Scalable Bayesian Preference Learning from Crowds

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

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

1 Introduction

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

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

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

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

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

2 Scalable Bayesian Preference Learning

Following Chu and Ghahramani (**?**), we model the relationship between a latent preference function, f, and each observed pairwise label, $v_k \succ u_k$, where k is an index into a list of P pairs, as follows:

$$p(v_k \succ u_k | f(v_k), f(u_k), \delta_{v_k}, \delta_{u_k}) = \begin{cases} 1 & \text{if } f(v_k) + \delta_{v_k} \geq f(u_k) + \delta_{u_k} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\delta_i \sim \mathcal{N}(0, 1)$ is Gaussian-distributed noise. The noise term allows for variations in the observed preferences, which may occur if different annotators disagree or change their minds, or if the preferences are derived from noisy implicit data such as clicks streams. We deviate from Chu and Ghahramani (**?**) by assuming δ_i has variance $\sigma = 1$, and instead scale the function f relative to this. This formulation is equivalent but is more convenient for variational inference. We marginalise the noise terms to obtain the preference likelihood:

$$p(v_k \succ u_k | f(v_k), f(u_k)) = \Phi\left(\frac{f(v_k) - f(u_k)}{\sqrt{2}}\right), \tag{2}$$

where Φ is the cumulative distribution function of a standard Gaussian distribution. For the latent function , f, we assume a Gaussian process prior: $f \sim \mathcal{GP}(0, k_\theta/s)$, where k_θ is a kernel function with hyper-parameters θ, and $s \sim \mathcal{G}(a_0, b_0)$ is an inverse scale parameter drawn from a gamma prior with shape a_0 and scale b_0. The kernel function controls the correlation between values of f at different points in the feature space.

The inference goal is to learn the posterior distribution over the function values $f(i)$ for each item i. Chu and Ghahramani (**?**) used gradient descent to optimise a Laplace approximation. However, this approach produces a maximum a-posteriori (MAP) approximation, which takes the most probable values of parameters rather than integrating over their distributions in a Bayesian manner and has been shown to perform poorly for tasks such as classification (**?**). We address the desire for a better approximation by adapting a variational method based on the extended Kalman filter (EKF) (**?**; **?**) to the preference likelihood given by Equation **??**. To make inference tractable, we approximate the preference likelihood using a Gaussian distribution: $p(v_k \succ u_k | f(v_k), f(u_k)) \approx \mathcal{N}(v_k \succ u_k; \Phi(\hat{f}_{v_k} - \hat{f}_{u_k}/\sqrt{2}), \nu_k)$. The variance v_k is estimated by moment matching with the variance of a beta posterior distribution $\mathcal{B}(p(v_k \succ u_k | f(v_k), f(u_k)); 1 + [v_k \succ u_k], 2 - v_k \succ u_k)$. The values of v_k for all pairs form a diagonal matrix $\boldsymbol{Q}$. This approximation means that the posterior distribution over f for items in the training set is also Gaussian: $p(f(\boldsymbol{x})|\boldsymbol{y}) \approx \mathcal{N}(f(\boldsymbol{x})|\hat{\boldsymbol{f}}, \boldsymbol{C})$ where $\boldsymbol{x}$ is a matrix of input features for the training items. Variational inference is then used to optimise the mean $\hat{\boldsymbol{f}}$ and covariance $\boldsymbol{C}$. by maximising a lower bound, $\mathcal{L}$, on the log marginal likelihood, $p(\boldsymbol{y}|\theta, a_0, b_0)$:

$$\begin{aligned}\mathcal{L}(q) \approx &-\frac{1}{2}\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}| \\ &+ (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &+ (\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))\} \\ &+ \Gamma(a) - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\widehat{\ln s} \\ &+ (b - b_0)\hat{s} - a \log b, \end{aligned} \tag{3}$$

where L is the number of observed preference labels, $\boldsymbol{y} = [v_1 \succ u_1, ..., v_L \succ u_L]$ is a vector of binary preference labels, $\widehat{\ln s}$ and $\hat{s}$ are expected values of s, and $\hat{\boldsymbol{z}} = \left\{\frac{\hat{f}_{v_k} - \hat{f}_{u_k}}{\sqrt{2}} \forall k = 1, ..., P\right\}$.

The variational approach described so far requires a scalable inference algorithm. We therefore adapt stochastic variational inference (SVI) (**?**; **?**) to preference learning. For SVI, we assume M *inducing points* with features $\boldsymbol{x}_m$. The inducing points act as a substitute for the complete set of feature vectors of the observed arguments, and allow us to choose $M << N$ to limit the computational and memory requirements. To choose representative inducing points, we use K-means to rapidly cluster the feature vectors, then used the cluster centres as inducing points. Given the inducing points, SVI further limits computational costs by using an iterative algorithm that only considers a subset of the data containing $P_n << P$ pairs at each iteration. The algorithm proceeds as follows:

1. Randomly initialise the mean at the inducing points, $\hat{\boldsymbol{f}}_m$, the covariance of the inducing points, $\boldsymbol{S}$, the inverse function scale expectations $\hat{s}$ and $\widehat{\ln s}$, and the Jacobian of the pairwise label probabilities, $\boldsymbol{G}$.

