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, instances 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 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 instances and or predicting the instance with the highest value in a pair. For many preference learning tasks, annotators have

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

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 instances (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 instances 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 instances, since it avoids cases where instances have the same values (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.

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 unseen instance using the observed preferences of similar users for that instance (Resnick and Varian 1997). A typical approach is to represent observed ratings in a user-instance matrix, then apply *matrix factorization* (Koren et al. 2009) to decompose a user-instance 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 sources, users or annotators, 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 instances (Chen et al. 2013). For classification tasks, Simpson et al. (2017) show that modeling instance 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 instances 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, instances 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 annotator's choices as a random function of the latent utility of the instances (Thurstone 1927). 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 instance 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 instances from a set. They apply a simple counting technique to infer a ranking over the instances, which requires each instance to have a sufficient number of comparisons.

A popular method for predicting pairwise labels of new instances 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 instances 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 with to predict the preferences of individual annotators or users. This task has previously been addressed by Yi et al. (2013) and Kim et al. (2014). Both models identify multiple latent rankings and infer the preference of each annotator toward those rankings, but neither can generalize to new test instances. Gaussian processes can, however, exploit instance features. Guo et al. (2010) propose an 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 factorization to perform collaborative filtering. However, this approach does not model the relationship between input features and the latent 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 instances 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, annotators or data sources, 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 instances 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 instances, 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 instances. This means that the inducing points M have to stand in for P, which is typically larger than the number of instances, N. A GP over the pairs also makes it difficult to extract posteriors for latent function values for individual instances, 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 instances 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:

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

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:

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

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)\}$ are the utilities of the N items referred to by $\boldsymbol{y}$.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:

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

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

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

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 instances, 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* instances with inputs $\boldsymbol{x}_m$, utilities $\boldsymbol{f}_m$, covariance $\boldsymbol{K}_{mm}$, and covariance between the observed and inducing instances, $\boldsymbol{K}_{nm}$. For clarity, we omit θ from this point on, and provide further detailed equations in Appendix B. The posterior over the inducing and training instances 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 by following the natural gradient (Hensman et al. 2015):

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

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

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).

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.

For the inverse scale, $q(s)$ is updated using Equation 15, then its expect value is computed as $\mathbb{E}[s] = \frac{2\alpha_0 + M}{2\beta}$.

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. The variational factors for the inducing points can be obtained by deriving expectations as follows, beginning with the latent item factors:

$$\log q(\boldsymbol{V}_m) = \mathbb{E}\left[\log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)\right] + \sum_{c=1}^{C} \log \mathcal{N}\left(\boldsymbol{v}_{m,c}; \boldsymbol{0}, \frac{\boldsymbol{K}_{v,mm}}{\mathbb{E}[s_c]}\right) + \text{const}$$
$$= \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{v}_{m,c}; \hat{\boldsymbol{v}}_{m,c}, \boldsymbol{S}_{v,c}). \tag{22}$$

where the precision is given by:

$$\boldsymbol{S}_{v,c}^{-1} = \boldsymbol{K}_{v,mm}^{-1}\mathbb{E}[s_c] + \boldsymbol{A}_v^T\boldsymbol{G}^T\text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}}^2 + \boldsymbol{\Sigma}_{c,\boldsymbol{u},\boldsymbol{u}})\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{A}_v, \tag{23}$$

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 31 and 30), 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:

$$\hat{\boldsymbol{v}}_{m,c} = \boldsymbol{S}_{v,c}\boldsymbol{A}_v^T\boldsymbol{G}^T\text{diag}(\hat{\boldsymbol{w}}_{c,\boldsymbol{u}})\boldsymbol{Q}^{-1}\left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \sum_{j=1}^{U}\boldsymbol{H}_j(\hat{\boldsymbol{v}}_c^T\hat{\boldsymbol{w}}_{c,j})\right), \tag{24}$$

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:

$$H_{j,p,i} = \Phi(\mathbb{E}[z_p])(1 - \Phi(\mathbb{E}[z_p]))(2y_p - 1)([i = a_p] - [i = b_p])[j = u_p]. \tag{25}$$

