Sparse Gaussian Processes for Learning Preferences

M. Ehsan Abbasnejad
Australian National University & NICTA
7 London Circuit, Canberra
ehsan.abbasnejad@nicta.com.au

Edwin V. Bonilla
NICTA & Australian National University
7 London Circuit, Canberra
edwin.bonilla@nicta.com.au

Scott Sanner
NICTA & Australian National University
7 London Circuit, Canberra
scott.sanner@nicta.com.au

Abstract

Preference learning has recently gained significant attention in the machine learning community. This is mainly due to its increasing applications in real-world problems such as recommender systems. In this paper, we investigate a Gaussian process framework for learning preferences that uses Expectation Propagation (EP) as its main inference method. This framework is capable of using the collaborative information from all the users for prediction of preferences unlike traditional approaches that only consider single users. We further extend this framework to a sparse setting and show its empirical efficiency. The contribution of this paper is the use of sparse Gaussian process for multi-user preference learning and the comparison of this approach with full EP and the Laplace approximation.

1 Introduction

With the rise of interest in providing personalized services, we witness significant contemporary attention to recommender systems. Any recommender system seeks to learn the preferences of its users for the products and the services it provides. It is a challenging problem since, unlike traditional cases, the relation between two sets–users on one side and their preferred products (items) on the other side–should be uncovered. In this paper, we address this problem using Gaussian processes (GPs) [6]. GPs are a set of non-parametric Bayesian methods that have solid theoretical foundation and proved empirical effectiveness. GPs have been used in [2] for preference learning and extended by [1] to the multi-user case. In the latter approach, poterior inference is carried out with the Laplace approximation, which scales poorly with the number of users and items. In this paper we extend the model proposed in [1] and seek to improve its scalability by sequentially performing local updates. Subsequently, a sparse version of our method is used and shown to predict preferences better than its counterparts. This sparse method makes the model applicable to larger datasets.

In general pairwise preference learning problem, we are given a set of n users $U = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ and m items $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. The preferences of each user $\mathbf{u} \in U$ are given by $\mathcal{D} = \{\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}}\}$ with $1 \leq i, j \leq m$. We denote the latent functions for all the users and items with $\mathbf{f} = [f_1^{\mathbf{u}_1}, f_2^{\mathbf{u}_1}, \ldots, f_m^{\mathbf{u}_n}]$ f and define the likelihood over all the preferences given the latent functions as:

$$p(\mathcal{D}|\mathbf{f}) = \prod_{\mathbf{u} \in U} \prod_{\{i,j\} \in \mathcal{D}} p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}), \tag{1}$$

with

$$p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) = \Phi\left(\frac{f_i^{\mathbf{u}} - f_j^{\mathbf{u}}}{\gamma}\right), \tag{2}$$

where $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(y; 0, 1) dy$ and $\mathcal{N}(y; 0, 1)$ is a zero-mean Gaussian distribution with variance one. In this model, $p(\mathbf{f})$ is the prior over the latent functions that is defined as a zero mean Gaussian distribution with a covariance (kernel) matrix $\mathbf{K}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \mathbf{0}, \mathbf{K}), \quad \mathbf{K} = \mathbf{K}_u \otimes \mathbf{K}_x. \tag{3}$$

In this equation, $\mathbf{K}$ is defined as in [1] where the kernel matrix is composed of the product of the kernel matrix over the users $\mathbf{K}_u$ and the kernel matrix over the items $\mathbf{K}_x$. One interesting advantage of such a kernel matrix is the inherent transfer of preferences across users, which will subsequently help the prediction on those users who do not have many preferences recorded. Given the likelihood model and the prior model above, the posterior of the latent functions $\mathbf{f}$ given all the preferences is:

$$p(\mathbf{f}|\mathcal{D}) = \frac{1}{Z} p(\mathbf{f}) p(\mathcal{D}|\mathbf{f}), \tag{4}$$

with Z being the normalizer. This posterior is analytically intractable due the non-Gaussian nature of the likelihood. Moreover, since the kernel matrix $\mathbf{K}$ can be very large due to the number of users and items, we need to use an efficient approximate method to compute the posterior. We further detail the approximate posterior in the subsequent section where the EP method for preference learning in this framework is described and an efficient update upon observing a pair of items is found.

