Gaussian Process Factorization Machines for Context-aware Recommendations

Trung V. Nguyen*
ANU & NICTA
Canberra, Australia
u5075561@anu.edu.au

Alexandros Karatzoglou
Telefonica Research
Barcelona, Spain
alexk@tid.es

Linas Baltrunas
Telefonica Research
Barcelona, Spain
linas@tid.es

ABSTRACT

Context-aware recommendation (CAR) can lead to significant improvements in the relevance of the recommended items by modeling the nuanced ways in which context influences preferences. The dominant approach in context-aware recommendation has been the multidimensional *latent factors* approach in which users, items, and context variables are represented as latent features in a low-dimensional space. An interaction between a user, item, and a context variable is typically modeled as some *linear* combination of their latent features. However, given the many possible types of interactions between user, items and contextual variables, it may seem unrealistic to restrict the interactions among them to linearity.

To address this limitation, we develop a novel and powerful non-linear probabilistic algorithm for context-aware recommendation using Gaussian processes. The method which we call Gaussian Process Factorization Machines (GPFM) is applicable to both the explicit feedback setting (e.g. numerical ratings as in the Netflix dataset) and the implicit feedback setting (i.e. purchases, clicks). We derive stochastic gradient descent optimization to allow scalability of the model. We test GPFM on five different benchmark contextual datasets. Experimental results demonstrate that GPFM outperforms state-of-the-art context-aware recommendation methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords

Collaborative Filtering, Context-Aware Recommendation, Gaussian Processes, Implicit Feedback, Nonlinear, Probabilistic Modeling

*Part of this work was conducted when the first author was an intern at Telefonica Research, Barcelona.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

ACM 978-1-4503-2257-7/14/07 ...$15.00.
http://dx.doi.org/10.1145/2600428.2609623.

1. INTRODUCTION

Collaborative Filtering (CF) methods capable of modeling vast amounts of user data now make it possible for on-line stores and content providers to recommend items tailored to a specific user's interests. Companies are finding that getting those personalized recommendations right - or even close - can mean significantly higher user engagement and sales. This is the driving force behind research on methods that retrieve relevant items to be recommended to on-line users. Recommendation is an Information Retrieval problem whereby the task is to retrieve for a specific user a relatively small number (5-100) of "relevant" items out of an inventory of potentially tens of thousands of items.

CF methods predict the preferences of users based on their collective past consumption behavior, which can be effectively inferred from the data traces stored in web-logs. These traces come either in the form of implicit feedback, that is we know which items a user interacted with, e.g., purchased, used, or clicked, etc., or in the form of explicit ratings e.g. in a 1 to 5 stars Likert scale. The learning of preference functions in the context of collaborative filtering using implicit or explicit feedback data can be cast as either a regression problem where a potential rating is to be predicted, [8, 29], or a ranking problem where an optimal list of items is to be computed [20, 26, 30].

Context. In the quest for the perfectly relevant recommendations it is essential to use all the information that is available and can influence the relevance of an item. This additional information that defines the environment in which a recommendation is provided is often referred to as *context* [1]. Context could be for example the location where the user is listening to a song on his/her mobile phone or the time and weekday of the user-item interaction. Context-aware recommendations (CARs) can significantly improve the recommendation relevance and quality, compared to conventional recommendations that are solely based on user-item interactions [1, 4, 10, 21, 26, 25]. The most successful approaches in context-aware collaborative filtering are based on contextual modeling whereby the user-item-context interactions are modeled jointly in some types of factor models. Two particularly popular classes of factor models for context-aware recommendation are the Tensor Factorization [10, 25] models and the Factorization Machines [21, 19]. Both classes represent the user-item-context interaction as a linear combination of the latent factors to be inferred from the data. Building models without this limitation of linearity to better capture the complex interplay between users'

preferences and their contexts is the main goal of our work in this paper.

Gaussian processes. Gaussian processes (GP) is one of the most widely used family of stochastic processes for modeling dependent data. GP-based models can use flexible covariance functions, which are the same as the class of positive definite kernels used in Support Vector Machines (SVM), they can thus model very complex functions without restricting them to manually chosen parametric (e.g. linear, polynomial) forms. GP have become an important tool for modeling non-linear complex patterns in real-world settings for which human preferences is a prime example. While Gaussian processes have been used in conventional collaborative filtering [5, 7, 15], GPFM is the first GP-based attempt for context-aware recommendations.

