between similar workers and a target preference function. We devise an SVI inference schema to enable the model to scale to real-world datasets. Experiments compare results using standard variational inference, laplace approximation and SVI. On real-world data we show the benefit of the personalised model over a GP preference learning approach that treats all labels as coming from the same source, as well as established alternative methods and classifier baselines. We show that the model is able to identify a number of latent features for the workers and for textual arguments.

1 Introduction

Many tasks are more suited to pairwise comparisons than classification etc. Crowds of non-expert annotators may label more accurately if presented with pairs. Implicit feedback may be taken from user actions in an application that can be represented as a preference, such as choosing an option over other options.

There are several works for learning from noisy pairwise comparisons so far (Horvitz et al. 2013 or something like that?). However, these do not provide a way to take account of item features or to model different but valid subjective viewpoints. They assume there is a single ground truth and can therefore

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

model only one task and one user's (or a consensus of all users) preferences at once.

Work by Felt et al. 2015, Simpson et al. 2015 etc. shows that item features are particularly useful when combining crowdsourced data. A Gaussian process has not been tested for this purpose before?

GP preference learning presents a way to learn from noisy preferences but assumes constant noise and a single underlying preference function. The collaborative Gaussian process (Houlsby et al. 2012) learns multiple users' preferences. However existing implementations do not scale and do not identify ground truth.

We show how to scale it using SVI and how to use the model to identify ground truth from subjective preferences.

In this paper, we develop methodology to solve the following questions:

1. How can we learn a rating function over large sets of items given a large number of pairwise comparisons?
2. How do we account for the different personal preferences of annotators when inferring the ground truth?

To answer these questions we make the following technical contributions:

1. We propose a method for predicting either gold-standard or personalized ratings by aggregating crowdsourced preference labels using a model of the noise and biases of individual annotators.
2. To enable this method to scale to large, real-world datasets, we develop stochastic variational inference for Bayesian matrix factorization and Gaussian process preference learning.
3. To expedite hyper-parameter tuning, we introduce a technique for gradient-based length-scale optimization of Gaussian processes.

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 and hyper-parameter optimisation technique in Section 5. Then, in Section 6, we evaluate our approach empirically, showing first its behaviour on synthetic data, then its scalability and predictive performance on several real-world datasets.

2 Related Work

2.1 Preference Learning from Crowds

Several works have analyzed bounds on error rates or sample complexity for pairwise learning (Chen and Suh 2015; Shah et al. 2015), but do not propose methods for learning multiple rankings from crowds of users. Chen et al. (2013) account for the varying quality of pairwise labels obtained from a crowd by learning an individual model of agreement with the true pairwise labels for each worker. This approach treats the inconsistencies between annotators' labels as noise and does not consider the items' features. Therefore, this method does not learn the workers' individual preferences and cannot model how their accuracy depends on the items considered. In contrast, Fu et al. (2016) consider item features when learning to rank from pairwise labels, but do not model individual annotators at all. Uchida et al. (2017) do model the confidence of individual annotations and propose a fuzzy ranking SVM to make predictions given item features. However, their approach also assumes a single ranking over items. The benefit of jointly learning to rank and group items has also been explored(Li et al. 2018), again assuming a single ordering.

Tian and Zhu (2012) consider crowdsourcing tasks where there may be more than one correct answer. They use a nonparametric Dirichlet process model to infer a variable number of clusters of answers for each task, and also infer annotator reliability. However, they do not apply the approach to ranking using pairwise labels. Several other works learn multiple rankings from crowdsourced pairwise labels rather than a single gold-standard ranking, but do not consider the item or user features so cannot extrapolate to new users or items (Yi et al. 2013; Kim et al. 2014; Wang et al. 2016; Kim et al. 2017). Both Yi et al. (2013) and Kim et al. (2017) learn a small number of latent ranking functions that can be combined to construct personalized preferences, although neither provide a Bayesian treatment to handle data sparsity. Wang et al. (2016) consider the case where different rankings correspond to lists of items provided in response to search queries. While they model the dependence of annotator accuracy on the domain of a query, their approach was not applied to personal or subjective rankings.

A number of studies consider actively selecting pairs of items for comparison to minimize the number of pairwise labels required (Radlinski and Joachims 2007; Qian et al. 2015; Maystre and Grossglauser 2017; Cai et al. 2017). Related research treats the selection of pairwise labels as a multi-armed bandit problem (Busa-Fekete et al. 2018). In this work, we do not study the process of learning from an oracle or user that we can query. Rather, we develop a model for aggregating pairwise labels from multiple sources, which can be used as the basis of active learning methods that exploit the model uncertainty estimates provided by this Bayesian approach.

2.2 Bayesian Preference Learning

A Bayesian approach to preference learning with Gaussian processes, *GPPL*, uses item features to can make predictions for unseen items and share information between similar items (Chu and Ghahramani 2005). This model assumes a single preference function over items, so cannot be used to model the individual preferences of multiple users. The approach was extended by Houlsby et al. (2012) to capture individual preferences using a latent factor model. Pairwise labels from users with common interests help to predict each other's preference function, hence this can be seen as a *collaborative* learning method, as used in *recommender systems*. The inference techniques proposed for this model mean it scales poorly, with computational complexity $\mathcal{O}(N^3 + NP)$, where N is the number of items and P is the number of pairwise labels, and memory complexity $\mathcal{O}(N^2+NP+P^2)$. In this paper, we address this issue and adapt the model for aggregating crowdsourced data. An alternative to using a latent factor model is to cluster users according to preferences (Abbasnejad et al. 2013), but this is less flexible in that it does not allow for collaborative learning between users with common preferences for only subsets of items (e.g. two users may both like one genre of music, while having different preferences over other genres).

2.3 Bayesian Matrix Factorization

Preference data can be represented as an item-user matrix with N rows and M columns, where N is the number of items and M is the number of users. In this paper, we are interested in the task of predicting values in this matrix given only sparse observations of pairwise comparisons. Matrix factorization techniques are commonly used to discover latent user and item features but can fail if the data is very sparse, unless suitably regularised or given a Bayesian treatment (Salakhutdinov and Mnih 2008). Recent work on scaling Bayesian matrix factorization (BMF) to large datasets has focused on parallelizing inference (Ahn et al. 2015; Vander Aa et al. 2017; Chen et al. 2018). Instead of distributing the computation, this paper focuses on reducing the computational cost, although the method we propose is amenable to parallelization.

Several extensions of BMF use Gaussian process priors over latent factors to model correlations between items given side information or observed item features (Adams et al. 2010; Zhou et al. 2012; Houlsby et al. 2012; Bolgár and Antal 2016). However, these techniques are not directly applicable to learning from pairwise comparisons as they assume that the observations are Gaussian-distributed numerical ratings (Shi et al. 2017).

To combine Bayesian matrix factorization with a pairwise likelihood, Houlsby et al. (2012) propose a combination of expectation propagation and variational Bayesian inference. However, their proposed method does not scale sufficiently to the numbers of items, users or pairwise labels found in many important application domains. In contrast, Khan et al. (2014) develop a scalable variational EM algorithm for matrix factorization but combine this with a separate GP to model each user's preferences. However, while the proposed method can be trained with pairwise labels, it does not capture correlations between items or users in the latent factors. Furthermore, their scalable inference method sub-samples training data rather than learning from the complete training set.

2.4 Stochastic Variational Inference

Models that combine Gaussian processes with non-Gaussian likelihoods require approximate inference methods that often scale poorly with the amount of training data available. This problem can be tackled using *Stochastic variational inference (SVI)* (Hoffman et al. 2013). SVI has been successfully applied to Gaussian processes (Hensman et al. 2013), including Gaussian process classifiers (Hensman et al. 2015), and Gaussian process preference learning (GPPL) (Simpson and Gurevych 2018). This paper adapts SVI to Bayesian matrix factorization for the first time as part of a solution for collaborative preference learning. We also provide the first full derivation of SVI for GPPL and introduce a technique for efficiently tuning the length-scale hyperparameters of the Gaussian processes.

3 Bayesian Preference Learning for Crowds

3.1 Modeling Pairwise Preferences

A pairwise comparison, $y(a \succ b)$, between items a and b has a binary label that is one if a is preferred to b, or zero if b is preferred to a (also written $a \prec b$). We assume that the likelihood of pairwise label $y(a, b)$ depends on the underlying value of the items to the user, represented through a latent function of the items' features, $f(\boldsymbol{x}_a)$, where $\boldsymbol{x}_b$ is a vector representation of the features of item a. The relationship between the value function, f, and the pairwise labels can be modeled by any of several different likelihood functions, including the Bradley-Terry model (Bradley and Terry 1952; Plackett 1975; Luce 1959) and the Thurstone-Mosteller model (Thurstone 1927; Mosteller 2006). The Bradley-Terry model takes the following form:

$$p(y(a \succ b)|f) = \frac{1}{1 + \exp(f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b))} \tag{1}$$

This is a logistic likelihood, which allows pairwise labels that do not reflect the true relative values of the items, due to labeling errors, variability in the user's judgements, or if the preferences are derived from noisy implicit data such as clicks streams. The error rate is determined by the relative difference in f values of the items.