2 Expectation Propagation for Preference Learning

Expectation propagation (EP) [5] is an approximate inference method that stemmed from Assumed Density Filtering (ADF). EP seeks to minimize the KL divergence between the true distribution and the approximated one in an iterative manner. In a nutshell, EP assumes that the distribution under investigation is factorized to a product of base (potentially simpler) distributions, i.e. to approximate the true distribution $p(\mathbf{z})$ with $q(\boldsymbol{\theta})$ we assume q is factorized as:

$$p(\mathbf{z}|\boldsymbol{\theta}) \approx q(\boldsymbol{\theta}) = \frac{\prod_i t_i(\boldsymbol{\theta})}{\int \prod_i t_i(\boldsymbol{\theta})} \tag{5}$$

Subsequently, EP approximates each factor by replacing it with its true likelihood and minimizing the KL divergence between the product of the true likelihood and the rest of the factors with its approximation. Consequently, in case each of the factors are Gaussian distributions, they can be updated by moment matching with the following formula:

$$t_i(\boldsymbol{\theta}) = Z_i q(\boldsymbol{\theta}) / q_{\setminus i}(\boldsymbol{\theta}) \tag{6}$$

where we use $\_$ in the index to denote the elimination of that factor. By iterating on the factors until convergence the approximate distribution can be found. Now we can extend EP to preference learning: this is done by firstly writing the desired posterior in a factorized form by expanding Eq. 4:

$$p(\mathbf{f}|\mathcal{D}) = \frac{1}{Z} p(\mathbf{f}) \prod_{\mathbf{u} \in U} \prod_{\{i,j\} \in \mathcal{D}} p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) \tag{7}$$

We are interested in locally approximating each preference in Eq. 7 as:

$$\begin{aligned} p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) \approx t(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) &= \tilde{Z}_{i,j}^{\mathbf{u}} \mathcal{N}(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}; \tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]}, \tilde{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]}) \\ \tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} \in \mathbb{R}^{2\times 1}, \tilde{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]} \in \mathbb{R}^{2\times 2}. \end{aligned} \tag{8}$$

In this equation, we used the convention of denoting a bivariate Gaussian distribution with $\mathcal{N}(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}; \tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]}, \tilde{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]})$ over $f_i^{\mathbf{u}}$ and $f_j^{\mathbf{u}}$, $\tilde{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]} = \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_{ii} & \tilde{\boldsymbol{\Sigma}}_{ij} \\ \tilde{\boldsymbol{\Sigma}}_{ji} & \tilde{\boldsymbol{\Sigma}}_{jj} \end{bmatrix}$ and similarly $\tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} = \begin{bmatrix} \tilde{\boldsymbol{\mu}}_i \\ \tilde{\boldsymbol{\mu}}_j \end{bmatrix}$ of the user $\mathbf{u}$'s block. Then we can approximate $p(\mathbf{f}|\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}})$ by $q(\mathbf{f})$:

$$q(\mathbf{f}) = \frac{1}{\tilde{Z}} p(\mathbf{f}) \prod_{\mathbf{u} \in U} \prod_{\{i,j\} \in \mathcal{D}} t(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{9}$$

where

$$\boldsymbol{\mu}_{\mathbf{u},[i,j]} = \boldsymbol{\Sigma}_{\mathbf{u},[i,j]}\boldsymbol{\nu}_{\mathbf{u},[i,j]} \quad , \qquad \boldsymbol{\Sigma}^{-1}_{\mathbf{u},[i,j]} = (\mathbf{K}^{-1}_{\mathbf{u},[i,j]} + \boldsymbol{\Lambda}_{\mathbf{u},[i,j]})$$
$$\boldsymbol{\nu}_{\mathbf{u},[i,j]} = \tilde{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]}\tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} \quad , \qquad \boldsymbol{\Lambda}_{\mathbf{u},[i,j]} = \tilde{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]} \tag{10}$$