The vast majority of CF methods developed as of today using latent factors approaches are based on linear models. In this work we present a powerful non-linear context-aware collaborative filtering method that is based on Gaussian Processes and builds upon past non-linear matrix factorization methods [16] called *Gaussian Processes Factorization Machines (GPFM)*:

- *GPFM* is the first non-linear and non-parametric *context-aware CF method*.
- *GPFM* can seamlessly utilize both implicit and explicit feedback by changing the kernel of the Gaussian Process.
- We derive a stochastic gradient descent optimization procedure which allows *GPFM* to scale linearly to the number of user-item-context, *GPFM* can thus be used on large-scale industry datasets.
- We extensively test *GPFM* on 5 benchmark datasets and compare it to two state-of-the-art context-aware collaborative filtering methods.

2. RELATED WORK

Factor models in Collaborative Filtering been shown to perform well in terms of predictive accuracy and scalability [2, 13, 23, 8, 30]. Restricted Boltzmann Machines (RBM's) [24] have been successfully used in the Netflix prize and could be seen as some form of nonlinear model since the activation functions of the units typically are of nonlinear form (sigmoid, tanh etc.). RBM's models for CF are currently restricted to the user-item problem and have not been extended to context.

Context-aware recommendation (CAR). Early work in CAR utilized contextual information for pre-processing, where the context drives data selection, or post-processing, where the context is used to filter recommendations [1, 4]. More recent work has focused on building models that integrate contextual information with the user-item relations and model the user, item and context interactions directly. Two state-of-the-art approaches have been proposed as of today, one based on Tensor Factorization (Multivers Recommendations) [10, 31] and the other on Factorization Machines (FM) [21]. However, both approaches have been designed exclusively for the rating prediction problem, *i.e.* for explicit feedback. TFMAP a ranking based Tensor Factorization methods has been proposed by [25] for implicit feedback data.

Note that recommendation approaches have been proposed to take into account additional information (also referred as metadata, side information, or attributes) about users or items, *e.g.*, collective matrix factorization [27], localized factor models [3] and graph-based approaches [11]. However, this type of information would go beyond our definition of "context", since we refer to context as information that is associated with both the user and the item at the same time. Finally, note that a recommended item set from a recommender is regarded as the "context" of user choice in the work of [34]. However, this type of context is still extracted from the user-item relations, thus, not in the scope of the context studied in this paper.

Gaussian processes. Gaussian processes have been used to model relational data e.g. in [6] data that contains relational information in the form of an undirected graph is modeled using GP. The model is then applied to classify web-pages, documents and handwritten digits. A GP model for modeling multi-relational data that can include undirected graph or bi-partite graph relationships is built in [32] and tested on Country interactions data and the MovieLens data. Link Analysis models using GP's have been also used on collaborative filtering e.g. [35]. Note that none of the above models is fit for context modeling since they either deal with undirected or bi-partite graph type relationships, while also having scalability constrains. A non-linear method for Matrix Factorization [15] based on GP's was shown to outperform standard Matrix Factorization on the MovieLens data. This method while scalable does not deal with context, and does only deal with explicit feedback data. A Bayesian approach to Tensor decomposition (Tucker decomposition) is introduced in [33]. The method is used in chemometrics and link prediction in social networks.

3. MODEL DESCRIPTION

In this section we shortly introduce the context-aware recommendations problem and explain how to represent user-item-context interactions in the latent feature space. Next we describe the GPFMs and relate it to other models. Finally we derive an extension of GPFMs through modifying the kernel to obtain GPPW, a pairwise preference model, for learning with the implicit feedback.

3.1 Problem Setting

For ease of exposition, we describe the context-aware recommendations problem with a running example in mobile applications (app) recommendation. Let $U = \{Alice, Bob, Charlie, \ldots\}$ be the set of users and $V = \{AB, CCS, D4, \ldots\}$ be the set of items. An event is observed when a user runs an app in the user's current context (location, time etc.). For example, we observe that *Bob* used app *AB* in the morning at work 4 times in total and *Charlie* used app *D4* in the evening at home 11 times in total. Here there are two contextual factors: time of the day, i.e. $C_1 = \{morning, afternoon, evening\}$ and location, i.e. $C_2 = \{home, work, public\}$. Since context is multi-dimensional, we call these dimensions the *contextual factors* to avoid confusion. A specific context is a unique combination of different contextual factors: e.g. two context values in the example are $(morning, work)$ and $(evening, home)$. Let $P = |U|$, $N = |V|$, and $L_m = |C_m|$, where $m = 1, \ldots, M$ and M is the number of contextual factors (i.e. the dimension of con-

text). We reserve the subscripts $i, j, c_1, \ldots, c_M$ for indexing the users, items, and contextual factors, respectively.

We represent each observation as a tuple of (user, item, context, utility). For instance, the two events in the above example correspond to the tuples $(Bob, AB, (morning, work), 4)$ and $(Charline, D4, (evening, home), 11)$. Note that the data in this example is typically considered as implicit feedback. Our problem description, however, applies to both the implicit and explicit feedback settings so we use the term *utility* to enclose both scenarios. Given all observed interactions, the goal of context-aware recommendations is to predict the utility of items in different contexts, which can be used for e.g. to construct an ordered list of items to recommend to the users.

