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

Scalable Bayesian Preference Learning with Crowds

Edwin Simpson · Iryna Gurevych

Received: date

Abstract We propose a scalable Bayesian preference learning method for inferring a consensus from crowdsourced pairwise labels as well as 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 data for each user also present 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 computational and memory costs. Our experiments on a computational argumentation task demonstrate the method's scalability and show that modeling preferences of individual annotators in a crowd improves the quality of an inferred gold standard. On a recommendation task, accuracy is competitive with previous methods while the model is able to scale to far larger datasets. We also show how to apply gradient-based optimization to length-scale hyper-parameters to improve performance. 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). Here, we focus on ranking items and predicting the item with the highest value

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

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

in a pair. For many preference learning tasks, annotators have divergent opinions on the correct labels, 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 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, solutions are required for modeling individual preferences given limited data per user and producing gold standard labels from crowds of annotators with different opinions.

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 preference 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, such *implicit annotations* can be noisy, as are crowdsourced preference labels (Habernal and Gurevych 2016). Therefore, while pairwise labels provide a valuable form of training data, preference learning methods must be able to to aggregate noisy sources of pairwise data from crowds or from different types of implicit annotation. 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 sources – in this case, 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, and multiplying the low-dimensional representations predicts ratings for unseen user-item pairs. However, traditional approaches are not able to handle pairwise labels, nor extrapolate to new users or items.

The limited amount of 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, for example, has been shown to benefit from a Bayesian treatment when data is sparse (Salakhutdinov and Mnih 2008). Previous work has introduced a Bayesian approach for combining crowdsourced preferences, but this does models only ground truth rather than individual preferences and cannot make predictions for new users or items (Chen et al. 2013). For 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 make predictions on test items using a Bayesian framework (Chu and Ghahramani 2005; Houlsby et al. 2012; Khan et al. 2014), but they do not scale to large numbers of items, users, or pairwise labels as computational and memory complexity is polynomial in the number of items and pairs, and linear or worse in the number of users.

In this paper, we propose a scalable Bayesian approach to pairwise preference learning with crowds, whose members may be annotators, users or sources of implicit annotations. We introduce a model that captures both 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 or 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 personal preferences as well as an objective gold-standard given crowdsourced data.

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 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, assuming that these provide 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), using a Bayesian approach that learns the individual accuracy of each worker. Wang et al. (2016) improved performance by modeling the level of noise in the latent utility function for each annotator in a given domain, rather than in the pairwise labels. However, neither Chen et al. (2013) nor Wang et al. (2016) exploits 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 annoators. To crowdsource sentiment annotations, Kiritchenko and Mohammad (2016) propose to use an extension of pairwise comparions 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, these approaches do not model divergence of opinion between annotators and do not provide a Bayesian solution. Related works have also investigated budget constraints for crowdsourcing pairwise labels (Cai et al. 2017). In summary, previous work does not provide a Bayesian approach for aggregating pairwise labels from crowds that can make predictions for new items and model the divergence of opinions between annotators.

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. This task has previously been addressed by Yi et al. (2013) and Kim et al. (2014), who learn multiple latent rankings and infer the preference of each user toward those rankings. Salimans et al. (2012) use Bayesian matrix factorization to identify latent rankings, but none of these approaches exploits item features to remedy labeling errors or generalize to new test items. In contrast, Guo et al. (2010) propose a Gaussian process approach that learns over a joint space of users and features. However, this scales cubically in the number of users, hence 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) learn GP for each user, and combine them with matrix factoriza-

tion to perform collaborative filtering. However, this approach does not model the relationship between input features and the latent factors, and does not place a prior over item factors. 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, more scalable approaches to inference are needed to permit the use of such models with datasets containing thousands of items or users.

2.3 Scalable Approximate Bayesian 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. 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 sparse *generalized fully independent training conditional* (GFITC) approximation (Snelson and Ghahramani 2006) to reduce the computational complexity to $\mathcal{O}(NM^2)$ and the memory complexity to $\mathcal{O}(NM)$, but this is not sufficiently scalable for very large numbers of items, users or pairs. Houlsby et al. (2012) also introduce a kernel for pairwise preference learning and therefore place a sparse GP over pairs, rather than items. This means that the inducing points M have to stand in for P, which is typically larger than the number of items, N. A GP over the pairs also makes it difficult to extract posteriors for latent function values for individual items, and prevents mixing pairwise training labels with observed ratings, which may be necessary when aggregating multiple sources of annotations.

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 (Simpson and Gurevych 2018). This paper provides the first full derivation of this approach, including showing how to learn the observation noise as part of the variational approach. Other recent work

on scalable inference for Bayesian matrix factorization focuses on distributing and parallelizing inference rather than reducing total costs, but is not directly applicable to Gaussian processes (Ahn et al. 2015; Vander Aa et al. 2017; Chen et al. 2018). As our method includes Bayesian matrix factorization (BMF) as part of the model, we believe this paper is the first to apply SVI to BMF.

3 Bayesian Preference Learning for Crowds

For a pair of items, a and b, we write $a \prec b$ to symbolize that a is preferred to b. A pairwise label has value $y(a \succ b) = 1$ if $a \prec b$ 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 or noise in implicit annotations, such as click streams, as well as variability in a user's judgments. Here, we assume the utility to be a function of the items' features, $f(\boldsymbol{x}_a)$, where $\boldsymbol{x}_a$ is a vector representation of the features of item a. Pairwise labels are typically modeled using either 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, $\delta \sim \mathcal{N}(0, 0.5)$:

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

Integrating out the unknown values of δ_a and δ_b gives:

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

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. 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 $f(\boldsymbol{x}_b)$. We now require a model for the utility function f, which we introduce in the next section.