Subsequently, we are interested in finding $\boldsymbol{\Lambda}$ (a $(n \times m) \times (n \times m)$ matrix ultimately corresponding to $\tilde{\boldsymbol{\Sigma}}$) and $\boldsymbol{\nu}$ (a $n \times m$ vector ultimately corresponding to $\tilde{\boldsymbol{\mu}}$). Finding these parameters will lead to determining the factors in the approximated posterior in Eq. 8. As such, in line with the EP procedure, a factor is replaced with its true likelihood and then projected back. This is done by iterating through the following steps:

1. **Dividing a factor from the posterior:** In EP, the minimization of the difference between the product of the exact likelihood and the rest of the factors is done to locally update each factor. To do that, firstly, we define $\hat{q}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j)$ as this local product:

$$\hat{q}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) = p(\mathbf{x}^{\mathbf{u}}_i \succ \mathbf{x}^{\mathbf{u}}_j | f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) q_{\backslash}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j), \tag{11}$$

where $q_{\backslash}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j)$ is derived from q when the factor corresponding to $f^{\mathbf{u}}_i$ and $f^{\mathbf{u}}_j$ is not taken into account, i.e.

$$q_{\backslash}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) = \frac{q(\mathbf{f})}{t(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j)} \tag{12}$$

Therefore, the value of $q_{\backslash}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j)$ is computed from Eq. 9,

$$q_{\backslash}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) = \mathcal{N}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j; \boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]}, \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}) \tag{13}$$

and consequently:

$$\begin{aligned} \boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]} &= \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}({\boldsymbol{\Sigma}_{\mathbf{u},[i,j]}}^{-1}\boldsymbol{\mu}_{\mathbf{u},[i,j]} - \tilde{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]}\tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]}) \\ \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]} &= (\boldsymbol{\Sigma}^{-1}_{\mathbf{u},[i,j]} - \tilde{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]})^{-1} \end{aligned} \tag{14}$$

2. **Projecting back to the approximation:** Once a factor $t(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j)$ is replaced by its true likelihood, we need to calculate its product with all the other approximated factors in Eq. 11. As such, we approximate this product with an unnormalized Gaussian:

$$\hat{q}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) \approx \hat{Z}_{\mathbf{u},[i,j]}\mathcal{N}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j; \hat{\boldsymbol{\mu}}_{\mathbf{u},[i,j]}, \hat{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]}) \tag{15}$$

Therefore, if we are able to calculate the parameters of this distribution, we will be able to update the value of the factor we have pulled out in the previous step (namely $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$). Ultimately by calculating $\hat{q}$ and projecting it back to its approximation, we will be able to locally update the factor we have pulled out. Thus, the corresponding values in Eq. 11 are replaced[1]:

$$\hat{q}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j) = \hat{Z}^{-1}\mathcal{N}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j; \boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]}, \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]})\Phi\left(\frac{f^{\mathbf{u}}_i - f^{\mathbf{u}}_j}{\gamma}\right) \tag{16}$$

where

$$\begin{aligned} \hat{Z} &= \int_{-\infty}^{\infty} \mathcal{N}(f^{\mathbf{u}}_i, f^{\mathbf{u}}_j; \boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]}, \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]})\Phi(\frac{f^{\mathbf{u}}_i - f^{\mathbf{u}}_j}{\gamma}) = \Phi(\frac{\boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]}\mathbf{1_1}}{\gamma^2 + \text{tr}(\boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}\mathbf{1_2})}) = \Phi(r_{i,j}) \\ & \mathbf{1_1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{1_2} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \end{aligned} \tag{17}$$