A different view is to treat the errors as the result of random noise in the value function:

$$p(y(a \succ b)|f, \delta_a, \delta_b) \qquad = \begin{cases} 1 & \text{if } f(\boldsymbol{x}_a) + \delta_a \geq f(b) + \delta_b \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\delta \sim \mathcal{N}(0, 0.5)$ is Gaussian-distributed noise. Integrating out the unknown values of δ_a and δ_b, we get a probit likelihood:

$$\begin{aligned} p(y(a \succ b)|f) &= \int\int p(y(a \succ b)|f, \delta_a, \delta_b)\mathcal{N}(\delta_a; 0, 0.5)\mathcal{N}(\delta_b; 0, 0.5)d\delta_a d\delta_b \\ &= \Phi(z), \end{aligned} \tag{3}$$

where $z = f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b)$, and Φ is the cumulative distribution function of the standard normal distribution. This is a Thurstone-Mosteller model, sometimes referred to as *Thurstone case V*, and was used for Gaussian process preference learning (GPPL) by Chu and Ghahramani (2005), with the difference that they learned the variance of the random noise, δ rather than assuming it is 0.5. However, this is unnecessary in practice, since we scale instead the value function, f, to reduce or increase the certainty in the pairwise labels. Both the logistic and probit approaches can be used here, but we proceed with the probit likelihood (as in (Herbrich et al. 2007; Chu and Ghahramani 2005) because it allows us to handle uncertainty in f in a simple manner by modifying z:

$$\hat{z} = \frac{\mu_a - \mu_b}{\sqrt{1 + \sigma_a + \sigma_b - \sigma_{a,b}}} \tag{4}$$

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 $f(\boldsymbol{x}_b)$.

3.2 Single User Preference Learning

First consider modeling the preferences of a single user. In this case, we assume that the value function, f, is a function of item features and has a Gaussian process prior: $f \sim \mathcal{GP}(0, k_\theta/s)$, where k_θ is a kernel function with hyper-parameters θ, and $1/s$ is the scale of the function 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 3. The kernel function takes item features as inputs and determines the covariance between values of f for different items. Typically, we choose a kernel function that produces higher covariance between items with similar feature values, such as the *squared exponential* or *Matérn* functions. The choice of kernel function is a model selection problem as it controls the shape and smoothness of the function across the feature space. However, the Matérn and squared exponential make minimal assumptions and so are effective in a wide range of tasks (see Rasmussen and Williams (2006) for more).

We observe a set of P pairwise preference labels for a single user, $\boldsymbol{y} = \{y_1, ..., y_P\}$, where the pth label, $y_p = y(a_p \succ b_p)$.The joint distribution over all variables is as follows:

$$\begin{aligned} p\left(\boldsymbol{y}, \boldsymbol{f}, s | k_\theta, \alpha_0, \beta_0\right) &= \prod_{p=1}^{P} p(y_p|\boldsymbol{f}) \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s) \mathcal{G}(s; \alpha_0, \beta_0) \\ &= \prod_{p=1}^{P} \Phi\left(z_p\right) \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s) \mathcal{G}(s; \alpha_0, \beta_0), \end{aligned} \tag{5}$$

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)\}$ are the latent values for the N items referred to by the pairwise labels $\boldsymbol{y}$, and θ, α_0 and β_0 are hyper-parameters.

3.3 Latent components: Bayesian Matrix Factorization

We wish to exploit similarities between the value functions of different users or label sources to improve our preference model, particularly when faced with sparse data. In a scenario with multiple users or label sources, we can represent preference values in a matrix, $\boldsymbol{F}$, where rows correspond to items, columns to users, and entries are preference values. If we factorize this matrix, we obtain two lower-dimensional matrices, one for users, $\boldsymbol{W} \in \mathcal{R}^{C \times U}$, and one for the items, $\boldsymbol{V} \in \mathcal{R}^{N \times C}$, where C is the number of latent components, U is the number of users, and N is the number of items: $\boldsymbol{F} = \boldsymbol{V}^T \boldsymbol{W}$. Each row of V matrices is a vector representation of an item, while each row of W is a vector representation of a user, both containing the values of latent features. Users with similar values for a certain feature will have similar preferences for the subset of items with corresponding feature values. The features could represent, for example, in the case of book recommendation, interests in a particular genre of book. Using vector representations for users and items reflects that users may have overlapping sets of interests, and that items may have multiple features that make them attractive.

Besides latent features, we may also observe a number of item features, $\boldsymbol{x}$, and user features, $\boldsymbol{u}$. In the single user model, we assumed a single latent value function, f, of the observed item features. For the multi-user case, we assume that there are C latent functions, v_c over item features and C latent functions, w_c, over user features, and thereby model the relationship between each observed feature and each of the latent features. The matrices V and W are evaluations of these functions at the points corresponding to the observed users and items. Therefore, the latent preference function, f, for a user with features $\boldsymbol{u}$ is a weighted sum over latent functions:

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

To provide a Bayesian treatment to matrix factorization, we place Gaussian process priors over the latent functions:

$$v_c \sim \mathcal{GP}(\boldsymbol{0}, k_\theta / s_c) \qquad\qquad w_c \sim \mathcal{GP}(\boldsymbol{0}, k_\theta). \tag{7}$$

It is not necessary to learn a separate scale for w_c, since v_c and w_c are multiplied with each other, making a single s_c equivalent to the product of two separate scales. The choice of C can be treated as a hyperparameter, or modeled using a non-parametric prior, such as the Indian Buffet Process, which assumes an infinite number of latent components (Ding et al. 2010). For simplicity, we assume fixed values of C in this paper, and allow the scale parameter $s_c \approx 0$ to effectively remove any dimensions that are not required to model the data. This section described a Bayesian matrix factorization model, which we will subsequently extend to a preference learning model for crowds of users and label sources.

3.4 Crowd Preference Learning

We combine the matrix factorization method with the preference likelihood of Equation 3 to obtain a joint preference model for multiple users or label sources. In addition to the latent components, we introduce a common value function over item features, $t \sim \mathcal{GP}(\mathbf{0}, k_\theta/\sigma_t)$, that is shared across all users. Its values $\boldsymbol{t} = \{t(\boldsymbol{x}_1), ..., t(\boldsymbol{x}_N)\}$ represent a consensus between users, if present, while allowing individual users' preferences to deviate from this value through $\boldsymbol{V}^T\boldsymbol{W}$. Hence, $\boldsymbol{t}$ can model the underlying ground truth or consensus in crowdsourcing scenarios, or when using multiple label sources to learn preferences for one individual. The joint distribution of this crowd model is:

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

$$\prod_{c=1}^{C} \left\{ \mathcal{N}(\boldsymbol{v}_c; \mathbf{0}, \boldsymbol{K}_{v,\theta}/s_c) \mathcal{N}(\boldsymbol{w}_c; \mathbf{0}, \boldsymbol{K}_{w,\theta}) \mathcal{G}(s_c; \alpha_0, \beta_0) \right\}, \tag{8}$$

$$\text{where } z_p = \boldsymbol{v}_{.,a_p}^T \boldsymbol{w}_{.,u_p} + t_{a_p} - \boldsymbol{v}_{.,b_p}^T \boldsymbol{w}_{.,u_p} - t_{b_p}, \tag{9}$$

and σ_t is the inverse scale of t. The index p, which identifies one observation, now refers to a tuple, $\{u_p, a_p, b_p\}$ that identifies the user and a pair of items.

4 Scalable Inference

Given a set of pairwise labels, $\boldsymbol{y}$, the goal is to infer the posterior distribution over the preference values $\boldsymbol{f}$, in the single user case, or $\boldsymbol{F} = \boldsymbol{V}^T\boldsymbol{W}$ in the multi-user case. Previous approaches include a Laplace approximation for the single user case (Chu and Ghahramani 2005) and 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 tasks such as classification (Nickisch and Rasmussen 2008). EP approximates the true posterior with a simpler, factorized distribution that can be learned using an iterative algorithm. The true posterior is multi-modal, since the latent factors can be re-ordered arbitrarily without affecting $\boldsymbol{F}$: this is the non-identifiability problem. A standard EP approximation would average these modes before predicting $\boldsymbol{F}$, producing uninformative predictions over $\boldsymbol{F}$. Houlsby et al. (2012) resolve this by incorporating a VB step, which approximates a single mode. A drawback of EP is that unlike VB, convergence is not guaranteed (Minka 2001).