3.1 Single User Preference Learning

We assume that the utility function, f, 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 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 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. 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). 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:

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

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)\}$ are the utilities of the N items referred to by $\boldsymbol{y}$.We henceforth refer to this single-user model as *GPPL*.

3.2 Crowd Preference Learning

When there are multiple users, we wish to exploit similarities between their utility functions to improve our predictions for each user when faced with sparse data. We represent utilities in a matrix, $\boldsymbol{F}$, 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, $\boldsymbol{W} \in \mathcal{R}^{C \times U}$, and one for the items, $\boldsymbol{V} \in \mathcal{R}^{N \times C}$, where C is the number of latent components: $\boldsymbol{F} = \boldsymbol{V}^T \boldsymbol{W}$. The rows V_a and W_j are latent vector representations of items and users, 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, and C latent functions of user features, w_c. The matrices V and W are evaluations of these functions at the points corresponding to the users and items observed during training. Therefore, the latent preference function, f, is a weighted sum over latent functions:

$$f(\boldsymbol{x}_a, \boldsymbol{u}_j) = \sum_{c=1}^{C} w_c(\boldsymbol{u}_j) v_c(\boldsymbol{x}_a), \qquad v_c \sim \mathcal{GP}(\mathbf{0}, k_\theta / s_c), \qquad w_c \sim \mathcal{GP}(\mathbf{0}, k_\theta), \quad (5)$$

where $\boldsymbol{u}_j$ are the features of user j, and we provide a Bayesian treatment to matrix factorization by placing Gaussian process priors over the latent functions. 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 Bayesian approach to avoid 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 or label sources. Since our goal is to infer a consensus from a crowd as well as to model individual users' preferences, we also 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. The joint distribution of our crowd model, *crowdGPPL*, is:

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

$$\mathcal{G}(\sigma_t; \alpha_0, \beta_0) \prod_{c=1}^{C} \left\{ \mathcal{N}(\boldsymbol{v}_c; \mathbf{0}, \boldsymbol{K}_{v,\theta} / s_c) \mathcal{N}(\boldsymbol{w}_c; \mathbf{0}, \boldsymbol{K}_{w,\theta}) \mathcal{G}(s_c; \alpha_0, \beta_0) \right\}, \quad (6)$$

$$\text{where } z_p = \boldsymbol{v}_{\cdot,a_p}^T \boldsymbol{w}_{\cdot,u_p} + t_{a_p} - \boldsymbol{v}_{\cdot,b_p}^T \boldsymbol{w}_{\cdot,u_p} - t_{b_p}, \quad (7)$$

and σ_t is the inverse scale of t. The index p now refers to a tuple, $\{u_p, a_p, b_p\}$, that 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 tasks such as 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 $\mathcal{O}(N^2)$. 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 $\mathcal{O}(NM^2)$ computational and $\mathcal{O}(NM)$ costs become prohibitively high when N is large, and GFITC is not amenable to distributed computation (Hensman et al. 2015). In contrast to crowdGPPL, Houlsby et al. (2012) place GPs over the space of pairs rather than items, which is typically much larger, meaning that $\mathcal{O}(PM^2+UM^2)$ computational and $\mathcal{O}(PM+UM)$ memory costs dominate. We derive a more scalable approach for GPPL and crowdGPPL using stochastic variational inference (SVI), an iterative scheme that allows the computational and memory costs at each iteration to be constrained (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:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathbb{E}\left[\prod_{p=1}^{P} \Phi(z_p)\right] = \prod_{p=1}^{P} \Phi(\hat{z}_p) \approx \mathcal{N}(\boldsymbol{y}; \Phi(\hat{\boldsymbol{z}}), \boldsymbol{Q}), \tag{8}$$

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:

$$\begin{aligned} Q_{p,p} &= \mathbb{E}_f[\Phi(z_p)(1-\Phi(z_p))] \\ &\approx \frac{(y_p+\gamma_0)(1-y_p+\lambda_0)}{(\gamma_0+\lambda_0+1)} - \frac{(y_p+\gamma_0)(1-y_p+\lambda_0)}{(\gamma_0+\lambda_0+1)^2(\gamma_0+\lambda_0+2)}, \end{aligned} \tag{9}$$

where γ_0 and λ_0 are parameters of a beta distribution with the same variance as the prior $p(\Phi(z_p)|\boldsymbol{K}_\theta, \alpha_0, \beta_0)$ 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 single user GPPL (for crowdGPPL, replace $\boldsymbol{f}$ with $\boldsymbol{F}$ in the following):

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

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

where $\boldsymbol{G}$ is a matrix containing elements $G_{p,i}$, which are the partial derivatives of the pairwise likelihood with respect to each of the latent function values, $\boldsymbol{f}$. This creates a 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 describe below. The 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). In the next section, we use the approximate likelihood to define an approximate posterior for stochastic variational inference (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, and provide further detailed equations in Appendix A. The posterior over the inducing and training items is approximated as:

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

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

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

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:

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

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:

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

where $\alpha = \alpha_0 + \frac{M}{2}$ and $\beta = \beta_0 + \frac{\text{tr}(\boldsymbol{K}_{mm}^{-1}(S + \hat{\boldsymbol{f}}_m\hat{\boldsymbol{f}}_m^T))}{2}$. None of the other factors depend on $\log q(\boldsymbol{f})$, so it need not be computed unless required for prediction.