From this equation, the corresponding moments in $\hat{q}$ are calculated as:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} &= \boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]} + \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}\mathbf{w}_{\mathbf{u},[i,j]} \\ \hat{\boldsymbol{\Sigma}}_{\mathbf{u},[i,j]} &= \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]} - \boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}(\mathbf{w}_{\mathbf{u},[i,j]}\mathbf{w}^{\top}_{\mathbf{u},[i,j]} + \frac{r_{i,j}}{\gamma^2 + \text{tr}(\boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]}\mathbf{1_2})}\mathbf{w}_{\mathbf{u},[i,j]}{\mathbf{1_1}}^{\top})\boldsymbol{\Sigma}_{\backslash\mathbf{u},[i,j]} \end{aligned} \tag{18}$$

[1]Similar results are obtained for EP in [2]

where

$$\mathbf{w}_{\mathbf{u},[i,j]} = \frac{\mathcal{N}(r_{i,j})}{\Phi(r_{i,j})(\gamma^2 + \text{tr}(\mathbf{\Sigma}_{\backslash \mathbf{u},[i,j]}\mathbf{1_2}))}\mathbf{1_1} \tag{19}$$

and since

$$\hat{q}(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) \approx p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) q_{\backslash}(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) \Leftrightarrow q_{\backslash}(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) t(f_i^{\mathbf{u}}, f_j^{\mathbf{u}}) \tag{20}$$

by performing moment matching, we can calculate the corresponding parameters in $t(f_i^{\mathbf{u}}, f_j^{\mathbf{u}})$ as:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} &= \tilde{\mathbf{\Sigma}}_{\mathbf{u},[i,j]}(\hat{\mathbf{\Sigma}}^{-1}_{\mathbf{u},[i,j]}\hat{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} - \mathbf{\Sigma}^{-1}_{\backslash\mathbf{u},[i,j]}\boldsymbol{\mu}_{\backslash\mathbf{u},[i,j]}) \\ \tilde{\mathbf{\Sigma}}_{\mathbf{u},[i,j]} &= (\hat{\mathbf{\Sigma}}^{-1}_{\mathbf{u},[i,j]} - \mathbf{\Sigma}^{-1}_{\backslash\mathbf{u},[i,j]})^{-1} \end{aligned} \tag{21}$$

By having the values of $\tilde{\boldsymbol{\mu}}$ and $\tilde{\mathbf{\Sigma}}$ discovered, we can iterate through all the factors and update the local approximations sequentially. This leads to a simple update in Eq. 9 that completes the EP algorithm.

3 Sparse Approximation

Since the prediction in Gaussian processes scales poorly with the number of training examples, sparse methods have been proposed to enhance this phase by selecting a subset that contributes most to the final decision. This leads to a more efficient prediction and eliminates the need to keep the whole training set around. Here we extend one of the well-known sparse methods, namely *informative vector machine* (IVM) [4], to the preference learning. In compliance with IVM, we try to keep the points for prediction that have the highest impact on the factorized latent functions in EP. This impact is measured by the differential entropy $\delta^e = H(t_i^{\text{new}}) - H(t_i^{\text{old}})$. Since all the approximations are in Gaussian, the entropy only depends on the value of the variance. Additionally, because the variance is only updated in each iteration, it is easy to show that the value of this differential entropy is equal to:

$$\delta^e_{\mathbf{u},i} = \frac{1}{2}\log(1 + \mathbf{\Sigma}^{\text{new}}_{\mathbf{u},ii}/\mathbf{\Sigma}^{\text{old}}_{\mathbf{u},ii}) \tag{22}$$

where $\mathbf{\Sigma}^{\text{old}}_{\mathbf{u},ii}$ and $\mathbf{\Sigma}^{\text{new}}_{\mathbf{u},ii}$ denotes the variance of item i and its updated value. The latent functions that maximize this entropy are the ones that keep decreasing the variance.