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:

$$\begin{aligned}
\log q(\boldsymbol{t}_m) &= \mathbb{E}_{q(\boldsymbol{V})q(\boldsymbol{W})}[\log\mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \mathbb{E}[\log\mathcal{N}(\boldsymbol{t}_m; \boldsymbol{0}, \boldsymbol{K}_{t,mm}/s)] + \text{const} \\
&= \log\mathcal{N}\left(\boldsymbol{t}; \hat{\boldsymbol{t}}, \boldsymbol{S}_t\right) && (26) \\
\boldsymbol{S}_t^{-1} &= \boldsymbol{K}_{t,mm}^{-1}/\mathbb{E}[\sigma] + \boldsymbol{A}_t^T\boldsymbol{G}^T\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{A}_t && (27) \\
\hat{\boldsymbol{t}}_m &= \boldsymbol{S}_t\boldsymbol{A}_t^T\boldsymbol{G}^T\boldsymbol{Q}^{-1}\left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \boldsymbol{G}(\hat{\boldsymbol{t}})\right), && (28)
\end{aligned}$$

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:

$$\begin{aligned}
\log q(\boldsymbol{W}_m) &= \mathbb{E}_{q(\boldsymbol{V})q(\boldsymbol{t})}[\log\mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), Q\right)] + \sum_{c=1}^{C}\mathbb{E}[\log\mathcal{N}(\boldsymbol{w}_c; \boldsymbol{0}, \boldsymbol{K}_{w,mm})] + \text{const} \\
&= \sum_{c=1}^{C}\log\mathcal{N}\left(\boldsymbol{w}_c; \hat{\boldsymbol{w}}_c, \boldsymbol{\Sigma}\right), && (29)
\end{aligned}$$

where the variational parameters are:

$$\boldsymbol{\Sigma}_c^{-1} = \boldsymbol{K}_{w,mm}^{-1} + \boldsymbol{A}_w^T \left(\sum_{p=1}^{P} \boldsymbol{H}_{.,p}^T \text{diag}\left(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}^2 + \boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{a}} + \hat{\boldsymbol{v}}_{c,\boldsymbol{b}}^2 + \boldsymbol{S}_{c,\boldsymbol{b},\boldsymbol{b}} - 2\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}\hat{\boldsymbol{v}}_{c,\boldsymbol{b}} - 2\boldsymbol{S}_{c,\boldsymbol{a},\boldsymbol{b}} \right) \boldsymbol{Q}^{-1} \sum_{p=1}^{P} \boldsymbol{H}_{.,p} \right) \boldsymbol{A}_w \tag{30}$$

$$\hat{\boldsymbol{w}}_{m,c} = \boldsymbol{\Sigma}_c \boldsymbol{A}_w^T \sum_{p=1}^{P} \boldsymbol{H}_{.,p} \left(\text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{a}}) - \text{diag}(\hat{\boldsymbol{v}}_{c,\boldsymbol{b}}) \right) \boldsymbol{Q}^{-1} \left(\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \sum_{j=1}^{U} \boldsymbol{H}_u (\hat{\boldsymbol{v}}_c^T \hat{\boldsymbol{w}}_{c,j}) \right), \tag{31}$$

where the subscripts $\cdot_{\boldsymbol{a}} = \{\cdot_{a_p} \forall p \in 1, ..., P\}$ and $\cdot_{\boldsymbol{b}} = \{\cdot_{b_p} \forall p \in 1, ..., P\}$ 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 17 and 18. 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):

$$\log q(\boldsymbol{V}) = \sum_{c=1}^{C} \log \mathcal{N} \left(\boldsymbol{v}_c; \boldsymbol{A}_v \hat{\boldsymbol{v}}_{m,c}, \frac{\boldsymbol{K}_v}{\mathbb{E}[s_c]} + \boldsymbol{A}_v (\boldsymbol{S}_{m,c} - \frac{\boldsymbol{K}_{v,mm}}{\mathbb{E}[s_c]}) \boldsymbol{A}_v \right) \tag{32}$$