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 (Equation 12) by maximizing a lower bound, $\mathcal{L}$, on the log marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{K}, \alpha_0, \beta_0) = \text{KL}(q(\boldsymbol{f}, \boldsymbol{f}_m, s)||p(\boldsymbol{f}, \boldsymbol{f}_m, s|\boldsymbol{y}, \boldsymbol{K}, \alpha_0, \beta_0)) + \mathcal{L}. \tag{16}$$
$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{f}, \boldsymbol{f}_m, s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)]$$

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 $\boldsymbol{Q}_i$, which contains rows and columns of $\boldsymbol{Q}$ for observations in $\boldsymbol{P}_i$, $\boldsymbol{K}_{im}$ and $\boldsymbol{A}_i$, which contain only rows referred to by $\{y_p \forall p \in \boldsymbol{P}_i\}$, $\boldsymbol{G}_i$, which contains rows 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

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.

by following the natural gradient (Hensman et al. 2015):

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

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

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 so 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 15, 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, $f(\boldsymbol{x}^*)$:

$$\boldsymbol{f}^* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m, \tag{19}$$

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

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

4.3 SVI for CrowdGPPL

We now extend the SVI method to the crowd preference learning model proposed in Section 3.2. To begin with, we extend the variational posterior in

Equation 12 to approximate the crowd model defined in Equation 7:

$$p(\boldsymbol{V}, \boldsymbol{V}_m, \boldsymbol{W}, \boldsymbol{W}_m, \boldsymbol{t}, \boldsymbol{t}_m, s_1, ..., s_C, \sigma | \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{u}_m, k, \alpha_0, \beta_0) \approx$$
$$q(\boldsymbol{V})q(\boldsymbol{W})q(\boldsymbol{t})q(\boldsymbol{V}_m)q(\boldsymbol{W}_m)q(\boldsymbol{t}_m)\prod_{c=1}^{C} q(s_c)q(\sigma), \tag{21}$$

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 49 in the Appendix):

$$\mathcal{L}_{cr} = \mathbb{E}_q \Bigg[\log p(\boldsymbol{y}|\boldsymbol{F}) + \sum_{c=1}^{C} \{\log p(\boldsymbol{v}_{m,c}, s_c | \boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \log q(\boldsymbol{v}_{m,c})$$
$$+ \log p(\boldsymbol{w}_{m,c}|\boldsymbol{K}_{w,mm}) - \log q(\boldsymbol{w}_{m,c})\} + \log p(\boldsymbol{t}_m, \sigma | \boldsymbol{K}_{mm}, \alpha_0, \beta_0) - \log q(\boldsymbol{t}_m)] \,. \tag{22}$$

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 $\boldsymbol{f}_m$ using the equations given below.

The expectations for the inverse scales, $s_1, ..., s_c$ and σ, can be computed using Equation 15 by substituting in the corresponding terms for each $\boldsymbol{v}_c$ or $\boldsymbol{t}$ instead of $\boldsymbol{f}$. The variational factor for the inducing item factors is:

$$\log q(\boldsymbol{V}_m) = \mathbb{E}\left[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{v}_{m,c}; \boldsymbol{0}, \frac{\boldsymbol{K}_{v,mm}}{\mathbb{E}[s_c]}\right) + \text{const}$$
$$= \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{v}_{m,c}; \hat{\boldsymbol{v}}_{m,c}, \boldsymbol{S}_{v,c}). \tag{23}$$

The precision estimate for $\boldsymbol{S}_{v,c}^{-1}$ at iteration i is given by:

$$\boldsymbol{S}_{v,c,i}^{-1} = (1-\rho_i)\boldsymbol{S}_{v,c,i-1}^{-1} + \rho_i \left(\boldsymbol{K}_{v,mm}^{-1}\mathbb{E}[s_c]\right.$$
$$\left. + w_i \boldsymbol{A}_{v,i}^T \boldsymbol{G}_i^T \text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}})\boldsymbol{Q}_i^{-1}\boldsymbol{G}_i\boldsymbol{A}_{v,i}\right), \tag{24}$$

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 32 and 31), 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 $\boldsymbol{V}_m$:

$$\hat{\boldsymbol{v}}_{m,c,i} = \boldsymbol{S}_{v,c,i}\left((1-\rho_i)\boldsymbol{S}_{v,c,i-1}^{-1}\hat{\boldsymbol{v}}_{m,c,i-1} + \rho_i w_i \left(\boldsymbol{S}_{v,c,i}\boldsymbol{A}_i^T\boldsymbol{G}_i^T \text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}})\boldsymbol{Q}_i^{-1}\right.\right.$$
$$\left.\left.\left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_j^{(i)}(\hat{\boldsymbol{v}}_c^T\hat{\boldsymbol{w}}_{c,j})\right)\right)\right), \tag{25}$$

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:

$$H^{(i)}_{j,p,n} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\mathbb{E}[z_p]))(2y_p - 1)([n = a_p] - [n = b_p])[j = u_p]. \quad (26)$$

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

$$\begin{aligned}
\log q(\boldsymbol{t}_m) = &\ \mathbb{E}[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \mathbb{E}[\log \mathcal{N}(\boldsymbol{t}_m; \boldsymbol{0}, \boldsymbol{K}_{t,mm}/s)] + \text{const} \\
= &\ \log \mathcal{N}\left(\boldsymbol{t}; \hat{\boldsymbol{t}}, \boldsymbol{S}_t\right) && (27) \\
\boldsymbol{S}_{t,i}^{-1} = &\ (1-\rho_i)\boldsymbol{S}_{t,i-1}^{-1} + \rho_i \boldsymbol{K}_{t,mm}^{-1}/\mathbb{E}[\sigma] + \rho_i w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1} \boldsymbol{G}_i \boldsymbol{A}_i && (28) \\
\hat{\boldsymbol{t}}_{m,i} = &\ \boldsymbol{S}_{t,i}\left((1-\rho_i)\boldsymbol{S}_{t,i-1}^{-1}\hat{\boldsymbol{t}}_{m,i-1}\right. \\
&\left. + \rho_i w_i \boldsymbol{A}_i^T \boldsymbol{G}_i^T \boldsymbol{Q}_i^{-1}\left(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \boldsymbol{G}_i \boldsymbol{A}_i \hat{\boldsymbol{t}}_{m,i}\right)\right). && (29)
\end{aligned}$$