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 fixed at $\mathcal{O}(M^3)$. 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. Houlsby et al. (2012) use a FITC approximation (Snelson and Ghahramani 2006) with their EP method to limit the costs of inference. However, in practice, FITC is unsuitable for datasets with more than a few thousands points as it is not amenable to distributed computation, does it address other expensive operations with computational complexity $\mathcal{O}(NP)$ and memory complexity $\mathcal{O}(P^2+NP+N^2)$, which may become limiting when the number of data points is large(Hensman et al. 2015). We turn to stochastic variational inference (SVI) (Hoffman et al. 2013) to derive a more scalable approach for Gaussian process preference learning, including a multi-user model founded on Bayesian matrix factorization. First, we define an approximate posterior that can be estimated using SVI, then provide the update equations for an iterative algorithm to optimize this approximation. We begin with the model for a single user, then extend this to the multi-user case using matrix factorization.

4.1 An Approximate Preference Likelihood

Due to the non-Gaussian likelihood, Equation 3, the posterior distribution over $\boldsymbol{f}$ contains intractable integrals:

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \beta_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p) \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s) \mathcal{G}(s; \alpha_0, \beta_0) ds}{\int \int \prod_{p=1}^{P} \Phi(z_p) \mathcal{N}(\boldsymbol{f}'; \boldsymbol{0}, \boldsymbol{K}_\theta/s) \mathcal{G}(s; \alpha_0, \beta_0) ds df'}. \tag{10}$$

To simplify the integral in the denominator, we approximate the preference likelihood with a Gaussian:

$$\prod_{p=1}^{P} \Phi(z_p) \approx \mathcal{N}(\boldsymbol{y}; \Phi(\boldsymbol{z}), \boldsymbol{Q}), \tag{11}$$

where $\boldsymbol{z} = \{z_1, ..., z_P\}$ and $\boldsymbol{Q}$ is a diagonal noise covariance matrix. We estimate the diagonal entries of $\boldsymbol{Q}$ by moment matching the approximate likelihood with a beta-binomial with variance given by:

$$Q_{p,p} = \mathbb{E}_{\boldsymbol{f}}[\Phi(z_p)(1 - \Phi(z_p))] = \frac{(y_p + \gamma_0)(1 - y_p + \lambda_0)}{(2 + \gamma_0 + \lambda_0}, \tag{12}$$

where γ_0 and λ_0 are parameters of a Bernoulli distribution that has the same variance as the prior $p(\Phi(z_p)|\boldsymbol{K}_\theta, \alpha_0, \beta_0)$ using numerical integration. Setting $\boldsymbol{Q}$ in this way matches the moments of the true likelihood, $\Phi(z_p)$, to those of the Gaussian approximation.

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 expected value of $\boldsymbol{f}$:

$$\Phi(\boldsymbol{z}) \approx \tilde{\Phi}(\boldsymbol{z}) = \boldsymbol{G}(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}]) + \Phi(\mathbb{E}[\boldsymbol{z}]), \tag{13}$$

$$G_{p,i} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\mathbb{E}[z_p]))(2y_p - 1)([i = a_p] - [i = b_p]) \tag{14}$$

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 dependency between the posterior mean of $\boldsymbol{f}$ and the linearization terms in the likelihood, which can be estimated iteratively using variational inference (Steinberg and Bonilla 2014), as we will describe below. The linearization makes the approximate likelihood conjugate to $\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)$, so that the approximate posterior over $\boldsymbol{f}$ is also Gaussian.

Given our likelihood approximation, we can now use variational inference to estimate the marginal over $\boldsymbol{f}$, by optimizing an approximate posterior over all latent variables:

$$\begin{aligned}
p(\boldsymbol{f}, s|\boldsymbol{y}, \boldsymbol{x}, k_\theta, \alpha_0, \beta_0) &\approx q(s)q(\boldsymbol{f}), \\
\text{where } \log q(s) &= \mathbb{E}_{\boldsymbol{f}}[\log p(s|\boldsymbol{f}, \alpha_0, \beta_0)] \\
&= \log \mathcal{N}(\mathbb{E}[\boldsymbol{f}]; \boldsymbol{0}, \boldsymbol{K}_\theta/s) + \log \mathcal{G}(s; \alpha_0, \beta_0) + \text{const}, \\
\log q(\boldsymbol{f}) &= \mathbb{E}_s[\log p(\boldsymbol{f}|k_\theta, s, \boldsymbol{x}, \boldsymbol{y})] \\
&= \log \mathcal{N}(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}) + \log \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/\mathbb{E}[s]) + \text{const}.
\end{aligned} \tag{15}$$

Since the Gamma is conjugate to the variance of a Gaussian, $q(s)$ is also a Gamma distribution. Likewise, since the likelihood approximation is Gaussian and its mean is a linear function of $\boldsymbol{f}$, which has a multivariate Gaussian prior, $q(\boldsymbol{f})$ is also a multivariate Gaussian. The Gaussian likelihood approximation and linearization also appear in methods based on expectation propagation (Rasmussen and Williams 2006) and the extended Kalman filter (Reece et al. 2011; Steinberg and Bonilla 2014). However, neither these methods nor our approximation in Equation 15 make inference sufficiently scalable, as they all require inversion of an $N \times N$ matrix and further computations involving $N \times P$ and $P \times P$ matrices. We therefore modify Equation 15 to enable stochastic variational inference (SVI).

4.2 Sparse Approximation for the Single-User Model

We introduce a sparse approximation to the Gaussian process that allows us to limit the size of the covariance matrix that needs to be inverted, and permit stochastic inference methods that consider only a subset of the P observations at each iteration (Hensman et al. 2013, 2015). To do this, we introduce a set of $M \ll N$ *inducing points*, with inputs $\boldsymbol{x}_m$, function values $\boldsymbol{f}_m$, covariance $\boldsymbol{K}_{mm}$, and covariance between the observations and the inducing points is $\boldsymbol{K}_{nm}$. The inducing points act as proxies for the observed points during inference, and thereby reduce the number of data points we have to perform costly operations over. We modify the variational approximation in Equation 15 to introduce the inducing points (for clarity, we omit θ from this point on):

$$p(\boldsymbol{f}, \boldsymbol{f}_m, s|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}_m, k_\theta, \alpha_0, \beta_0) \approx q(\boldsymbol{f}, \boldsymbol{f}_m, s) = q(s)q(\boldsymbol{f})q(\boldsymbol{f}_m), \tag{16}$$

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

$$\begin{aligned} \log q(\boldsymbol{f}_m) &= \log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}\right)] + \log \mathcal{N}\left(\boldsymbol{f}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}\left[s\right]\right) + \text{const}, \\ &= \log \mathcal{N}(\boldsymbol{f}_m; \hat{\boldsymbol{f}}_m, \boldsymbol{S}), & (17) \\ \boldsymbol{S}^{-1} &= \boldsymbol{K}_{mm}^{-1}/\mathbb{E}[s] + \boldsymbol{A}^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} \boldsymbol{G} \boldsymbol{A}, & (18) \\ \hat{\boldsymbol{f}}_m &= \boldsymbol{S} \boldsymbol{A}^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} (\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \boldsymbol{G}\mathbb{E}[\boldsymbol{f}]), & (19) \end{aligned}$$

where $\boldsymbol{A} = \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1}$. We choose an approximation of $q(\boldsymbol{f})$ that depends only on the function values at the inducing points, $\boldsymbol{f}_m$, and is independent of the observations:

$$\log q(\boldsymbol{f}) = \log \mathcal{N}(\boldsymbol{f}; \boldsymbol{A}\hat{\boldsymbol{f}}_m, \boldsymbol{K} + \boldsymbol{A}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{A}^T). \tag{20}$$

This means that we no longer need to invert an $N \times N$ covariance matrix to compute the posterior. The factor $q(s)$ is also modified to depend on the inducing points:

$$\log q(s) = \log \mathcal{N}(\mathbb{E}[\boldsymbol{f}_m|\boldsymbol{0}, \boldsymbol{K}_m m/s] + \log \mathcal{G}(s; \alpha_0, \beta_0) + \text{const} = \log \mathcal{G}(s; \alpha, \beta) \tag{21}$$

where $\alpha = \alpha_0 + \frac{M}{2}$ and $\beta = \beta_0 + \frac{\text{tr}(\boldsymbol{K}_{mm}^{-1}(S + \hat{\boldsymbol{f}}_m \hat{\boldsymbol{f}}_m^T))}{2}$. Since none of the other factors depend on $\log q(\boldsymbol{f})$, it need not be estimated until prediction time once the other factors have already been estimated, as explained in the next subsection.

Since the inducing points stand in for the observed locations, their choice affects the quality of our approximation. To choose inducing points that represent the spread of data in our observations across feature space, we use K-means++ Arthur and Vassilvitskii (2007) with $K = M$ to cluster the feature vectors, then take the cluster centers as inducing points. An alternative approach would be to learn the placement of inducing points as part of the variational inference procedure (**?**), or by maximizing the variational lower bound on the log marginal likelihood (see next section). However, the former breaks the convergence guarantees, and both approaches may add substantial computational cost. Therefore, in this paper, we show that it is often sufficient to place inducing points up-front, and leaving their optimization for future work.