$$\log q(\boldsymbol{t}) = \log \mathcal{N} \left(\boldsymbol{t}; \boldsymbol{A}_t \hat{\boldsymbol{t}}_m, \frac{\boldsymbol{K}_t}{\mathbb{E}[\sigma]} + \boldsymbol{A}_t (\boldsymbol{S}_t - \frac{\boldsymbol{K}_{t,mm}}{\mathbb{E}[\sigma]}) \boldsymbol{A}_t \right) \tag{33}$$

$$\log q(\boldsymbol{W}) = \sum_{c=1}^{C} \log \mathcal{N} \left(\boldsymbol{w}_c; \boldsymbol{A}_w \hat{\boldsymbol{w}}_{m,c}, \boldsymbol{K}_w + \boldsymbol{A}_w (\boldsymbol{\Sigma} - \boldsymbol{K}_{w,mm}) \boldsymbol{A}_w \right). \tag{34}$$

The expectations for the inverse scales, $s_1, ..., s_c$ and σ, can be computed using the formulas in Equations 54 and 55 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 32, 33 and 34 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:

$$\begin{aligned}\mathcal{L}_{crowd} = \sum_{p=1}^{P} \mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|\boldsymbol{v}_{a_p}^T\boldsymbol{w}_{a_p} + t_{a_p}, \boldsymbol{v}_{b_p}^T\boldsymbol{w}_{b_p} + t_{b_p})] - \frac{1}{2}\Bigg\{ \sum_{c=1}^{C} \Bigg\{ -M_n - M_u \\ + \log|\boldsymbol{K}_{v,mm}| + \log|\boldsymbol{K}_{w,mm}| - \log|\boldsymbol{S}_{v,c}| - \mathbb{E}[\log s_c] + \hat{\boldsymbol{v}}_{m,c}^T \mathbb{E}[s_c] \boldsymbol{K}_{v,mm}^{-1} \hat{\boldsymbol{v}}_{m,c} \\ + \mathrm{tr}(\mathbb{E}[s_c]\boldsymbol{K}_{v,mm}^{-1}\boldsymbol{S}_{v,c}) - \log|\boldsymbol{\Sigma}_c| + \hat{\boldsymbol{w}}_{m,c}^T \boldsymbol{K}_{w,mm}^{-1} \hat{\boldsymbol{w}}_{m,c} + \mathrm{tr}(\boldsymbol{K}_{w,mm}^{-1}\boldsymbol{\Sigma}_c) \Bigg\} \\ - M_n + \log|\boldsymbol{K}_{t,mm}| - \log|\boldsymbol{S}_t| - \mathbb{E}[\log\sigma] + \hat{\boldsymbol{t}}^T \mathbb{E}[\sigma] \boldsymbol{K}_{t,mm}^{-1} \hat{\boldsymbol{t}} \\ + \mathrm{tr}(\mathbb{E}[\sigma]\boldsymbol{K}_{t,mm}^{-1}\boldsymbol{S}_t) \Bigg\} - (C+1)(\log\Gamma(\alpha_0) + \alpha_0(\log\beta_0)) \\ + \sum_{c=1}^{C} \Bigg\{ \log\Gamma(\alpha_c) + (\alpha_0 - \alpha_c)\mathbb{E}[\log s_c] + (\beta_c - \beta_0)\mathbb{E}[s_c] - \alpha_c \log\beta_c \Bigg\} \\ + \log\Gamma(\alpha_\sigma) + (\alpha_0 - \alpha_\sigma)\mathbb{E}[\log\sigma] + (\beta_\sigma - \beta_0)\mathbb{E}[s_c] - \alpha_\sigma\log\beta_\sigma, \end{aligned} \tag{35}$$

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:

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d\left(\frac{|x_d - x_d'|}{l_d}, \boldsymbol{\theta}_d\right) \tag{36}$$

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:

$$l_{d,MH} = D\text{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1, .., N, \forall j = 1, ..., N\}). \tag{37}$$

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 36. 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 16 for a single user, and Equation 35 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:

$$\nabla_{l_d}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{f}}_m}\frac{\partial \hat{\boldsymbol{f}}_m}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}^{-1}}\frac{\partial \boldsymbol{S}^{-1}}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial b}\frac{\partial b}{\partial l_d} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{K}}\frac{\partial \boldsymbol{K}}{\partial l_d}. \tag{38}$$