Finally, the variational distribution for the inducing users factors is:

$$\begin{aligned}
\log q(\boldsymbol{W}_m) = &\ \mathbb{E}\left[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \sum_{c=1}^{C} \mathbb{E}[\log \mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{K}_{w,mm})] + \text{const} \\
= &\ \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{w}_c; \hat{\boldsymbol{w}}_c, \boldsymbol{\Sigma}\right). && (30)
\end{aligned}$$

The SVI updates for the parameters are:

$$\begin{aligned}
\boldsymbol{\Sigma}_{c,i}^{-1} = &\ (1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}^{-1} + \rho_i \boldsymbol{K}_{w,mm}^{-1} + \rho_i w_i \boldsymbol{A}_{w,i}^T \Bigg(\sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)T} \text{diag}\left(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 + \right. \\
&\left. \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}} - 2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}\hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}}\right) \boldsymbol{Q}_i^{-1} \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \Bigg) \boldsymbol{A}_{w,i} && (31) \\
\hat{\boldsymbol{w}}_{m,c,i} = &\ \boldsymbol{\Sigma}_{c,i} \Bigg((1-\rho_i)\boldsymbol{\Sigma}_{c,i-1}\hat{\boldsymbol{w}}_{m,c,i-1} + \rho_i w_i \boldsymbol{A}_{w,i}^T \sum_{p \in P_i} \boldsymbol{H}_{.,p}^{(i)} \left(\text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}})\right. \\
&\left. - \text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{b}})\right) \boldsymbol{Q}_i^{-1} \Bigg(\boldsymbol{y}_i - \Phi(\mathbb{E}[\boldsymbol{z}_i]) + \sum_{j=1}^{U} \boldsymbol{H}_u^{(i)} (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \Bigg) \Bigg), && (32)
\end{aligned}$$

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

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

$$\boldsymbol{F}^* = \hat{\boldsymbol{t}}^* + \sum_{c=1}^{C} \hat{\boldsymbol{v}}_c^{*T} \hat{\boldsymbol{w}}_c^*, \qquad \boldsymbol{\Lambda}_u^* = \boldsymbol{C}_t^* + \sum_{c=1}^{C} \omega_{c,u}^* \boldsymbol{C}_{v,c}^* + \hat{w}_{c,u}^2 \boldsymbol{C}_{v,c}^* + \omega_{c,u}^* \hat{\boldsymbol{v}}_c \hat{\boldsymbol{v}}_c^T, \tag{33}$$

$$\hat{\boldsymbol{t}}^* = \boldsymbol{K}_{*m} \boldsymbol{K}_{mm}^{-1} \hat{\boldsymbol{t}}_m, \qquad \hat{\boldsymbol{v}}_c^* = \boldsymbol{K}_{*m} \boldsymbol{K}_{mm}^{-1} \hat{\boldsymbol{v}}_{m,c}, \qquad \hat{\boldsymbol{w}}_c^* = \boldsymbol{K}_{w,*m} \boldsymbol{K}_{w,mm}^{-1} \hat{\boldsymbol{w}}_{m,c}, \tag{34}$$

$$\boldsymbol{C}_t^* = \frac{\boldsymbol{K}_{**}}{\mathbb{E}[\sigma]} + \boldsymbol{A}_{*m}(\boldsymbol{S}_t - \boldsymbol{K}_{mm})\boldsymbol{A}_{*m}^T, \qquad \boldsymbol{C}_{v,c}^* = \frac{\boldsymbol{K}_{**}}{\mathbb{E}[s_c]} + \boldsymbol{A}_{*m}(\boldsymbol{S}_{v,c} - \boldsymbol{K}_{mm})\boldsymbol{A}_{*m}^T \tag{35}$$

$$\omega_{c,u}^* = 1 + \boldsymbol{A}_{w,um}(\boldsymbol{\Sigma}_{w,c} - \boldsymbol{K}_{w,mm})\boldsymbol{A}_{w,um}^T \tag{36}$$

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. The mean and covariance estimates 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 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, often of the following form:

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d \left(\frac{|x_d - x_d'|}{l_d}, \boldsymbol{\theta}_d \right), \tag{37}$$

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. It is possible to replace the product with a sum, allowing covariance to increase for every feature that is similar (analogous to OR), rather than only being high if all dimensions are similar (AND). The length-scale controls the smoothness of k_d across the feature space. If a feature has a large length-scale, its values 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 a median heuristic, which effectively normalizes the feature but allows extreme values to remain outliers:

$$l_{d,MH} = D\text{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1, .., N, \forall j = 1, ..., N\}). \tag{38}$$

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 37. This heuristic has been shown to

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	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 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.

work reasonably well for the task of comparing distributions (Gretton et al. 2012), but has no guarantees of optimality.