4.3 SVI for Single-User Preference Learning

To estimate the approximate posterior, we can apply variational inference, which iteratively reduces the KL-divergence between our approximate posterior, $q(s)q(\boldsymbol{f})q(\boldsymbol{f}_m)$ and the true posterior, $p(s, \boldsymbol{f}, \boldsymbol{f}_m | \boldsymbol{K}, \alpha_0, \beta_0, \boldsymbol{y})$, by maximizing a lower bound, $\mathcal{L}$, on the marginal likelihood, $\log p(\boldsymbol{y}|\boldsymbol{K}, \alpha_0, \beta_0)$:

$$\log p(\boldsymbol{y}|\boldsymbol{K}, \alpha_0, \beta_0) = \mathrm{KL}(q(\boldsymbol{f}, \boldsymbol{f}_m, s)||p(\boldsymbol{f}, \boldsymbol{f}_m, s|\boldsymbol{y}, \boldsymbol{K}, \alpha_0, \beta_0)) - \mathcal{L}. \tag{22}$$

By taking expectations with respect to the variational q distributions, the lower bound is:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f}, \boldsymbol{f}_m, s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)] \\ &= \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|f_{a_p}, f_{b_p})] - \frac{1}{2}\bigg\{ \log|\boldsymbol{K}_{mm}| - \mathbb{E}[\log s] - \log|\boldsymbol{S}| - M \\ &\quad + \hat{\boldsymbol{f}}_m^T \mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\bigg\} + \log\Gamma(\alpha) - \log\Gamma(\alpha_0) + \alpha_0(\log\beta_0) \\ &\quad + (\alpha_0 - \alpha)\mathbb{E}[\log s] + (\beta - \beta_0)\mathbb{E}[s] - \alpha\log\beta, \end{aligned} \tag{23}$$

where the terms relating to $\mathbb{E}[p(\boldsymbol{f}|\boldsymbol{f}_m) - q(\boldsymbol{f})]$ cancel. The iterative algorithm proceeds by updating each of the q factors 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). Hence, we replace Equations 19 and 18 for computing $\hat{\boldsymbol{f}}_m$ and $\boldsymbol{S}$ over all observations with a sequence of stochastic updates.

For the ith update, 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. Rather than using the complete matrices, we perform updates using subsets: $\boldsymbol{Q}_i$ contains rows and columns for observations in $\boldsymbol{P}_i$, $\boldsymbol{K}_{im}$ and $\boldsymbol{A}_i$ contain only rows referred to by $y_p \forall p \in \boldsymbol{P}_i$, $\boldsymbol{G}_i$ contains rows in $\boldsymbol{P}_i$ and columns referred to by $a_p \forall p \in \boldsymbol{P}_i$ and $b_p \forall p \in \boldsymbol{P}_i$, 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):

$$\boldsymbol{S}_i^{-1} = (1-\rho_i)\boldsymbol{S}_{i-1}^{-1} + \rho_i\left(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1} + w_i\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{im}^T\boldsymbol{G}_i^T\boldsymbol{Q}_i^{-1}\boldsymbol{G}_i\boldsymbol{K}_{im}\boldsymbol{K}_{mm}^{-T}\right) \tag{24}$$

$$\begin{aligned}\hat{\boldsymbol{f}}_{m,i} = \boldsymbol{S}_i\Big((1-\rho_i)\boldsymbol{S}_{i-1}^{-1}\hat{\boldsymbol{f}}_{m,i-1} + \\ \rho_i w_i \boldsymbol{K}_{mm}^{-1}\,\boldsymbol{K}_{im}^T\boldsymbol{G}_i^T\boldsymbol{Q}_i^{-1}\left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i\boldsymbol{A}_i\hat{\boldsymbol{f}}_{m,i}\right)\Big)\end{aligned} \tag{25}$$

where $\rho_i = (i + delay)^{-forgettingRate}$ is a mixing coefficient that controls the update rate, $w_i = \frac{P}{|P_i|}$ weights each update according so sample size, and *delay* and *forgettingRate* are hyperparameters of the algorithm (Hoffman et al. 2013), .

The scale parameter, s, can also be learned as part of the SVI procedure. Its variational factor, $q(s)$, has the following update equations:

$$\mathbb{E}[s] = \frac{2\alpha_0 + M}{2\beta} \tag{26}$$

$$\mathbb{E}[\log s] = \Psi(2\alpha_0 + M) - \log(2\beta), \tag{27}$$

where Ψ is the digamma function.