In our case, since in each iteration we have two items for each user to evaluate, we calculate these values separately. At the end, we will select a subset of users and items (with given parameters ℓ_u and ℓ_x) that maximized this entropy. It should be noted that this approach is different from the original IVM algorithm where the entropy is calculated at each iteration and then the data point that maximizes this entropy is selected and subsequently included in the active set as one of the factors of the approximated marginal. Therefore, we are calculating the sum of entropy of including each factor and approximating final marginal with the most *informative* ones, i.e.

$$p(\mathbf{f}|\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}}) = \frac{1}{Z}p(\mathbf{f}) \prod_{\mathbf{u}\in\max_{\ell_u}(\delta^e)} \prod_{i,j\in\max_{\ell_x}(\delta^e)} p(\mathbf{x}_i^{\mathbf{u}} \succ \mathbf{x}_j^{\mathbf{u}} | f_i^{\mathbf{u}}, f_j^{\mathbf{u}}), \tag{23}$$

where we have used $\max_{\ell_u}(\delta^e)$ and $\max_{\ell_u}(\delta^e)$ to denote the set of users and items that maximized the differential entropy. Intuitively, it means we only take into account the pair of items that are important in building our belief about the preferences. This procedure leads to a simple and efficient approach that is described in Algorithm 1.

Algorithm 1 Sparse EP-Preference GP

```
$\mathbf{K}^{\text{inv}} \leftarrow \mathbf{K}_u^{-1} \otimes \mathbf{K}_x^{-1}$
$\boldsymbol{\Sigma} \leftarrow \mathbf{K}^{\text{inv}}$
**while** not converged **do**
  **for** each pair of items $i, j$ in preference set **do**
    $\boldsymbol{\Lambda}_{\setminus \mathbf{u},[i,j]} \leftarrow \boldsymbol{\Sigma}^{-1}_{\mathbf{u},[i,j]} - \boldsymbol{\Lambda}_{\mathbf{u},[i,j]}$
    $\boldsymbol{\nu}_{\setminus \mathbf{u},[i,j]} \leftarrow \boldsymbol{\Sigma}^{-1}_{\mathbf{u},[i,j]} \boldsymbol{\mu}_{\mathbf{u},[i,j]} - \boldsymbol{\nu}_{\mathbf{u},[i,j]}$
    $\hat{\boldsymbol{\Sigma}}$, $\hat{\boldsymbol{\mu}}$ from Eq. 18
    $\Delta\boldsymbol{\Lambda} \leftarrow \hat{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]} - \boldsymbol{\Lambda}_{\setminus \mathbf{u},[i,j]} - \boldsymbol{\Lambda}_{\mathbf{u},[i,j]}$
    $\boldsymbol{\Sigma}_{\mathbf{u},[i,j]} \leftarrow \boldsymbol{\Lambda}_{\mathbf{u},[i,j]} + \Delta\boldsymbol{\Lambda}$
    $\boldsymbol{\nu}_{\mathbf{u},[i,j]} \leftarrow \hat{\boldsymbol{\Sigma}}^{-1}_{\mathbf{u},[i,j]} \hat{\boldsymbol{\mu}}_{\mathbf{u},[i,j]} - \boldsymbol{\nu}_{\setminus \mathbf{u},[i,j]}$
    $\boldsymbol{\Lambda}_{\mathbf{u},[i,j]} \leftarrow \mathbf{K}^{\text{inv}}_{\mathbf{u},[i,j]} + \boldsymbol{\Lambda}_{\mathbf{u},[i,j]}$
    $\delta^e_{\mathbf{u},i} = \frac{1}{2}\log(1 + \boldsymbol{\Lambda}^{new}_{\mathbf{u},ii} / \boldsymbol{\Lambda}^{\text{old}}_{\mathbf{u},ii}) + \delta^e_{\mathbf{u},i}$
    $\delta^e_{\mathbf{u},j} = \frac{1}{2}\log(1 + \boldsymbol{\Lambda}^{new}_{\mathbf{u},jj} / \boldsymbol{\Lambda}^{\text{old}}_{\mathbf{u},jj}) + \delta^e_{\mathbf{u},j}$
  **end for**
**end while**
$\mathcal{U} = \text{argmax}_{\ell_u}(\delta^e)$
$\mathcal{X} = \text{argmax}_{\ell_x}(\delta^e)$
**return** $\boldsymbol{\Lambda}_{\mathcal{U},\mathcal{X}}, \boldsymbol{\nu}_{\mathcal{U},\mathcal{X}}, \mathcal{U}, \mathcal{X}$
```