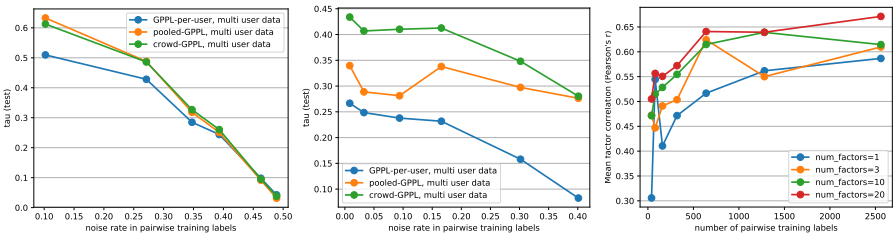
An alternative for setting l_d is Bayesian model selection using *type II maximum likelihood*, which chooses the value of l_d that maximizes the approximation to the log marginal likelihood, $\mathcal{L}$. (Equation 16 for GPPL and and Equation 22 for crowdGPPL). Optimizing kernel length-scales in this 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. Computing derivatives of $\mathcal{L}$ with respect to l_d enables the use of efficient gradient-based optimization methods such as L-BFGS-B (Zhu et al. 1997). For the single user GPPL, the required gradient with respect to the dth length-scale, l_d, is as follows:

$$\nabla_{l_d}\mathcal{L} = \frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{f}}_m}\frac{\partial\hat{\boldsymbol{f}}_m}{\partial l_d} + \frac{\partial\mathcal{L}}{\partial\boldsymbol{S}^{-1}}\frac{\partial\boldsymbol{S}^{-1}}{\partial l_d} + \frac{\partial\mathcal{L}}{\partial a}\frac{\partial a}{\partial l_d} + \frac{\partial\mathcal{L}}{\partial b}\frac{\partial b}{\partial l_d} + \frac{\partial\mathcal{L}}{\partial\boldsymbol{K}}\frac{\partial\boldsymbol{K}}{\partial l_d}. \tag{39}$$

The partial derivatives with respect to the variational parameters $\hat{\boldsymbol{f}}_m$, $\boldsymbol{S}$, a and b depend indirectly on the length-scale through the expectations in the variational q factors. However, when the variational inference algorithm has converged, $\mathcal{L}$ is at a maximum, these terms are zero, and the derivative is relatively simple to compute (see Appendix B for the full equations).

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 crowdGPPL using



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

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

SVI. In Section 6.1, we use simulated data to test the robustness of our method to noise, small training sets and unknown numbers of latent components. Section 6.2 evaluates the ability of our approach to recover personal and consensus utilities on an NLP task with high-dimensional feature vectors. Using the same dataset, we then analyze the scalability of our SVI approach. In Section 6.3, we compare our method against previous approaches for predicting the personal preferences of thousands of users on the *Sushi* datasets (Kamishima 2003). Finally, Section 6.4 evaluates whether the Bayesian approach was able to ignore redundant components when C is set larger than necessary.

6.1 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 (*GPPL*), 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 all models, we set hyperparameters $\alpha_0 = \beta_0 = 0.1$ and, for crowdGPPL, $C = 25$.

For the first test, we generate data by selecting 100 points at random from a $10x10$ grid and choosing 500 pairs of these points at random. We generate pairwise labels by drawing from crowdGPPL with 25 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 chosen points are split into 50% training and test sets. We repeat the complete experiment 25 times, including generating new data for each value of σ.

Figure 1a shows that the crowdGPPL 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 second simulation repeats the same setup as the first, but here we evaluate the methods' ability to to recover the personal preferences of simulated users. We fix $\sigma = 10$ and vary s. 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 when the generating model has a larger number of latent components, more training data is required to learn the more complex model. 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 the pair of unmatched components with the highest correlation until all true components are matched. 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. Performance is best when $C_{true} = 20$, possibly because the predictive model has $C = 25$ and hence is a closer match to the generating model. However, performance with > 500 labels remains above 0.5, for all values of C_{true}, showing the model is reasonably robust to mismatches between C and C_{true}.

6.2 Argument Convincingness

We evaluate consensus learning, personal preference learning and scalability on an NLP task, namely identifying convincing arguments. The dataset, *UKPConvArgCrowdSample*, contains arguments written by users of online debating forums and crowdsourced judgments for pairs of arguments indicating the most convincing argument. It is a subsample obtained by Simpson and Gurevych (2018) from the crowdsourced data provided by Habernal and Gurevych (2016). Each argument is are 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. 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 somewhat on their prior beliefs and understanding of the subject discussed and the language used, and therefore crowdGPPL 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.

Table 2 shows performance for GPPL and crowdGPPL with the median heuristic (*medi.*) and with gradient-based optimization (*opt.*). The hyperpa-

	Consensus			Personal		
Method	Acc	CEE	Kend.	Acc	CEE	Kend.
GPPL medi.	.77	.50	.40	.72	.55	.32
GPPL opt.	.76	.51	.47	.70	.57	.30
crowdGPPL medi.	.78	.48	.51	.70	.59	.30
crowdGPPL opt.	?	?	?	?	?	?

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.

rameters were set to $\alpha_0 = 2$, $\beta_0 = 200$ by comparing training set accuracy against $\alpha_0 = 2, \beta_0 = 20000$ and $\alpha_0 = 2, \beta_0 = 2$. We set $C = 50$ and did not optimize this value. CrowdGPPL outperforms GPPL at predicting the consensus pairwise labels (all significant with $p < 0.05$, Wilcoxon signed-rank test), shown by classification accuracy (*acc*) and cross entropy error (*CEE*), and the consensus ranking ($p < 0.05$), shown by Kendall's tau rank correlation (*Kend*). For the personal preference predictions, crowdGPPL outperforms GPPL at ranking ($p < 0.05$) but pairwise label prediction performance is comparable. This may be because the pairwise comparisons in the test folds are noisy and contain numerous contradictions (Habernal and Gurevych 2016), which are not present in the gold standard rankings, and hence are difficult to predict. Length-scale optimization improves performance over the median heuristic, although the difference is small and required approximately 5 times longer to run.