The complete SVI algorithm is summarized in Algorithm 1. The use of an

```
    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 $\boldsymbol{G}_i$ given $\mathbb{E}[\boldsymbol{f}_i]$ ;
5           Compute $\hat{\boldsymbol{f}}_{m,i}$ and $\boldsymbol{S}_i$ ;
6           Compute $\mathbb{E}[\boldsymbol{f}_i]$ ;
        end
7       Update $q(s)$ and compute $\mathbb{E}[s]$ and $\mathbb{E}[\log s]$;
    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 preference learning with a single user.

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, $f(\boldsymbol{x}^*)$:

$$\boldsymbol{f}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m, \tag{28}$$

$$\boldsymbol{C}^* = \boldsymbol{K}_{**} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm}/\mathbb{E}[s])\boldsymbol{K}_{*m}^T\boldsymbol{K}_{mm}^{-1}, \tag{29}$$

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 points. It is possible to recover the lower bound proposed by Hensman et al. (2015) for classification by generalizing the likelihood to arbitrary nonlinear functions, and omitting terms relating to $p(s|\alpha_0, \beta_0$ and $q(s)$. However, our approach avoids expensive quadrature methods by linearizing the likelihood to enable analytical updates. We also infer s in a Bayesian manner, rather than treating as a hyper-parameter, which is important for preference learning where s controls the noise level of the observations relative to f.

4.4 SVI for Crowd Preference Learning

We now extend the SVI method to the crowd preference learning model proposed in Section 3.4. To begin with, we extend the variational posterior in Equation 16 to approximate the crowd model defined in Equation 9.

$$p(\boldsymbol{V}, \boldsymbol{V}_m, \boldsymbol{W}, \boldsymbol{W}_m, \boldsymbol{t}, \boldsymbol{t}_m, s_1, ..., s_C, \sigma | \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{u}_m, k, \alpha_0, \beta_0) \approx \\ q(\boldsymbol{V})q(\boldsymbol{W})q(\boldsymbol{t})q(\boldsymbol{V}_m)q(\boldsymbol{W}_m)q(\boldsymbol{t}_m) \prod_{c=1}^{C} q(s_c)q(\sigma), \quad (30)$$

where $\boldsymbol{u}_m$ are the feature vectors of inducing points for the users. This approximation factorizes the joint distribution between the latent item factors, $\boldsymbol{V}$, the latent user factors, $\boldsymbol{W}$, and the common means, $\boldsymbol{t}$. The variational factors for the inducing points can be obtained by deriving expectations as follows, beginning with the latent item factors:

$$\begin{aligned} \log q(\boldsymbol{V}_m) = & \ \mathbb{E}_{q(\boldsymbol{W}),q(\boldsymbol{t})}[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] \\ & + \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{v}_{m,c}; \boldsymbol{0}, \boldsymbol{K}_{v,mm}/\mathbb{E}[s_c]) + \text{const} \\ = & \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{v}_{m,c}; \hat{\boldsymbol{v}}_{m,c}, \boldsymbol{S}_{v,c}). \end{aligned} \quad (31)$$

where the precision is given by:

$$\boldsymbol{S}_{v,c}^{-1} = \boldsymbol{K}_{v,mm}^{-1}\mathbb{E}[s_c] + \boldsymbol{A}_v^T \boldsymbol{G}^T \text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}}) \boldsymbol{Q}^{-1} \boldsymbol{G} \boldsymbol{A}_v, \quad (32)$$

where $\boldsymbol{A}_v = \boldsymbol{K}_{v,nm}\boldsymbol{K}_{v,mm}^{-1}$, $\hat{\boldsymbol{w}}_c$ and $\boldsymbol{\Sigma}_c$ are the variational mean and covariance of the cth latent user component (defined below in Equations 40 and 39), and the subscript $\boldsymbol{u} = \{u_p \forall p \in 1, ..., P\}$ is the vector of user indexes in the observations, $\boldsymbol{y}$. The term $\text{diag}(\hat{\boldsymbol{W}}_{c,\boldsymbol{j}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{j}})$ scales the diagonal observation precision, $\boldsymbol{Q}^{-1}$, by the latent user factors. We use $\boldsymbol{S}_{v,c}^{-1}$ to compute the variational means for each row of $\boldsymbol{V}_m$ as follows:

$$\hat{\boldsymbol{v}}_{m,c} = \boldsymbol{S}_{v,c} \boldsymbol{A}_v^T \boldsymbol{G}^T \text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}) \boldsymbol{Q}^{-1} \left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \sum_{j=1}^{U} \boldsymbol{H}_j (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \right), \quad (33)$$

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

$$H_{j,p,i} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\mathbb{E}[z_p]))(2y_p - 1)([i = a_p] - [i = b_p])[j = u_p]. \quad (34)$$

This is needed to replace $\boldsymbol{G}$ in the single-user model, since the vector of latent function values, $\boldsymbol{f}$, has been replaced by the matrix $\boldsymbol{F}$, where each column of $\boldsymbol{F}$ corresponds to a single user.

The variational component for the inducing points of the common item mean follows a similar pattern:

$$\begin{aligned}
\log q(\boldsymbol{t}_m) &= \mathbb{E}_{q(\boldsymbol{V})q(\boldsymbol{W})}[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \mathbb{E}[\log \mathcal{N}(\boldsymbol{t}_m; \boldsymbol{0}, \boldsymbol{K}_{t,mm}/s)] + \text{const} \\
&= \log \mathcal{N}\left(\boldsymbol{t}; \hat{\boldsymbol{t}}, \boldsymbol{S}_t\right) && (35) \\
\boldsymbol{S}_t^{-1} &= \boldsymbol{K}_{t,mm}^{-1}/\mathbb{E}[\sigma] + \boldsymbol{A}_t^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} \boldsymbol{G} \boldsymbol{A}_t && (36) \\
\hat{\boldsymbol{t}}_m &= \boldsymbol{S}_t \boldsymbol{A}_t^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} \left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \boldsymbol{G}(\hat{\boldsymbol{t}})\right), && (37)
\end{aligned}$$

where $\boldsymbol{A}_t = \boldsymbol{K}_{t,nm} \boldsymbol{K}_{t,mm}^{-1}$.

Finally, for the latent user factors, $\boldsymbol{W}$, the variational distribution for the inducing points is:

$$\begin{aligned}
\log q(\boldsymbol{W}_m) &= \mathbb{E}_{q(\boldsymbol{V})q(\boldsymbol{t})}[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \sum_{c=1}^{C} \mathbb{E}[\log \mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{K}_{w,mm})] + \text{const} \\
&= \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{w}_c; \hat{\boldsymbol{w}}_c, \boldsymbol{\Sigma}\right), && (38)
\end{aligned}$$

where the variational parameters are:

$$\begin{aligned}
\boldsymbol{\Sigma}_c^{-1} &= \boldsymbol{K}_{w,mm}^{-1} + \boldsymbol{A}_w^T \left(\sum_{p=1}^{P} \boldsymbol{H}_{.,p}^T \text{diag}\left(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 + \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}} \right.\right. \\
&\qquad \left.\left. -2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}} \hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}}\right) \boldsymbol{Q}^{-1} \sum_{p=1}^{P} \boldsymbol{H}_{.,p} \right) \boldsymbol{A}_w && (39) \\
\hat{\boldsymbol{w}}_{m,c} &= \boldsymbol{\Sigma}_c \boldsymbol{A}_w^T \sum_{p=1}^{P} \boldsymbol{H}_{.,p} \left(\text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}) - \text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\right) \boldsymbol{Q}^{-1} \\
&\qquad \left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \sum_{j=1}^{U} \boldsymbol{H}_u (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \right), && (40)
\end{aligned}$$

where the subscripts $._{\boldsymbol{a}} = \{._{a_p} \forall p \in 1, ..., P\}$ and $._{\boldsymbol{b}} = \{._{b_p} \forall p \in 1, ..., P\}$ are lists of indices to the first and second items in the pairs, respectively, and $\boldsymbol{A}_w = \boldsymbol{K}_{w,um} \boldsymbol{K}_{w,mm}^{-1}$.

The equations for the means and covariances can be adapted for stochastic updating by applying weighted sums over the stochastic update and the previous values in the same way as Equation 24 and 25. The stochastic updates for the inducing points of the latent factors depend on expectations with respect to the observed points. As with the single user case, the variational factors at the observed items are independent of the observations given the variational factors of the inducing points (likewise for the observed users):

$$\log q(\boldsymbol{V}) = \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{v}_c; \boldsymbol{A}_v \hat{\boldsymbol{v}}_{m,c}, \frac{\boldsymbol{K}_v}{\mathbb{E}[s_c]} + \boldsymbol{A}_v(\boldsymbol{S}_{m,c} - \frac{\boldsymbol{K}_{v,mm}}{\mathbb{E}[s_c]})\boldsymbol{A}_v\right) \quad (41)$$

$$\log q(\boldsymbol{t}) = \log \mathcal{N}\left(\boldsymbol{t}; \boldsymbol{A}_t \hat{\boldsymbol{t}}_m, \frac{\boldsymbol{K}_t}{\mathbb{E}[\sigma]} + \boldsymbol{A}_t(\boldsymbol{S}_t - \frac{\boldsymbol{K}_{t,mm}}{\mathbb{E}[\sigma]})\boldsymbol{A}_t\right) \quad (42)$$

$$\log q(\boldsymbol{W}) = \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{w}_c; \boldsymbol{A}_w \hat{\boldsymbol{w}}_{m,c}, \boldsymbol{K}_w + \boldsymbol{A}_w(\boldsymbol{\Sigma} - \boldsymbol{K}_{w,mm})\boldsymbol{A}_w\right). \quad (43)$$

The expectations for the inverse scales, $s_1, ..., s_c$ and σ, can be computed using the formulas in Equations 26 and 27 by substituting in the corresponding terms for each $\boldsymbol{v}_c$ or $\boldsymbol{t}$ instead of $\boldsymbol{f}$. Predictions in the latent component model can be made using Equations 41, 42 and 43 by substituting the covariance terms relating to observation items, $\boldsymbol{x}$, and users, $\boldsymbol{u}$, with corresponding covariance terms for the prediction items and users.

As with the single user model, the lower bound on the marginal likelihood contains sums over the observations, hence is suitable for stochastic variational updates:

$$\begin{aligned}\mathcal{L}_{crowd} = & \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p | \boldsymbol{v}_{a_p}^T \boldsymbol{w}_{a_p} + t_{a_p}, \boldsymbol{v}_{b_p}^T \boldsymbol{w}_{b_p} + t_{b_p})] - \frac{1}{2}\Bigg\{ \sum_{c=1}^{C} \Bigg\{ -M_n - M_u \\ & + \log|\boldsymbol{K}_{v,mm}| + \log|\boldsymbol{K}_{w,mm}| - \log|\boldsymbol{S}_{v,c}| - \mathbb{E}[\log s_c] + \hat{\boldsymbol{v}}_{m,c}^T \mathbb{E}[s_c] \boldsymbol{K}_{v,mm}^{-1} \hat{\boldsymbol{v}}_{m,c} \\ & + \mathrm{tr}(\mathbb{E}[s_c]\boldsymbol{K}_{v,mm}^{-1}\boldsymbol{S}_{v,c}) - \log|\boldsymbol{\Sigma}_c| + \hat{\boldsymbol{w}}_{m,c}^T \boldsymbol{K}_{w,mm}^{-1} \hat{\boldsymbol{w}}_{m,c} + \mathrm{tr}(\boldsymbol{K}_{w,mm}^{-1}\boldsymbol{\Sigma}_c) \Bigg\} \\ & - M_n + \log|\boldsymbol{K}_{t,mm}| - \log|\boldsymbol{S}_t| - \mathbb{E}[\log\sigma] + \hat{\boldsymbol{t}}^T \mathbb{E}[\sigma] \boldsymbol{K}_{t,mm}^{-1} \hat{\boldsymbol{t}} \\ & + \mathrm{tr}(\mathbb{E}[\sigma]\boldsymbol{K}_{t,mm}^{-1}\boldsymbol{S}_t) \Bigg\} - (C+1)(\log \mathit{\Gamma}(\alpha_0) + \alpha_0(\log\beta_0)) \\ & + \sum_{c=1}^{C} \Bigg\{ \log \mathit{\Gamma}(\alpha_c) + (\alpha_0 - \alpha_c)\mathbb{E}[\log s_c] + (\beta_c - \beta_0)\mathbb{E}[s_c] - \alpha_c \log\beta_c \Bigg\} \\ & + \log \mathit{\Gamma}(\alpha_\sigma) + (\alpha_0 - \alpha_\sigma)\mathbb{E}[\log\sigma] + (\beta_\sigma - \beta_0)\mathbb{E}[s_c] - \alpha_\sigma \log\beta_\sigma, \end{aligned} \quad (44)$$

In this section, we proposed an SVI scheme for Bayesian matrix factorization given pairwise observations. The inference scheme can readily be adapted to regression or classification tasks by swapping out the preference likelihood, resulting in different values for $\boldsymbol{G}$ and $\boldsymbol{H}$. We now show how to learn the length-scale parameter required to compute covariances using typical kernel functions, then demonstrate how our method can be applied to learning user preferences or consensus opinion when faced with disagreement.

5 Gradient-based Length-scale Optimization

In the previous sections, we defined preference learning models that incorporate GP priors over the latent functions. The covariances of these GPs are defined by a kernel function k, typically of the following form:

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d \left(\frac{|x_d - x'_d|}{l_d}, \boldsymbol{\theta}_d \right) \tag{45}$$

where D is the number of features, l_d is a length-scale hyper-parameter, and $\boldsymbol{\theta}_d$ are additional hyper-parameters for an individual feature kernel, k_d. Each k_d is a function of the distance between the dth feature values in feature vectors $\boldsymbol{x}$ and and $\boldsymbol{x}'$. The product over features in k means that data points have high covariance only if the kernel functions, k_d, for all features are high (a soft AND operator). It is possible to replace the product with a sum, causing covariance to increase for every k_d that is similar (a soft OR operator), or other combinations of the individual feature kernels. The choice of combination over features is therefore an additional hyper-parameter.

The length-scale, l_d, controls the smoothness of the function, k_d, across the feature space and the contribution of each feature to the model. If a feature has a large length-scale, its values, $\boldsymbol{x}$, have less effect on $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ than if it has a shorter length-scale. Hence, it is important to set l_d to correctly capture feature relevance. A computationally frugal option is the median heuristic:

$$l_{d,MH} = D \text{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1, .., N, \forall j = 1, ..., N\}). \tag{46}$$

The motivation is that the median will normalize the feature, so that features are equally weighted regardless of their scaling. By using a median to perform this normalization, extreme values remain outliers with relatively large distances. Multiplying the median by the number of features, D, prevents the average covariance $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ between items from increasing as we add more features using the product kernel in Equation 45. This heuristic has been shown to work reasonably well for the task of comparing distributions (Gretton et al. 2012), but is a simple heursitic with no guarantees of optimality.

An alternative method for setting l_d is Bayesian model selection using the type II maximum likelihood method, which chooses the value of l_d that maximizes the marginal likelihood, $p(\boldsymbol{y}|\boldsymbol{\theta})$. Since the marginal likelihoods for our models are intractable, we maximize the value of the variational lower bound, $\mathcal{L}$, after convergence of the inference algorithm (defined in Equation 23 for a single user, and Equation 44 for the crowd model). Optimizing kernel length-scales in this manner is known as automatic relevance determination (ARD) (Rasmussen and Williams 2006), since the optimal value of l_d depends on the relevance of feature d.

To perform ARD on feature d, we only need to be able to evaluate $\mathcal{L}$ after variational inference has converged with any given value of l_d. However, if we can also compute derivatives of $\mathcal{L}$ with respect to l_d, we can use more efficient gradient-based methods, such as L-BFGS-B (Zhu et al. 1997). These methods perform iterative optimization, using gradients to guide changes for all D length-scales simultaneously. For the single user model, the required gradient with respect to the dth length-scale, l_d, is as follows:

$$\nabla_{l_d}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{f}}_m}\frac{\partial \hat{\boldsymbol{f}}_m}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}^{-1}}\frac{\partial \boldsymbol{S}^{-1}}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial b}\frac{\partial b}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{K}}\frac{\partial \boldsymbol{K}}{\partial l_d}. \tag{47}$$

We exploit S^{-1} The terms involving the variational paramters $\hat{\boldsymbol{f}}_m$, $\boldsymbol{S}$, a and b arise because they depend indirectly on the length-scale through the expectations in the variational factors, $\log q(.)$. However, when the variational inference algorithm has converged, $\mathcal{L}$ is at a maximum, so the partial derivatives of $\mathcal{L}$ with respect to $\hat{\boldsymbol{f}}_m$, $\boldsymbol{S}$, a and b are zero. Hence, after convergence, $\nabla_{l_d}\mathcal{L}$ simplifies to:

$$\nabla_{l_d}\mathcal{L} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\mathbb{E}[s](\hat{\boldsymbol{f}}_m\hat{\boldsymbol{f}}_m^T + \boldsymbol{S}^T)\boldsymbol{K}_{mm}^{-1} - \boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\boldsymbol{K}_{mm}^{-1}\right). \tag{48}$$

For the crowd model, we assume that C latent item components, $\boldsymbol{V}$ have the same kernel function, which is also shared with $\boldsymbol{t}$. The gradients with respect to the length-scale, $l_{w,d}$, for the dth item feature are therefore given by:

$$\nabla_{l_{v,d}}\mathcal{L}_{crowd} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\sum_{c=1}^{C}\mathbb{E}[s_c]\left\{\hat{\boldsymbol{v}}_{m,c}\hat{\boldsymbol{v}}_{m,c}^T + \boldsymbol{S}_{v,c}^T\right\}\boldsymbol{K}_{mm,v}^{-1} - C\boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,v}}{\partial l_{w,d}}\boldsymbol{K}_{mm,v}^{-1}\right)$$
$$+ \frac{1}{2}\mathrm{tr}\left(\left(\mathbb{E}[\sigma](\hat{\boldsymbol{t}}_m\hat{\boldsymbol{t}}_m^T + \boldsymbol{S}_t^T)\boldsymbol{K}_{mm,t}^{-1} - \boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,t}}{\partial l_{w,d}}\boldsymbol{K}_{mm,t}^{-1}\right). \tag{49}$$

The gradients for the dth user feature length-scale, $l_{w,d}$, follows the same form:

$$\nabla_{l_{w,d}}\mathcal{L}_{crowd} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\sum_{c=1}^{C}\left\{\hat{\boldsymbol{w}}_{m,c}\hat{\boldsymbol{w}}_{m,c}^T + \boldsymbol{\Sigma}_c^T\right\}\boldsymbol{K}_{mm,w}^{-1} - C\boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,w}}{\partial l_{w,d}}\boldsymbol{K}_{mm,w}^{-1}\right). \tag{50}$$

The partial derivative of the covariance matrix $\boldsymbol{K}_{mm}$ with respect to l_d depends on the choice of kernel function. The Matèrn $\frac{3}{2}$ function is a widely-applicable, differentiable kernel function that has been shown empirically to outperform other well-established kernels such as the squared exponential, and makes weaker assumptions of smoothness of the latent function (Rasmussen and Williams 2006). It is defined as:

$$k_d\left(\frac{|x_d - x_d'|}{l_d}\right) = \left(1 + \frac{\sqrt{3}|x_d - x_d'|}{l_d}\right)\exp\left(-\frac{\sqrt{3}|x_d - x_d'|}{l_d}\right). \tag{51}$$

Assuming that the kernel functions for each feature, k_d, are combined using a product, as in Equation 45, the partial derivative $\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}$ is a matrix, where each entry, i, j, is defined by:

$$\frac{\partial K_{mm,ij}}{\partial l_d} = \prod_{d'=1,d'\neq d}^{D} k_{d'}\left(\frac{|x_{d'} - x_{d'}'|}{l_{d'}}\right)\frac{3(\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d})^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|}{l_d}\right), \tag{52}$$

Dataset	Folds /subs-amples	Users	Items	Pairs, train	Pairs, test	Pref vals, test	Item fea-tures	User fea-tures
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	900	0	100	2	2
Simulation (d)	25	25	100	36 - 2304	0	100	2	2
Sushi A	25	1000	10	15000	5000	10000	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, generate the subsamples of data independently, for the Sushi dataset we select subsamples independently from the complete dataset, while UKPConvArgCrowd-Sample 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.

where we assume the use of Equation to combine kernel functions over features using a product

To make use of Equations 48 to 52, we nest the variational algorithm defined in Section 4 inside an iterative gradient-based optimization method. Optimization then begins with an initial guess for all length-scales, l_d, such as the median heuristic. Given the current values of l_d, the optimizer (e.g. L-BFGS-B) runs the VB algorithm to convergence, computes $\nabla_{l_d}\mathcal{L}$, then proposes a new candidate value of l_d. The process repeats until the optimizer converges or reaches a maximum number of iterations, and returns the value of l_d that maximized $\mathcal{L}$.