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:

$$\nabla_{l_d}\mathcal{L} \longrightarrow \frac{1}{2}\text{tr}\left(\left(\mathbb{E}[s](\hat{\boldsymbol{f}}_m\hat{\boldsymbol{f}}_m^T + \boldsymbol{S}^T)\boldsymbol{K}_{mm}^{-1} - \boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\boldsymbol{K}_{mm}^{-1}\right). \tag{39}$$

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:

$$\nabla_{l_{v,d}}\mathcal{L}_{crowd} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\sum_{c=1}^{C}\mathbb{E}[s_c]\left\{\hat{\boldsymbol{v}}_{m,c}\hat{\boldsymbol{v}}_{m,c}^T + \boldsymbol{S}_{v,c}^T\right\}\boldsymbol{K}_{mm,v}^{-1} - C\boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,v}}{\partial l_{w,d}}\boldsymbol{K}_{mm,v}^{-1}\right)$$
$$+\frac{1}{2}\mathrm{tr}\left(\left(\mathbb{E}[\sigma](\hat{\boldsymbol{t}}_m\hat{\boldsymbol{t}}_m^T + \boldsymbol{S}_t^T)\boldsymbol{K}_{mm,t}^{-1} - \boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,t}}{\partial l_{w,d}}\boldsymbol{K}_{mm,t}^{-1}\right). \quad (40)$$

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

$$\nabla_{l_{w,d}}\mathcal{L}_{crowd} \longrightarrow \frac{1}{2}\mathrm{tr}\left(\left(\sum_{c=1}^{C}\left\{\hat{\boldsymbol{w}}_{m,c}\hat{\boldsymbol{w}}_{m,c}^T + \boldsymbol{\Sigma}_c^T\right\}\boldsymbol{K}_{mm,w}^{-1} - C\boldsymbol{I}\right)\frac{\partial \boldsymbol{K}_{mm,w}}{\partial l_{w,d}}\boldsymbol{K}_{mm,w}^{-1}\right). \quad (41)$$

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:

$$k_d\left(\frac{|x_d - x'_d|}{l_d}\right) = \left(1 + \frac{\sqrt{3}|x_d - x'_d|}{l_d}\right)\exp\left(-\frac{\sqrt{3}|x_d - x'_d|}{l_d}\right). \quad (42)$$

Assuming that the kernel functions for each feature, k_d, are combined using a product, as in Equation 36, the partial derivative $\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}$ is a matrix, where each entry, i, j, is defined by:

$$\frac{\partial K_{mm,ij}}{\partial l_d} = \prod_{d'=1,d'\neq d}^{D} k_{d'}\left(\frac{|x_{d'} - x'_{d'}|}{l_{d'}}\right)\frac{3(\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d})^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|}{l_d}\right), \quad (43)$$

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

To make use of Equations 39 to 59, 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 $\mathcal{L}$.

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.

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\$\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 2; 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.

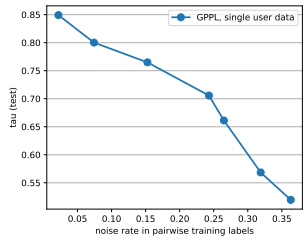
The third test repeats a similar setup to the second, but here we evaluate the methods' ability to to recover the personal preferences of individual simulated users. We fix $\sigma = 10$ and vary the precision of the item latent factors, s. Results for predicting personal preferences in the presence 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.