We assess our proposed SVI inference method by evaluating GPPL and crowdGPPL with different numbers of inducing points, M. Figure 2 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 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.

Figures 3a and 3b show runtimes as a function of the training set size, N_{tr}, and the number of pairwise training labels, P, respectively. The methods labeled "no SVI" show runtimes for GPPL and crowdGPPL with variational inference but no stochastic updates or inducing points. When using SVI, runtimes increase very little with N_{tr} or P.

6.3 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 crowdGPPL, as well as their runtimes, against previous approaches, and investigate the

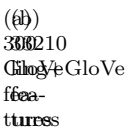


Fig. 2 Effect of varying M on accuracy and runtime (training+prediction) for GPPL and crowdGPPL on UKPConvArgCrowdSample. Means over 32 runs.

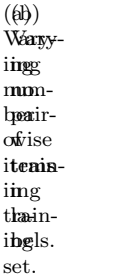


Fig. 3 Runtimes for training+prediction on UKPConvArgCrowdSample with varying subsample size. 300 GloVe features. Means over 32 runs. Note the logarithmic x-axis for (b).

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. For Sushi-A, we select a subset of 1000 workers at random, then split the data into training and test sets by 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. Beside GPPL, crowdGPPL and GPPL-per-user, we introduce four further baselines: *crowdGPPL*$\backslash \boldsymbol{u}$, which ignores the user features; *crowdGPPL*$\backslash \boldsymbol{u} \setminus \boldsymbol{x}$, which ignores both user and item features and so does not use GPs at all; *crowdGPPL*$\backslash \boldsymbol{u} \setminus \boldsymbol{t}$, 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. Standard crowdGPPL is labelled '*full*'. For these methods, the user covariance matrix, $\boldsymbol{K}_w$, in the crowdGPPL model is replaced by the identity matrix, and for crowdGPPL$\backslash \boldsymbol{u} \setminus \boldsymbol{x}$, 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 and $\alpha_0 = 1, \beta_0 = 100$ using a grid search over values $10^{\{-1,\ldots,3\}}$ on withheld user data from the Sushi-A dataset. The complete process of subsampling, training and testing, was repeated 25 times for each dataset.

The results in Table 3 illustrate the benefit of crowd models over single-user GPPL. The runtimes show that including feature data for items and users leads to faster convergence than crowdGPPL$\backslash \boldsymbol{u}$ and crowd$\backslash \boldsymbol{u} \backslash \boldsymbol{x}$. However, the runtimes for crowdGPPL are higher than those of GPPL. When using inducing points, the user features decrease the performance of crowdGPPL: the methods

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
crowdGPPL												
full	.67	.87	.39	77	.79	.51	.66	150	.69	2.23	.45	9163
full, opt.					.80	.48	.68	3718	-	-	-	-
\induc	.70	**.57**	.46	166	.84	**.33**	.79	315	-	-	-	-
$\backslash u$	.70	.58	.46	175	.84	**.33**	**.80**	336	**.76**	**.47**	**.60**	21052
$\backslash u$, opt.					**.85**	**.33**	**.80**	5016	-	-	-	-
$\backslash u \backslash x$	**.71**	**.57**	**.49**	5	**.85**	**.33**	**.80**	329	**.76**	.48	**.60**	15478
$\backslash u \backslash t$	.68	.60	.43	291	.84	**.33**	.79	540	.75	.49	.59	30484
singlePL												
GPPL	.65	.62	.31	**2**	.66	.62	.32	**2**	.65	.62	.31	**51**
per-user	.67	.64	.42	97	.83	.40	.79	67				

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.

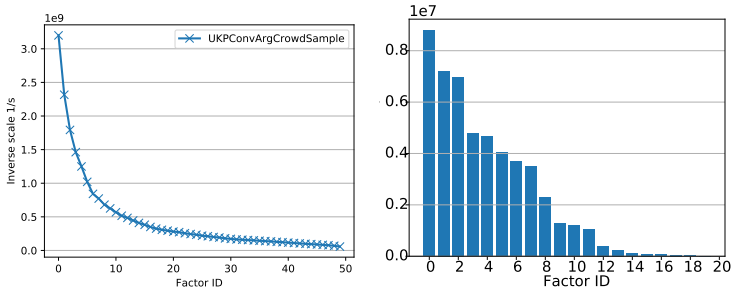
crowdGPPL$\backslash$*induc* and *crowdGPPL*$\backslash \boldsymbol{u}$ both outperform the full crowdGPPL. When using inducing points, there must be a strong relationship between neighbouring points, which is likely not to be the case given the features in this dataset. Again, we see small improvements in performance after length-scale optimization, but this takes approximately 22 times as long to train. 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=0.39 on Sushi-B). GPPL-per-user performs well on Sushi-A, but does not perform as well as other methods on Sushi-B.

6.4 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 4 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 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 even when a larger number of latent components was available.

7 Conclusions

We proposed a Bayesian preference learning approach for modeling both the individual utilities of members of a crowd and forming a consensus from un-



(a) UKPConvArgCrowdSample (b) Sushi-A

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

reliable annotations, such as those provided by a crowd or through implicit user feedback. The model learns the latent utilities of items from pairwise comparisons using a combination of Gaussian processes and Bayesian matrix factorization. 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 far better than previous Gaussian process preference learning methods without harming performance on individual utility prediction and significantly improving performance on consensus learning. Future work will evaluate the benefit of learning inducing point locations from data, and investigate the possibility of integrating deep generative models for learning feature representations from input data.