6 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 crowd-GPPL using SVI. In Section 6.2, we use simulated data to test the ability of our method to recover preference functions from noisy data when the correct number of latent factors is unknown. Section 6.3 evaluates our approach on an NLP task with high-dimensional feature vectors and a larger number of items, which involves using pairwise judgments of arguments from online debate forums to learn a function of argument *convincingness*. We use the *UKPConvArgSample*, which is sampled from data provided by Habernal and Gurevych (2016) dataset. This section first analyzes the scalability of our SVI approach, then its performance in predicting preferences. Finally, in Section 6.4, we compare our method against previous approaches for predicting the preferences of thousands of users on the *Sushi* datasets (Kamishima 2003).

6.1 Methods Compared

We refer to the multi-user variant of our model as *crowd-GPPL*. As baselines, we use GPPL to learn a single preference function from all users' preference labels, (*pooled-GPPL*), and a Gaussian process over the joint feature space of users and items (*joint-GPPL*), as proposed by Guo et al. (2010). For datasets up to 100 users (simulated data, subsamples of the real datasets), we also test separate GPPL instances per user with no collaborative learning (*GPPL-per-user*), but this could not be applied to the real datasets as the computation costs were too high. To test the benefit of using GPs to model item and user features, we also test two further baselines: *crowd-GPPL*$\backslash\boldsymbol{u}$, which ignores the user features, and *crowd-BMF*, which ignores both user and item features and so does not use GPs at all. For both of these methods, the user covariance matrix, $\boldsymbol{K}_w$, in the crowd-GPPL model is replaced by the identity matrix, and for *crowd-BMF*, the item covariance matrices, $\boldsymbol{K}_v$ and $\boldsymbol{K}_t$ are also replaced by the identity matrix.

6.2 Simulated Noisy Data

First, we test how well GPPL is able to recover an underlying consensus function in the presence of varying amounts of noise.

We generate data by first selecting 100 points at random from a $10x10$ 2-dimensional grid and choosing 500 pairs of these points at random. We generate pairwise labels by drawing from the single user GPPL model: first, draw latent preference function values for the selected points; then, compute the pairwise likelihoods for the selected pairs using Equation 3; draw pairwise labels from the Bernoulli likelihoods. We split the chosen points into 50% training and test sets, and train GPPL on all pairs involving only points in the training set. We then use GPPL to latent predict preference function for the points in the test set. We repeat this process, varying the value of s in the generation step, which controls the precision of a latent preference function: as s is increased, the latent function values have a smaller amplitude and the pairwise labels become noisier. The hyper-parameters of the GPPL model for prediction remain the same throughout, with $\alpha_0 = 2$, $\beta_0 = 2$. We repeat the complete experiment 25 times, including generating new data for each value of s.

The results of the first test in Figure 1a show that increasing the noise rate in the pairwise labels causes a near-linear decrease in the rank correlation between the predicted and true preference function values. Nonetheless, GPPL is able to recover the ranking of points with $\tau > 0.5$ when more than 1/3 pairwise labels are incorrect.

In the second simulation, we recover a consensus function from preference labels produced by multiple simulated users with varying differences in their individual preferences. We use the same process as the first simulation, except we now draw the pairwise labels from a crowd-GPPL model with 25 users and 3 latent factors, instead of a single-user model. We fix the inverse scale of the latent item factors, $s = 0.001$, but vary the precision of the consensus function, σ. We use three methods to recover the consensus function, GPPL-per-user, pooled-GPPL, and crowd-GPPL with $C = 25$ so that there is one factor per user. Figure 1b shows that the crowd-GPPL model is better able recover the latent consensus function than the other methods, even when noise levels are high. The pooled model'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 model.

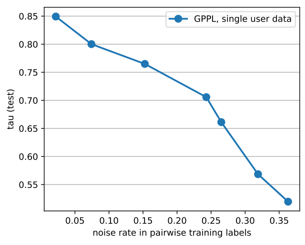
The third test repeats a similar setup to the second, but here we evaluate the methods' ability to to recover the personal preferences of individual simulated users. We fix $\sigma = 10$ and vary the precision of the item latent factors, s. Results for predicting personal preferences in the presence of noise are shown in Figure 1c. Crowd-GPPL is able to make better predictions when noise is below 0.3 but its benefit disappears when the noise level increases further.

In the final simulation, we evaluate the effect of the quantity of training data in scenarios with different numbers of latent factors. We hypothesized that a larger number of latent factors would indicate a more complex scenario that would require more training data. We generate data again from the crowd-GPPL model with $s = 0.2$ and $\sigma = 1$ and vary the number of true latent factors using values $C_{true} \in \{1, 3, 10, 20\}$. For each value of C_{true}, we run crowd-GPPL with increasing numbers of pairwise labels, and evaluate the correlation between inferred and true latent user factors. To match inferred factors to true factors, we compute Pearson correlations between each true factor and each inferred factor, then repeatedly select the pair of unmatched factors with the highest correlation as a new match until all true factors are matched.

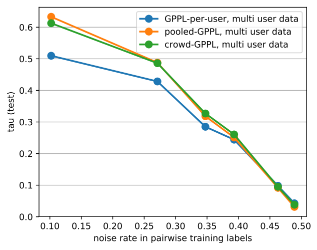
The results in Figure 1d show that for all values of C_{true}, increasing the number of training labels beyond 1000 does not increase the correlations between the best-matched latent user factors, and in fact the correlations decrease. Since the crowd-GPPL model is able to learn $C = 25$ factors, it may learn a more complex model that makes more use of a greater number of latent factors given more training data. In this case, the correlations may decrease as the true factors would be modeled by multiple inferred factors, rather than just the best match. This may also explain why the correlations are higher when $C_{true} = 20$, as the number of true factors is much closer to C. However, even when there is a large difference between C_{true} and C, the correlation remains above 0.4, showing that the model is able to infer reasonable latent factors even when C is set too high.

6.3 Argument Convincingness: Scalability

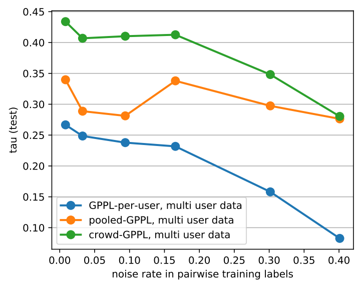
We evaluate our preference learning approaches on an NLP dataset containing $32,310$ different features for arguments written by users of online debating forums Habernal and Gurevych (2016). The task is to quantify how convincing



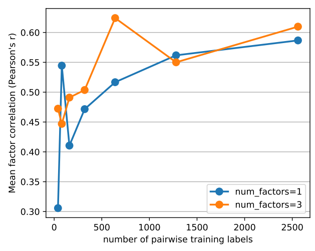
(a) Inferring preferences for a single user



(b) Inferring consensus function from crowdsourced labels



(c) Inferring personal preferences for members of a crowd



(d) Inferring the latent factors

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

each argument is by learning a model from pairwise preference labels obtained from crowdworkers on Amazon Mechanical Turk. The workers were presented with pairs of arguments and asked to choose which argument was more convincing. The dataset is divided into 32 parts, each corresponding to one of 16 topics and one of two opposing stances. We test the ability of the preference learning methods to predict the consensus by training on raw crowdsourced pairwise labels for 31 topics, and testing against the gold pairwise labels and rankings for the remaining topic. This process is repeated for all 32 topics. This dataset, *UKPConvArgCrowdSample*, was obtained in (Simpson and Gurevych 2018) by subsampling the raw crowdsourced labels provided by (Habernal and Gurevych 2016), and using the gold pairwise labels and rankings obtained in (Habernal and Gurevych 2016).

We assess our proposed SVI inference method by testing pooled-GPPL and crowd-GPPL with different numbers of inducing points, M. Figure 3 shows the trade-off between runtime and accuracy as an effect of choosing M. Accuracy close to the peak is attained using $M = 200$, after which the accuracy levels off, while the runtime increases rapidly as M increases. With 300 features, the polynomial training time complexity is visible in the runtime. However, with $33,210$ features, the runtime plot appears almost linear, since the cost

(a) Varying number of items in training set. GloVe features.
(b) Varying number of features. ling+GloVe features. $M = 500$.

Fig. 2 Runtimes for training+prediction on UKPConvArgCrowdSample with varying subsample size. Means over 32 runs. Note the logarithmic x-axis for (b).

of computing covariance matrices, which is linear in the number of features, dominates the runtimes. The plots show that the SVI method provides a substantial cut in runtimes while maintaining good prediction accuracy.

Figure 2a, we compare the training + prediction runtimes of different methods as a function of the training set size, N_tr. With a fixed M, runtimes increase very little with N_tr, as other overheads are inexpensive. The methods labeled "no SVI" show runtimes for pooled-GPPL and crowd-GPPL with variational inference but no stochastic updates or inducing points. These are compared with Bi-LSTM and SVM classifiers trained to output probabilities of pairwise labels. The plots clearly show the rapid increases in runtimes for these alternative methods. In Figure 2b, we plot the number of features against runtimes for each method on the whole dataset, with $M = 500$. For pooled-GPPL and crowd-GPPL, the cost of kernel computations becomes visible only with $33,210$ features, indicating the benefits of more compact representations. BiLSTM appears unaffected by additional input dimensions, while the SVM runtimes increase noticably from 30 to 300 and 3000 features.