4 Prediction

Once the posterior is computed and the parameters are learned, given a pair of items $\mathbf{x}_1^*, \mathbf{x}_2^*$ we will be able to determine their preference for a user $\mathbf{u}^*$ by integrating out the latent function:

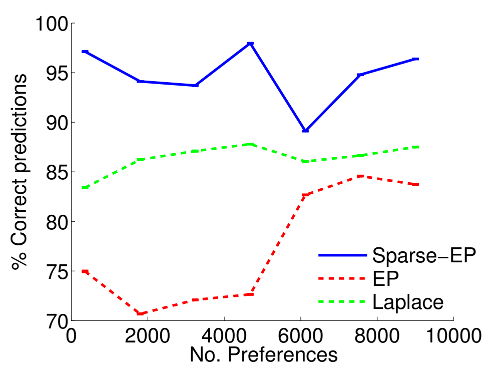
$$\begin{aligned} p(\mathbf{x}_1^* \succ \mathbf{x}_2^* | \mathcal{D}) &= \int p(\mathbf{x}_1^* \succ \mathbf{x}_2^* | f_1^*, f_2^*, \mathcal{D}) p(f_1^*, f_2^* | \boldsymbol{f}, \mathcal{D}) p(\boldsymbol{f}|\mathcal{D}) d\boldsymbol{f} \\ &= \Phi\left(\frac{\mathbf{K}^{*\top}\boldsymbol{\nu}\mathbf{1_1}}{\gamma^2 + \text{tr}((\boldsymbol{\Sigma}^* - \mathbf{K}^{*\top}\boldsymbol{\Lambda}\mathbf{K}^*)\mathbf{1_2})}\right) \end{aligned} \tag{24}$$

where $\mathbf{K}^* = \mathbf{K}_u^* \otimes \mathbf{K}_x^*$ that represents the kernel matrix of the test user and items with all the users and items in the training set, $\mathbf{K}_u^*$ is the $1 \times n$ kernel matrix of the queried user with other users, $\mathbf{K}_x^*$ is the $2 \times m$ kernel matrix of the queried pair of items with other items and $\boldsymbol{\Sigma}^*$ is the 2×2 kernel matrix built from the item pair $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$.

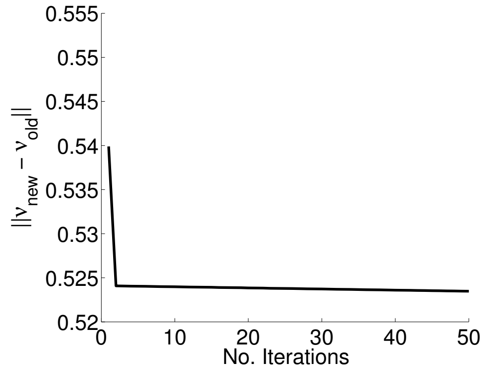
If the confidence on the predictions is not required, $p(\mathbf{x}_1^* \succ \mathbf{x}_2^* | \mathcal{D})$ only depends on $\boldsymbol{\nu}$ and $\mathbf{K}^*$. Compared to the case where Laplace approximation is used for inference, we can save time by using the $\boldsymbol{\nu}$ computed directly by EP instead of calculating it from the mean and covariance matrix of the model. In case Sparse-EP is opted for approximation, the subset of users and items and corresponding $\boldsymbol{\nu}$ is used.