3.2 Utility as a Function of Latent Representations

Since our framework is based on the latent factors approach, in this section we define how to transform an observation into its latent representation. Let the user i, item j, and contextual factor c_m be represented by hidden d-dimensional real-valued feature vectors $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{v}_{c_m}$, respectively. Note that we abuse the notation to avoid using different symbols for the item and contextual factors and instead identifying the two based on their subscripts (j for item, c_m for contextual factor). Conceptually the roles of item and contexts are equivalent so this should not be a problem.

We define a transformation of a pair of (item, context) as:

$$\begin{aligned} t :& V \times C_1 \times \ldots \times C_M \to \mathcal{R}^D \\ t(j, \mathbf{c}) =& [\mathbf{v}_j^T, \mathbf{v}_{c_1}^T, \ldots, \mathbf{v}_{c_M}^T]^T, \mathbf{c} = (c_1, \ldots, c_M) \end{aligned} \tag{1}$$

where $D = (M+1)d$ is the dimension of the latent representation, (d is the dimension for each individual item, context factor vectors) i.e., the latent representation is a stacked column vector of the feature vectors of the item and contextual factors. The mapping between an observation and its transformed representation is one-to-one so we will use them interchangeably henceforth.

Our user-centric approach assumes that, for any given user i, the utility of a pair of item and context $(j, \mathbf{c})$, is a function of the corresponding latent representation, $f_i(t(j, \mathbf{c}))$. The form of the utility function can be freely chosen, and different forms lead to different models as discussed in Section 3.4.3 where we connect different latent factors methods. In this paper we model the utility function using the powerful Gaussian Process (GP) framework which is reviewed in the next section.

3.3 Gaussian Processes (GP)

We briefly review GP, more details can be found in e.g. [18]. A GP is specified by a *mean function* $m(\mathbf{x})$ and a *covariance function* $k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$, where $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{R}^D$. A Gaussian process prior defines a distribution over a real-valued function $f(\mathbf{x})$ if, for any collection $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, the set of function values $\mathbf{f} = \{f(\mathbf{x}_n)\}_{n=1}^N$ has the multivariate Gaussian distribution,

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}; \mathbf{m}, \mathbf{K}),$$

where $\mathbf{m} = \{m(\mathbf{x}_n)\}_{n=1}^N$ and the covariance matrix $\mathbf{K}$ is the values of the covariance function evaluated between all pairs of $\mathbf{x}_n, \mathbf{x}_{n'} \in \mathbf{X}$, i.e. $\mathbf{K}_{nn'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$. We write the GP as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}))$.

An example of a mean function is $m(\mathbf{x}) = 0$, which is a typical assumption in GP models. An example of a covariance function is the popular RBF kernel,

$$k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = s^2 \exp\left[-\frac{1}{2l^2}(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')\right], \tag{2}$$

where $\boldsymbol{\theta} = \{s, l\}$ is called the *covariance hyperparameters*, s is known as the *signal variance* and l the *length-scale*. Using GP as a prior means that, *a priori*, we expect that the function values are *correlated*. The correlation depends on the similarity among the inputs. This makes GP an attractive choice to model the utility function in recommendations since similarity-based models, such as the neighborhood approach, have been shown to be effective for collaborative filtering [12].

3.4 Gaussian Process Factorization Machines

Having introduced GPs, we now describe the Gaussian Process Factorization Machines (GPFMs) for context-aware recommendations. Let $\mathbf{X}_i$ be the matrix of all latent representations of the observations of user i and $\mathbf{y}_i$ be the corresponding observed utilities. Let $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^P$. We use bold capital letters to denote matrices, bold letters for vectors, and regular letters for scalars.

The utility of each user is a function over the latent representations, therefore we will be operating exclusively on the space $\mathcal{R}^D$ of $\mathbf{X}$. We place independent GP priors for the utility function of each user i, i.e. $f_i(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_i))$ where $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{R}^D$ and $\boldsymbol{\theta}_i$ are the covariance hyperparameters unique to user i. Since $\mathbf{X}_i$ is a collection of inputs in $\mathbf{X}$, by the definition of GPs we have:

$$p(\mathbf{f}_i|\mathbf{X}_i, \boldsymbol{\theta}_i) = \mathcal{N}(\mathbf{f}_i; \mathbf{0}, \mathbf{K}^i), \tag{3}$$

where $\mathbf{f}_i = \{f_i(\mathbf{x}_z)\}_{z=1}^{N_i}$ is the range of f_i over $\mathbf{x}_z \in \mathbf{X}_i$, $N_i = |\mathbf{X}_i|$; $\mathbf{K}^i$ is the $N_i \times N_i$ covariance matrix (of user i) with elements $\mathbf{K}^i_{z,z'} = k(\mathbf{x}_z, \mathbf{x}_{z'}; \boldsymbol{\theta}_i)$ for $z, z' = 1 \ldots N_i$. The complete *prior* is thus given by:

$$p(\mathbf{f}_1, \ldots, \mathbf{f}_P|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^P \mathcal{N}(\mathbf{f}_i; \mathbf{0}, \mathbf{K}^i), \tag{4}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^P$.