The pooled-GPPL method was previously tested on *UKPConvArgCrowdSample* in (Simpson and Gurevych 2018), and shown to outperform SVM, Bi-LSTM and Gaussian process classifier methods. In this paper, we compare pooled-GPPL with crowd-GPPL and also test each method's ability to predict the raw crowdsourced labels, i.e. the individual preference labels supplied by each worker. We hypothesize that this task is subjective a worker's view of convincingness depends somewhat on their prior beliefs and understanding of the subject discussed and the language used. If this is the case, then provided that the data is sufficiently informative, the crowd-GPPL model may more accurately predict unseen pairwise labels or rankings for individual workers, and may be better able to predict the consensus preference function by accounting for the biases of individual workers.

The performance metrics are shown in Table 2....

(a)(b)
33210
Gling+GloVe
ffea-
ttures

Fig. 3 Effect of varying M on accuracy and runtime (training+prediction) for GPPL and crowd-GPPL on UKPConvArgCrowdSample. Means over 32 runs.

	Consensus			Personal		
Method	Acc	CEE	Kend.	Acc	CEE	Kend.
GPPL medi.	**.77**	**.50**	**.40**	.71	.56	.07
GPPL opt.				.70	.58	.06
Crowd-GPPL medi.	.51	.69	.01	.71	.69	.06
Crowd-GPPL opt.	.51	.69	.01	.70	.69	.06

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 *Kend.* shows Kendall's tau for the predicted preference function.

(a)(b)
USKsPhi-
CAon-
vaAnd
rgSushowd-
SBam-
ple

Fig. 4 Distribution of latent factor variances, $1/s_c$, for crowd-GPPL on UKPConvArgCrowdSample, Sushi-A and Sushi-B, averaged over all runs.

We investigate whether how many latent factors were actively used by the crowd-GPPL model by plotting the inferred variance scales, s_c, for the latent item factors in Figure 4. The plots show that many factors have a very small variance and therefore do not contribute strongly to the model's predictions. This indicates that our Bayesian approach, in which the priors of the latent factors have mean zero, preferred a simpler model even when a larger number of latent factors was available.

6.4 Sushi Preferences

We use two datasets, Sushi-A and Sushi-B (shown in Table 1), to benchmark the classification and ranking performance of GPPL and crowd-GPPL, as well as their runtimes, 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 pairwise labels. These labels can be considered noise-free, since they are derived directly from the gold standard ranking. For Sushi-A, we select a subset of 1000 workers at random, then split the data into training and test sets by

	Sushi-A-small				**Sushi-A**				**Sushi-B**			
Method	Acc	CE E	τ	Run-time	Acc	CE E	τ	Run-time	Acc	CE E	τ	Run-time
crowdPL-												
full	.67	.87	.39	76.91 (27.23)	.79	.51	.66	149.6 (20.86)				
full, opt.					.80	.48	.68	3718.20 (2193.72)				
\induc	.70	**.57**	.46	165.72 (14.81)	.84	**.33**	.79	315.17 (50.17)				
\cons	.67	.80	.39	7.45 (29.00)	.79	.57	.65	158.51 (25.06)				
\u	.70	.58	.46	174.72 (24.62)	.84	**.33**	**.80**	336.42 (4.93)				
\u, opt.					**.85**	**.33**	**.80**	5015.98 (1978.82)				
\u\x	**.71**	**.57**	**.49**	4.67 (3.74)	**.85**	**.33**	**.80**	328.83 (12.12)				
\u, FITC	.70	.58	.46	333.68 (57.36)	**.85**	**.33**	**.80**	551.56 (16.82)				
\u\ cons, FITC	.68	.60	.43	291.24 (48.62)	.84	**.33**	.79	540.08 (12.15)				
\u\ cons, FITC					.84	**.33**	**.80**	7413.53 (2849.63)				
pooled	.65	.62	.31	**1.54** (.26)	.66	.62	.32	**1.83** (.38)				
per-user	.67	.64	.42	97.47 (1.12)	.83	.40	.79	67.23 (.44)				
Houlsby et al CP					.84							
Houlsby et al CPU					.83							

Table 3 Predicting personal preferences: performance on Sushi-A dataset and Sushi-B datasets. Runtimes given in seconds, with standard deviation between repeats in brackets. For accuracy, all standard deviations are ≤ 0.02, for CEE ≤ 0.08, for Kend. ≤ 0.03.

randomly selecting 15 pairs for each user for training and 5 for testing. For Sushi-B, we use all 5000 workers, and subsample 10 training pairs and 1 test pair per user. We evaluate the quality of pairwise predictions using classification accuracy and cross entropy error (also known as log loss) to gauge the quality of the probabilities that each method outputs. We also evaluate each method's inferred personal preference values against the gold standard rankings by computing Kendall's τ rank correlation coefficient between the negative rank and the inferred preference function values, for all the items contained in each subsampled user's ranking. The complete process, including subsampling, was repeated 25 times.

The results in Table 3 illustrate the benefit of the crowd model over single-user GPPL. For the methods of (Houlsby et al. 2012) and (Khan et al. 2014) we re-state the previous performance metrics and did not re-implement these

methods. The runtimes (combining both training and prediction) show that including feature data for items and users does not noticably increase computational costs of crowd-GPPL over crowd-GPPL
u or crowd-BMF. Likewise, the use of matrix factorization leads to only a small increase in runtimes of these methods over joint-GPPL. However, the runtimes for crowd-GPPL are higher than those of pooled-GPPL, as are the performance metrics. Crowd-GPPL produces similar classification scores to the earlier method of (Houlsby et al. 2012), while scaling far better with the number of users and items than methods that do not use inducing points, as seen in Figure 2a. Ranking performance is also improved over that of (Khan et al. 2014), which uses a GP for each user in combination with BMF. GPPL-per-user does not perform as well as (Khan et al. 2014), illustrating the benefit of learning latent factors. Ignoring the user features and especially the item features decreases performance of crowd-GPPL, as reflected in the lower scores of crowd-GPPL
u and crowd-BMF. Figure 4 again shows that the inferred crowd-GPPL model relies heavily on a subset of the latent factors.

7 Conclusions and Future Work

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: IJCAI, pp 1213–1219

Adams RP, Dahl GE, Murray I (2010) Incorporating side information in probabilistic matrix factorization with gaussian processes. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, AUAI Press, pp 1–9

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

Bolgár BM, Antal P (2016) Bayesian matrix factorization with non-random missing data using informative gaussian process priors and soft evidences. Journal of Machine Learning Research 52:25–36

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

Busa-Fekete R, Hüllermeier E, El Mesaoudi-Paul A (2018) Preference-based online learning with dueling bandits: A survey. arXiv preprint arXiv:180711398

Cai C, Sun H, Dong B, Zhang B, Wang T, Wang H (2017) Pairwise ranking aggregation by non-interactive crowdsourcing with budget constraints. In: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, IEEE, pp 2567–2568

Chen G, Zhu F, Heng PA (2018) Large-scale bayesian probabilistic matrix factorization with memo-free distributed variational inference. ACM Trans Knowl Discov Data 12(3):31:1–31:24, DOI 10.1145/3161886, URL http://doi.acm.org/10.1145/3161886

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

Ding N, Qi Y, Xiang R, Molloy I, Li N (2010) Nonparametric bayesian matrix factorization by power-ep. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp 169–176

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

Gretton A, Sejdinovic D, Strathmann H, Balakrishnan S, Pontil M, Fukumizu K, Sriperumbudur BK (2012) Optimal kernel choice for large-scale two-sample tests. In: Advances in Neural Information Processing Systems, pp 1205–1213

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, Berlin, Germany, 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

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

Khan ME, Ko YJ, Seeger MW (2014) Scalable collaborative bayesian preference learning. In: AISTATS, vol 14, 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

Kim Y, Kim W, Shim K (2017) Latent ranking analysis using pairwise comparisons in crowdsourcing platforms. Information Systems 65:7–21

Li J, Baba Y, Kashima H (2018) Simultaneous clustering and ranking from pairwise comparisons. In: IJCAI

Luce RD (1959) On the possible psychophysical laws. Psychological review 66(2):81

Maystre L, Grossglauser M (2017) Just sort it! a simple and effective approach to active preference learning. In: International Conference on Machine Learning, pp 2344–2353

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

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

Qian L, Gao J, Jagadish H (2015) Learning user preferences by adaptive pairwise comparison. Proceedings of the VLDB Endowment 8(11):1322–1333

Radlinski F, Joachims T (2007) Active exploration for learning rankings from clickthrough data. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 570–579

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

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

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

Shi J, Zheng X, Yang W (2017) Survey on probabilistic models of low-rank matrix factorizations. Entropy 19(8):424

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

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

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

Tian Y, Zhu J (2012) Learning from crowds in the presence of schools of thought. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD ’12, pp 226–234, DOI 10.1145/2339530.2339571, URL `http://doi.acm.org/10.1145/2339530.2339571`

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

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

Zhou T, Shan H, Banerjee A, Sapiro G (2012) Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In: Proceedings of the 2012 SIAM international Conference on Data mining, SIAM, pp 403–414

Zhu C, Byrd RH, Lu P, Nocedal J (1997) Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS) 23(4):550–560