5 Empirical Results

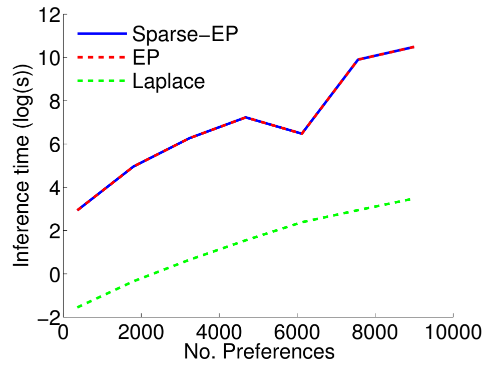
To evaluate the performance of the algorithm, the *sushi* dataset [3] is used. It is a dataset of preferences of people about 10 types of sushis which leads to 45 preferences (pairs of sushis) per user. In this experiment, we choose various subsets of users and select 80% of their preferences for inference and the rest for testing. We use the Gaussian kernel in this algorithm and set the hyper-parameters for both item and user kernels to 10 and 1 respectively. It should here be noted that in such problems, the kernel function plays a crucial role and needs to be learned, however, we left it for future work. The results of running the algorithm is shown in Fig. 1. As can be seen, Sparse-EP approach produced the best results in terms of correct prediction followed by EP and Laplace (from [1]). In Sparse-EP, 4 users and 8 items with the highest entropy are selected for the prediction. Furthermore, the time required for approximating the posterior is the fastest when Laplace approximation is used because in each step the whole matrix is updated at once unlike EP that iterates all the preferences



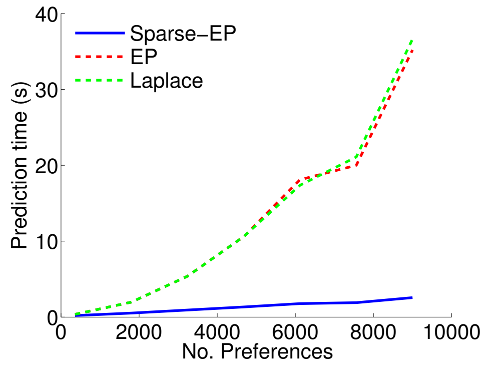
(a) The percentage of correctly predicted test examples based on the given number of preferences for inference



(b) The diagram of convergence of EP with 210 users and 7560 preferences for inference



(c) The diagram of posterior approximation time required for each of the algorithms



(d) The diagram of prediction time required for each of the algorithms

Figure 1: This figure illustrates the results from running the EP, Sparse-EP and Laplace on the Gaussian Process for preference learning

and may not converge quickly. It is specially interesting to note since EP takes longer to converge when the number of users increase (in particular when it exceeds 200 corresponding to more than 6000 preferences), we observe a sudden change in the time required for inference in EP and Sparse-EP. Probably by changing the hyper-parameters of the kernel and the convergence criteria, we can deter such a sudden change. Fianlly, it is shown that Sparse-EP outperforms others in prediction time by being the fastest.

6 Conclusion and Future work

This work is an initiative to using sparse approximations for multi-user preference learning. It is shown that the sparse extension of EP can be an efficient algorithm for learning the preferences. In future, we plan to extend this work to the case where hyper-parameters are learned during training and we hypothesize that this will dramatically increase the performance. Furthermore, the use of other customized kernels that are more suitable for preference learning should be tested.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. Gaussian process preference elicitation. *Advances in Neural Information Processing Systems*, 20:153–160, 2010.

[2] Wei Chu and Zoubin Ghahramani. Extensions of gaussian processes for ranking: semi-supervised and active learning. *Learning to Rank*, page 29, 2005.

[3] Toshihiro Kamishima. Nantonac collaborative filtering: recommendation based on order responses. *Proceedings of the ninth ACM SIGKDD international*, 2003:583–588, 2003.

[4] Neil Lawrence, Portobello Street, Matthias Seeger, Forrest Hill, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. *Advances in neural information*, 2003.

[5] Tom P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, 2001.

[6] Carl Edward Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*.