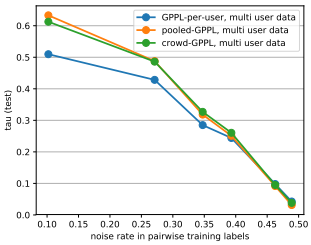
The results in Figure 1d show that for all values of C_{true}, increasing the number of training labels beyond 1000 does not increase the correlations between the best-matched latent user factors, and in fact the correlations decrease. Since the crowd-GPPL model is able to learn $C = 25$ factors, it 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

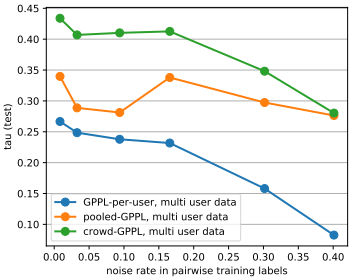
We evaluate our preference learning approaches on an NLP dataset containing 32,310 different features for arguments written by users of online debating forums Habernal and Gurevych (2016). The task is to quantify how convincing each argument is by learning a model from pairwise preference labels obtained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



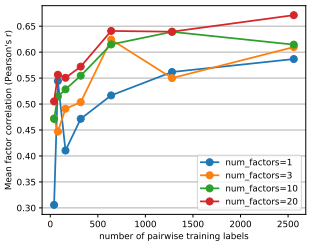
(a) Inferring preferences for a single user



(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.

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 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, $N_t r$. With a fixed M, runtimes increase very little with $N_t r$, 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

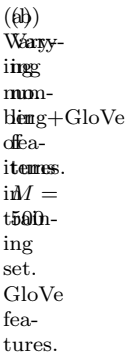


Fig. 2 Runtimes for training+prediction on UKPConvArgCrowdSample with varying subsample size. Means over 32 runs. Note the logarithmic x-axis for (b).

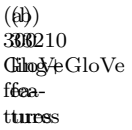


Fig. 3 Effect of varying M on accuracy and runtime (training+prediction) for GPPL and crowd-GPPL on UKPConvArgCrowdSample. Means over 32 runs.

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.

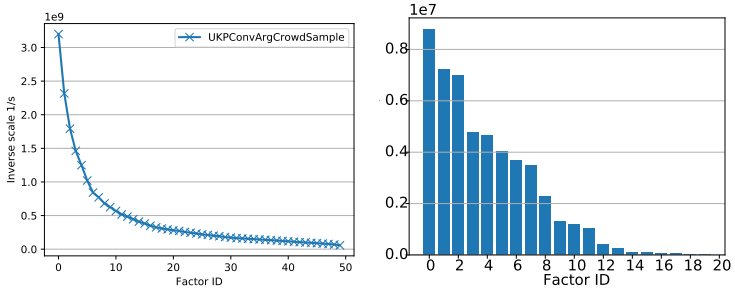
The pooled-GPPL method was previously tested on *UKPConvArgCrowdSample* in (Simpson and Gurevych 2018), and shown to outperform SVM, Bi-LSTM and Gaussian process classifier methods. In this paper, we compare pooled-GPPL with crowd-GPPL and also test each method's ability to predict the raw crowdsourced 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....

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

	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.	.70	.69	.51	.70	.62	.19
Crowd-GPPL opt.	?	.69	?	?	?	?

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) UKPConvArgCrowdSample (b) Sushi-A

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

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 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

Method	Sushi-A-small				Sushi-A				Sushi-B			
	Acc	CEE	τ	Runtime	Acc	CEE	τ	Runtime	Acc	CEE	τ	Runtime
crowdPL												
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												
pooled	.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.

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

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

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

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
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 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
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

A Posterior Inference

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

$$p(\boldsymbol{f}|\boldsymbol{y}, k_\theta, \alpha_0, \beta_0) = \frac{\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)ds}{\int\int \prod_{p=1}^{P} \Phi(z_p)\mathcal{N}(\boldsymbol{f}'; \boldsymbol{0}, \boldsymbol{K}_\theta/s)\mathcal{G}(s; \alpha_0, \beta_0)dsd\boldsymbol{f}'}. \tag{44}$$

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

$$\prod_{p=1}^{P} \Phi(z_p) \approx \mathcal{N}(\boldsymbol{y}; \Phi(\boldsymbol{z}), \boldsymbol{Q}), \tag{45}$$

where $\boldsymbol{z} = \{z_1, ..., z_P\}$ and $\boldsymbol{Q}$ is a diagonal noise covariance matrix.