2. Select a random subset of P_n pairwise labels.

3. Compute the mixing coefficient, $\rho_i = (n + \text{delay})^{-\text{forgetting}_\text{rate}}$, which controls the rate of change of the estimates, and the weight $w_n = \frac{P}{P_n}$, which weights each update according to the size of the random subsample.

4. Update each variable in turn using equations below.

5. Repeat from step 2 until convergence.

6. Use converged values to make predictions.

The equations for the updates at iteration n are as follows:

$$\boldsymbol{S}_n^{-1} = (1-\rho_n)\boldsymbol{S}_{n-1}^{-1} + \rho_n \left(\hat{s}\boldsymbol{K}_{mm}^{-1} \right. \\ \left. + w_n \boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{nm}^T \boldsymbol{G}^T \boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-T}\right) \quad (4)$$

$$\hat{\boldsymbol{f}}_{m,n} = \boldsymbol{S}_n \left((1-\rho_n)\boldsymbol{S}_{n-1}^{-1}\hat{\boldsymbol{f}}_{m,n-1} + \rho_n w_n \boldsymbol{K}_{mm}^{-1} \right. \\ \left. \boldsymbol{K}_{nm}^T \boldsymbol{G}^T Q^{-1} 1 \left(\frac{1 + [v_k \succ u_k]}{3} - \Phi(\hat{\boldsymbol{z}}_n) - \boldsymbol{G}\hat{\boldsymbol{f}} \right) \right) \quad (5)$$

$$\hat{s} = \frac{2a_0 + N}{2b} \quad (6)$$

$$\widehat{\ln s} = \Psi(2a_0 + N) - \log(2b) \quad (7)$$

$$\boldsymbol{G} = \frac{1}{2\pi}\exp\left(-\frac{1}{2}\hat{\boldsymbol{z}}_n^2\right) \quad (8)$$

where $\boldsymbol{K}_{mm}$ is the covariance between values at the inducing points, $\boldsymbol{K}_{nm}$ is the covariance between the subsample of pairwise labels and the inducing points, $\hat{\boldsymbol{z}}_n$ is the estimated preference label likelihood for the nth subsample, $b = b_0 + \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{K}_j^{-1}\left(\boldsymbol{\Sigma}_j + (\hat{\boldsymbol{f}}_j - \boldsymbol{\mu}_{j,i})(\hat{\boldsymbol{f}}_j - \boldsymbol{\mu}_{j,i})^T\right)\right)$, and Ψ is the digamma function. Given the converged estimates, we can make predictions for test arguments with feature vectors $\boldsymbol{x}_*$. The posteriors for the latent function values $\boldsymbol{f}_*$ at the test points have mean and covariance given by:

$$\hat{\boldsymbol{f}}_* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m \quad (9)$$

$$\boldsymbol{C}_* = \frac{\boldsymbol{K}_{**}}{\hat{s}} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm})\boldsymbol{K}_{mm}^{-T}\boldsymbol{K}_{*m}^T, \quad (10)$$

where $\boldsymbol{K}_{*m}$ is the covariance between the test items and the inducing points.

2.1 Kernel Length-scale Optimsation

The prior covariance of the latent function f is defined by a kernel k, typically of the form $k_\theta(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d(|x_d - x'_d|/l_d)$, where k_d is a function of the distance between the values of feature d for item x and x', and a length-scale hyper-parameter, l_d, which controls the smoothness of the function across the feature space. The product over features D may be replaced by other combinations, such as a sum. There are several ways to set l, including the median heuristic (**?**): $l_{d,MH} = \frac{1}{D}\mathrm{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1,..,N, \forall j = 1,...,N\})$. We can also optimise l_d by choosing the value that maximises the lower bound on the log marginal likelihood, $\mathcal{L}$, defined in Equation **??**. This process is known as maximum likelihood II and is often referred to as automatic relevance determination (ARD) (**?**), since features with large length-scales are less relevant because their values have less effect on $k_\theta(\boldsymbol{x}, \boldsymbol{x}')$ than features with short length-scales. The cost of optimisation may be reduced by simultaneously optimising all length-scales using a gradient-based method such as L-BFGS-B (**?**). Given our proposed SVI method, we substitute our inducing point approximation into Equation **??** to approximate the gradients of $\mathcal{L}(q)$ with respect to l_d as follows:

$$\nabla_{l_d}\mathcal{L}(q) = \frac{1}{2}\hat{s}\hat{\boldsymbol{f}}_m^T \boldsymbol{K}_{mm}^{-T}\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m \\ - \frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}_{mm}^{-1}\boldsymbol{S}\right)^T\left(\boldsymbol{S}^{-1} - \boldsymbol{K}_{mm}^{-1}/\hat{s}\right)\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\right). \quad (11)$$