Acknowledgments

References

Abbasnejad E, Sanner S, Bonilla EV, Poupart P, et al. (2013) Learning community-based preferences via dirichlet process mixtures of gaussian processes. In: Twenty-Third International Joint Conference on Artificial Intelligence, pp 1213–1219

Ahn S, Korattikara A, Liu N, Rajan S, Welling M (2015) Large-scale distributed bayesian matrix factorization using stochastic gradient mcmc. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 9–18

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

Banerji M, Lahav O, Lintott CJ, Abdalla FB, Schawinski K, Bamford SP, Andreescu D, Murray P, Raddick MJ, Slosar A, et al. (2010) Galaxy zoo: reproducing galaxy morphologies via machine learning. Monthly Notices of the Royal Astronomical Society 406(1):342–353

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

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

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

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

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 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

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: IJCAI

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

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

A Variational Inference

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

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \alpha_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)ds}{\int\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}'; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)dsd\boldsymbol{f}'}. \tag{40}$$

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

$$\mathcal{L}_1 = \sum_{i=1}^{L} \mathbb{E}_q \left[\log p\left(v_i \succ u_i | f(v_i), f(u_i)\right)\right] + \mathbb{E}_q\left[\log \frac{p\left(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K}/s\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_q\left[\log \frac{p\left(s|\alpha_0, \beta_0\right)}{q\left(s\right)}\right] \tag{41}$$

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

$$\begin{aligned}\mathcal{L}_1 = \mathbb{E}_q &\left[\sum_{i=1}^{L} [v_i \succ u_i] \log \Phi(z_i) + [v_i \prec u_i]\left(1 - \log \Phi(z_i)\right)\right] \\ &+ \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) + \mathbb{E}_q\left[\log \mathcal{G}\left(s; \alpha_0, \beta_0\right) - \log \mathcal{G}\left(s; \alpha, \beta\right)\right]\end{aligned} \tag{42}$$

We now replace the likelihood with a Gaussian approximation:

$$\begin{aligned}\mathcal{L}_1 \approx \mathcal{L}_2 &= \mathbb{E}_q\left[\mathcal{N}(\boldsymbol{y}|\Phi(\boldsymbol{z}), \boldsymbol{Q})\right] + \log \mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ &+ \mathbb{E}_q\left[\log \mathcal{G}\left(s; \alpha_0, \beta_0\right) - \log \mathcal{G}\left(s; \alpha, \beta\right)\right] \\ &= -\frac{1}{2}\Big\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}/s| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &+ \mathbb{E}_q\left[(\boldsymbol{y} - \Phi(\boldsymbol{z}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\boldsymbol{z}))\right]\Big\} - \Gamma(\alpha_0) + \alpha_0(\log \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\log s] \\ &+ \Gamma(\alpha) + (\beta - \beta_0)\hat{s} - \alpha \log \beta,\end{aligned} \tag{43}$$

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:

$$\begin{aligned}\mathcal{L}_2 \approx \mathcal{L}_3 = -\frac{1}{2}&\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}/\hat{s}| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &+ (\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))\Big\} - \Gamma(\alpha_0) + \alpha_0(\log \beta_0) + (\alpha_0 - \alpha)\mathbb{E}[\log s] \\ &+ \Gamma(\alpha) + (\beta - \beta_0)\hat{s} - \alpha \log \beta.\end{aligned} \tag{44}$$

Now, we can introduce the sparse approximation to obtain the bound in Equation 16:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{f}_m,s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K}, \alpha_0, \beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)] \\ &= \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|f_{a_p}, f_{b_p})] - \frac{1}{2}\Big\{ \log|\boldsymbol{K}_{mm}| - \mathbb{E}[\log s] - \log|\boldsymbol{S}| - M \\ &\quad + \hat{\boldsymbol{f}}_m^T \mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\Big\} + \log\Gamma(\alpha) - \log\Gamma(\alpha_0) + \alpha_0(\log\beta_0) \\ &\quad + (\alpha_0 - \alpha)\mathbb{E}[\log s] + (\beta - \beta_0)\mathbb{E}[s] - \alpha\log\beta, \end{aligned} \tag{45}$$

where the terms relating to $\mathbb{E}[p(\boldsymbol{f}|\boldsymbol{f}_m) - q(\boldsymbol{f})]$ cancel. Without stochastic sampling, the variational factor $\log q(\boldsymbol{f}_m)$ is given by:

$$\begin{aligned}\log q(\boldsymbol{f}_m) &= \log\mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}\right)] + \log\mathcal{N}\left(\boldsymbol{f}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}\left[s\right]\right) + \text{const}, \\ &= \log\mathcal{N}(\boldsymbol{f}_m; \hat{\boldsymbol{f}}_m, \boldsymbol{S}), \end{aligned} \tag{46}$$

$$\boldsymbol{S}^{-1} = \boldsymbol{K}_{mm}^{-1}/\mathbb{E}[s] + \boldsymbol{A}^T\boldsymbol{G}^T\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{A}, \tag{47}$$

$$\hat{\boldsymbol{f}}_m = \boldsymbol{S}\boldsymbol{A}^T\boldsymbol{G}^T\boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \boldsymbol{G}\mathbb{E}[\boldsymbol{f}]). \tag{48}$$