B Variational Inference

$$\begin{aligned}
\log q(\boldsymbol{f}_m) &= \log \mathcal{N}\left(\boldsymbol{y}; \tilde{\Phi}(\boldsymbol{z}), \boldsymbol{Q}\right)] + \log \mathcal{N}\left(\boldsymbol{f}_m; \boldsymbol{0}, \boldsymbol{K}_{mm}/\mathbb{E}\left[s\right]\right) + \text{const}, \\
&= \log \mathcal{N}(\boldsymbol{f}_m; \hat{\boldsymbol{f}}_m, \boldsymbol{S}), && (46) \\
\boldsymbol{S}^{-1} &= \boldsymbol{K}_{mm}^{-1}/\mathbb{E}[s] + \boldsymbol{A}^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} \boldsymbol{G} \boldsymbol{A}, && (47) \\
\hat{\boldsymbol{f}}_m &= \boldsymbol{S} \boldsymbol{A}^T \boldsymbol{G}^T \boldsymbol{Q}^{-1} (\boldsymbol{y} - \Phi(\mathbb{E}[\boldsymbol{z}]) + \boldsymbol{G}\mathbb{E}[\boldsymbol{f}]), && (48)
\end{aligned}$$

We derive the variational lower bound as follows:

$$\begin{aligned}
\mathcal{L}(q) = &\sum_{i=1}^{L} \mathbb{E}_q \left[\log p\left(v_i \succ u_i | f(v_i), f(u_i)\right)\right] \\
&+ \mathbb{E}_q \left[\log \frac{p\left(\boldsymbol{f} | \boldsymbol{\mu}, \boldsymbol{K}/s\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_q \left[\log \frac{p\left(s | a_0, b_0\right)}{q\left(s\right)}\right] \qquad (49)
\end{aligned}$$

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

$$\begin{aligned}
\mathcal{L}(q) = &\mathbb{E}_q \Bigg[\sum_{i=1}^{L} [v_i \succ u_i] \log \Phi(z_i) \\
&+ [v_i \prec u_i] \left(1 - \log \Phi(z_i)\right) \Bigg] \\
&+ \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\
&+ \mathbb{E}_q \left[\log \mathcal{G}\left(s; a_0, b_0\right) - \log \mathcal{G}\left(s; a, b\right)\right] \qquad (50)
\end{aligned}$$

We now replace the likelihood with a Gaussian approximation:

$$\begin{aligned}
\mathcal{L}(q) &\approx \mathbb{E}_q \left[\mathcal{N}(\boldsymbol{y} | \Phi(\boldsymbol{z}), \boldsymbol{Q})\right] \\
&\quad + \log \mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\
&\quad + \mathbb{E}_q \left[\log \mathcal{G}\left(s; a_0, b_0\right) - \log \mathcal{G}\left(s; a, b\right)\right] \\
&\quad \approx -\frac{1}{2} \{L \log 2\pi + \log |\boldsymbol{Q}| - \log |\boldsymbol{C}| \\
&\quad + \log |\boldsymbol{K}/s| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \hat{s} \boldsymbol{K}^{-1} (\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\
&\quad + \mathbb{E}_q \left[(\boldsymbol{y} - \Phi(\boldsymbol{z}))^T \boldsymbol{Q}^{-1} (\boldsymbol{y} - \Phi(\boldsymbol{z}))\right]\} \\
&\quad - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\mathbb{E}[\log s] \\
&\quad + \Gamma(a) + (b - b_0)\hat{s} - a \log b \qquad (51)
\end{aligned}$$

Finally, we use a Taylor-series linearisation to make the remaining expectation tractable:

$$\begin{aligned}\mathcal{L}(q) \approx -\frac{1}{2}\{&L\log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| \\ &+ \log|\boldsymbol{K}/\hat{s}| + (\hat{\boldsymbol{f}}-\boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}}-\boldsymbol{\mu}) \\ &+ (\boldsymbol{y}-\Phi(\hat{\boldsymbol{z}}))^T\boldsymbol{Q}^{-1}(\boldsymbol{y}-\Phi(\hat{\boldsymbol{z}}))\Big\} \\ &- \Gamma(a_0) + a_0(\log b_0) + (a_0-a)\mathbb{E}[\log s] \\ &+ \Gamma(a) + (b-b_0)\hat{s} - a\log b, \end{aligned} \tag{52}$$

where $\Gamma()$ is the gamma function, $\mathbb{E}[\log s] = \Psi(a) - \log(b)$, and $\Psi()$ is the digamma function.