In our implementation, we choose the Matèrn $\frac{3}{2}$ kernel function for k due to its general properties of smoothness (**?**), so that the matrix of partial derivatives is:

$$\frac{\partial \boldsymbol{K}}{\partial l_d} = \frac{\boldsymbol{K}}{k_d(|\boldsymbol{x}_d, \boldsymbol{x}'_d)}\frac{\partial \boldsymbol{K}_{l_d}}{\partial l_d}, \quad (12)$$

where each entry ij of the $\frac{\partial \boldsymbol{K}_{l_d}}{\partial l_d}$ is defined as:

$$\frac{\partial K_{d,ij}}{\partial l_d} = \frac{3|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|}{l_d}\right). \quad (13)$$

3 Experiments

4 Conclusions and Future Work

Acknowledgments

A Variational Inference

We derive the variational lower bound as follows:

$$\mathcal{L}(q) = \sum_{i=1}^{L} \mathbb{E}_q \left[\log p\left(v_i \succ u_i | f(v_i), f(u_i)\right)\right] + \mathbb{E}_q\left[\log \frac{p\left(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K}/s\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_q\left[\log \frac{p\left(s|a_0, b_0\right)}{q\left(s\right)}\right] \tag{14}$$

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

$$\begin{aligned}\mathcal{L}(q) =& \mathbb{E}_q\Bigg[\sum_{i=1}^{L}[v_i \succ u_i] \log \Phi(z_i) \\ & + [v_i \prec u_i]\left(1 - \log \Phi(z_i)\right)\Bigg] \\ & + \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\hat{\boldsymbol{f}}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ & + \mathbb{E}_q\left[\log \mathcal{G}\left(s; a_0, b_0\right) - \log \mathcal{G}\left(s; a, b\right)\right]\end{aligned} \tag{15}$$

We now replace the likelihood with a Gaussian approximation:

$$\begin{aligned}\mathcal{L}(q) \approx\ & \mathbb{E}_q\left[\mathcal{N}(\boldsymbol{y}|\Phi(\boldsymbol{z}), \boldsymbol{Q})\right] \\ & + \log \mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log \mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ & + \mathbb{E}_q\left[\log \mathcal{G}\left(s; a_0, b_0\right) - \log \mathcal{G}\left(s; a, b\right)\right] \\ \approx & -\frac{1}{2}\Big\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| \\ & + \log|\boldsymbol{K}/s| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ & + \mathbb{E}_q\left[(\boldsymbol{y} - \Phi(\boldsymbol{z}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\boldsymbol{z}))\right]\Big\} \\ & - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\mathbb{E}[\log s] \\ & + \Gamma(a) + (b - b_0)\hat{s} - a \log b\end{aligned} \tag{16}$$

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

$$\begin{aligned}\mathcal{L}(q) \approx & -\frac{1}{2}\Big\{L \log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| \\ & + \log|\boldsymbol{K}/\hat{s}| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ & + (\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))^T \boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))\Big\} \\ & - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\mathbb{E}[\log s] \\ & + \Gamma(a) + (b - b_0)\hat{s} - a \log b,\end{aligned} \tag{17}$$

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

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

$$\begin{aligned}\nabla_{l_d}\mathcal{L}(q) =& -\frac{1}{2}\Bigg\{\frac{\partial \log|\boldsymbol{K}/\hat{s}|}{\partial l_d} - \frac{\partial \log|\boldsymbol{C}|}{\partial l_d} \\ & -(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\frac{\partial K^{-1}}{\partial l_d}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\Bigg\} \\ =& -\frac{1}{2}\Bigg\{\frac{\partial \log|\frac{1}{\hat{s}}\boldsymbol{K}\boldsymbol{C}^{-1}|}{\partial l_d} \\ & + \hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\Bigg\}\end{aligned} \tag{18}$$

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:

$$\begin{aligned} &= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}^T\frac{\partial \boldsymbol{K}}{\partial l_d}\right) \\ &+ \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\left(\boldsymbol{C}^{-1} - \boldsymbol{K}^{-1}/\hat{s}\right)\frac{\partial \boldsymbol{K}}{\partial l_d}\right) \\ &+ \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}).\end{aligned} \tag{19}$$

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:

$$\frac{\partial \boldsymbol{K}}{\partial l_d} = \prod_{d'=1, d' \neq d}^{D} K_d \frac{\partial K_{l_d}}{\partial l_d} \tag{20}$$

$$\frac{\partial K_{l_d}}{\partial l_d} = \frac{3|\boldsymbol{x}_d - \boldsymbol{x}'_d|^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_d - \boldsymbol{x}'_d|}{l_d}\right) \tag{21}$$

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