For the crowdGPPL model, the variational lower bound is:

$$\begin{aligned}\mathcal{L}_{cr} &= \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|\boldsymbol{v}_{a_p}^T\boldsymbol{w}_{a_p} + t_{a_p}, \boldsymbol{v}_{b_p}^T\boldsymbol{w}_{b_p} + t_{b_p})] - \frac{1}{2}\Bigg\{ \sum_{c=1}^{C} \Big\{ -M_n - M_u \\ &\quad + \log|\boldsymbol{K}_{v,mm}| + \log|\boldsymbol{K}_{w,mm}| - \log|\boldsymbol{S}_{v,c}| - \mathbb{E}[\log s_c] + \hat{\boldsymbol{v}}_{m,c}^T\mathbb{E}[s_c]\boldsymbol{K}_{v,mm}^{-1}\hat{\boldsymbol{v}}_{m,c} \\ &\quad + \mathrm{tr}(\mathbb{E}[s_c]\boldsymbol{K}_{v,mm}^{-1}\boldsymbol{S}_{v,c}) - \log|\boldsymbol{\Sigma}_c| + \hat{\boldsymbol{w}}_{m,c}^T\boldsymbol{K}_{w,mm}^{-1}\hat{\boldsymbol{w}}_{m,c} + \mathrm{tr}(\boldsymbol{K}_{w,mm}^{-1}\boldsymbol{\Sigma}_c)\Big\} \\ &\quad - M_n + \log|\boldsymbol{K}_{t,mm}| - \log|\boldsymbol{S}_t| - \mathbb{E}[\log\sigma] + \hat{\boldsymbol{t}}^T\mathbb{E}[\sigma]\boldsymbol{K}_{t,mm}^{-1}\hat{\boldsymbol{t}} \\ &\quad + \mathrm{tr}(\mathbb{E}[\sigma]\boldsymbol{K}_{t,mm}^{-1}\boldsymbol{S}_t)\Bigg\} - (C+1)(\log\Gamma(\alpha_0) + \alpha_0(\log\beta_0)) \\ &\quad + \sum_{c=1}^{C}\Big\{ \log\Gamma(\alpha_c) + (\alpha_0 - \alpha_c)\mathbb{E}[\log s_c] + (\beta_c - \beta_0)\mathbb{E}[s_c] - \alpha_c\log\beta_c \Big\} \\ &\quad + \log\Gamma(\alpha_\sigma) + (\alpha_0 - \alpha_\sigma)\mathbb{E}[\log\sigma] + (\beta_\sigma - \beta_0)\mathbb{E}[s_c] - \alpha_\sigma\log\beta_\sigma, \end{aligned} \tag{49}$$

B Converged Lower Bound Derivatives

When $\mathcal{L}$ has converged to a maximum, $\nabla_{l_d}\mathcal{L}$ simplifies to:

$$\nabla_{l_d}\mathcal{L} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\mathbb{E}[s](\hat{\boldsymbol{f}}_m\hat{\boldsymbol{f}}_m^T + \boldsymbol{S}^T)\boldsymbol{K}_{mm}^{-1} - \boldsymbol{I}\right)\frac{\partial\boldsymbol{K}_{mm}}{\partial l_d}\boldsymbol{K}_{mm}^{-1}\right). \tag{50}$$

For crowdGPPL, assuming that $\boldsymbol{V}$ and $\boldsymbol{t}$ have the same kernel function, the gradient for the dth item feature is given by:

$$\begin{aligned}\nabla_{l_{v,d}}\mathcal{L}_{cr} \longrightarrow \frac{1}{2}\mathrm{tr}&\left(\left(\sum_{c=1}^{C}\mathbb{E}[s_c]\left\{\hat{\boldsymbol{v}}_{m,c}\hat{\boldsymbol{v}}_{m,c}^T + \boldsymbol{S}_{v,c}^T\right\}\boldsymbol{K}_{mm,v}^{-1} - C\boldsymbol{I}\right)\frac{\partial\boldsymbol{K}_{mm,v}}{\partial l_{w,d}}\right. \\ &\left.\boldsymbol{K}_{mm,v}^{-1}\right) + \frac{1}{2}\mathrm{tr}\left(\left(\mathbb{E}[\sigma](\hat{\boldsymbol{t}}_m\hat{\boldsymbol{t}}_m^T + \boldsymbol{S}_t^T)\boldsymbol{K}_{mm,t}^{-1} - \boldsymbol{I}\right)\frac{\partial\boldsymbol{K}_{mm,t}}{\partial l_{w,d}}\boldsymbol{K}_{mm,t}^{-1}\right). \end{aligned} \tag{51}$$

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

$$\nabla_{l_{w,d}}\mathcal{L}_{cr} \longrightarrow \frac{1}{2}\text{tr}\left(\left(\sum_{c=1}^{C}\left\{\hat{\boldsymbol{w}}_{m,c}\hat{\boldsymbol{w}}_{m,c}^{T} + \boldsymbol{\Sigma}_{c}^{T}\right\}\boldsymbol{K}_{mm,w}^{-1} - C\boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,w}}{\partial l_{w,d}}\boldsymbol{K}_{mm,w}^{-1}\right). \tag{52}$$

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:

$$k_d\left(\frac{|x_d - x'_d|}{l_d}\right) = \left(1 + \frac{\sqrt{3}|x_d - x'_d|}{l_d}\right)\exp\left(-\frac{\sqrt{3}|x_d - x'_d|}{l_d}\right). \tag{53}$$

Assuming that the kernel functions for each feature, k_d, are combined using a product, as in Equation 37, the partial derivative $\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}$ is a matrix, where each entry, i, j, is defined by:

$$\frac{\partial K_{mm,ij}}{\partial l_d} = \prod_{d'=1,d'\neq d}^{D} k_{d'}\left(\frac{|x_{d'} - x'_{d'}|}{l_{d'}}\right)\frac{3(\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d})^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|}{l_d}\right), \tag{54}$$

where we assume the use of Equation37 to combine kernel functions over features using a product.

To make use of Equations 50 to 54, 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 $\mathcal{L}$.