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{f}_m,s)}[\log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}_m, s|\boldsymbol{K},\alpha_0,\beta_0) - \log q(\boldsymbol{f}_m) - \log q(s)] \\ &= \sum_{p=1}^{P}\mathbb{E}_{q(\boldsymbol{f})}[\log p(y_p|f_{a_p},f_{b_p})] - \frac{1}{2}\Big\{\log|\boldsymbol{K}_{mm}| - \mathbb{E}[\log s] - \log|\boldsymbol{S}| - M \\ &\quad + \hat{\boldsymbol{f}}_m^T\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m + \mathrm{tr}(\mathbb{E}[s]\boldsymbol{K}_{mm}^{-1}\boldsymbol{S})\Big\} + \log\Gamma(\alpha) - \log\Gamma(\alpha_0) + \alpha_0(\log\beta_0) \\ &\quad + (\alpha_0-\alpha)\mathbb{E}[\log s] + (\beta-\beta_0)\mathbb{E}[s] - \alpha\log\beta, \end{aligned} \tag{53}$$

where the terms relating to $\mathbb{E}[p(\boldsymbol{f}|\boldsymbol{f}_m) - q(\boldsymbol{f})]$ cancel and we compute as follows:

$$\mathbb{E}[s] = \frac{2\alpha_0 + M}{2\beta} \tag{54}$$

$$\mathbb{E}[\log s] = \Psi(2\alpha_0 + M) - \log(2\beta), \tag{55}$$

where Ψ is the digamma function.

The gradient of $\mathcal{L}(q)$ with respect to the lengthscale, l_d, is as follows:

$$\begin{aligned}\nabla_{l_d}\mathcal{L}(q) &= -\frac{1}{2}\left\{\frac{\partial\log|\boldsymbol{K}/\hat{s}|}{\partial l_d} - \frac{\partial\log|\boldsymbol{C}|}{\partial l_d}\right. \\ &\quad \left.-(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\hat{s}\frac{\partial K^{-1}}{\partial l_d}(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\right\} \\ &= -\frac{1}{2}\left\{\frac{\partial\log|\frac{1}{\hat{s}}\boldsymbol{K}\boldsymbol{C}^{-1}|}{\partial l_d}\right. \\ &\quad \left.+\hat{s}(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\right\} \end{aligned} \tag{56}$$

Using the fact that $\log|A| = \mathrm{tr}(\log A)$, $\boldsymbol{C} = \left[\boldsymbol{K}^{-1} - \boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}^T\right]^{-1}$, and $\boldsymbol{C} = \boldsymbol{C}^T$, we obtain:

$$\begin{aligned}&= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}^T\frac{\partial\boldsymbol{K}}{\partial l_d}\right) \\ &+ \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}}-\boldsymbol{\mu}) \\ &= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\left(\boldsymbol{C}^{-1} - \boldsymbol{K}^{-1}/\hat{s}\right)\frac{\partial\boldsymbol{K}}{\partial l_d}\right) \\ &+ \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}}-\boldsymbol{\mu}). \end{aligned} \tag{57}$$

Assuming a product over kernels for each feature, $\boldsymbol{K} = \prod_{d=1}^{D} \boldsymbol{K}_d$, we can compute the kernel gradient as follows for the Matérn $\frac{3}{2}$ kernel function:

$$\frac{\partial \boldsymbol{K}}{\partial l_d} = \prod_{d'=1, d' \neq d}^{D} K_d \frac{\partial K_{l_d}}{\partial l_d} \tag{58}$$

$$\frac{\partial K_{l_d}}{\partial l_d} = \frac{3|\boldsymbol{x}_d - \boldsymbol{x}'_d|^2}{l_d^3} \exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_d - \boldsymbol{x}'_d|}{l_d}\right) \tag{59}$$

where $|\boldsymbol{x}_d - \boldsymbol{x}'_d|$ is the distance between input points.