For any observation $(i, j, \mathbf{c}, y)$, it is unlikely that the utility value $f_i(\mathbf{x} = t(j, \mathbf{c}))$ is exactly the same as the actual utility y. To account for this presence of noise we use the standard iid Gaussian likelihood for an observation:

$$p(y_{iz}|f_i(\mathbf{x}_z), \sigma_i) = \mathcal{N}(y; f_i(\mathbf{x}_z), \sigma_i^2), \tag{5}$$

where σ_i is the noise hyperparameter unique to user i and y_{iz} is the z-th element of the vector $\mathbf{y}_i$.

The complete *likelihood* of all the observations is:

$$\begin{aligned} p(\mathbf{y}_1, \ldots, \mathbf{y}_P|\mathbf{f}_1, \ldots, \mathbf{f}_P, \mathbf{X}, \boldsymbol{\sigma}) &= \prod_{i=1}^P \prod_{z=1}^{N_i} \mathcal{N}(y_{iz}; f_i(\mathbf{x}_z), \sigma_i^2) \\ &= \prod_{i=1}^P \mathcal{N}(\mathbf{y}_i; \mathbf{f}_i, \sigma_i^2 \mathbf{I}), \end{aligned} \tag{6}$$

where $\mathbf{I}$ is the identity matrix and $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^P$.

The GPFM model is specified by the *prior* in Equation 4 and the *likelihood* in Equation 6, and is thus a Bayesian

model. Next we discuss some of the covariance functions that can be used with the GP priors.

3.4.1 Covariance Functions for GPFM

The first covariance function we consider is the RBF function defined in Equation 2. Recall that the transformation $\mathbf{x} = t(j, \mathbf{c})$ is a concatenation of the individual latent vectors $\mathbf{v}_j$ and $\mathbf{v}_{c_m}, m = 1 \dots M$, hence we can rewrite the RBF function (say, for the user i) as:

$$k(\mathbf{x}_z, \mathbf{x}_{z'}; \boldsymbol{\theta}_i) = k(\mathbf{v}_j, \mathbf{v}_{j'}) \prod_{m=1}^{M} k(\mathbf{v}_{c_m}, \mathbf{v}_{c_{m'}}) \tag{7}$$

where

$$k(\mathbf{v}_j, \mathbf{v}_j) = s_i^2 \exp\left[-\frac{1}{l_i^2}(\mathbf{v}_j - \mathbf{v}_{j'})^T(\mathbf{v}_j - \mathbf{v}_{j'})\right]$$

$$k(\mathbf{v}_{c_m}, \mathbf{v}_{c_{m'}}) = \exp\left[-\frac{1}{2l_i^2}(\mathbf{v}_{c_m} - \mathbf{v}_{c_{m'}})^T\mathbf{v}_{c_m} - \mathbf{v}_{c_{m'}})\right],$$

and $\boldsymbol{\theta}_i = \{s_i, l_i\}$. Notice that the kernels $k(\mathbf{v}_j, \mathbf{v}_j)$ and $k(\mathbf{v}_{c_m}, \mathbf{v}_{c_{m'}})$ are also RBF. Hence the RBF kernel of GPFM has a special form: it is the product of the RBF covariances of the item and contextual factors.

Another popular kernel is the linear covariance defined as,

$$k(\mathbf{x}_z, \mathbf{x}_{z'}) = \mathbf{x}_z^T\mathbf{x}_{z'} = \mathbf{v}_j^T\mathbf{v}_{j'} + \sum_{m=1}^{M} \mathbf{v}_{c_m}^T\mathbf{v}_{c_{m'}}, \tag{8}$$

hence the linear kernel in GPFM has the special form of the sum of the linear kernels of the item and contextual factors.

3.4.2 Accounting for Biases in Recommendations

Bias is a well-known phenomenon in recommendations, e.g. some users always give high ratings or some items always receive high scores. To account for this, we allow each user, item, and contextual factor to have a latent bias b_i, b_j, and $b_{c_m}, m = 1 \dots M$, respectively.[1] For each user i, we define a bias function that absorbs its bias in addition to the bias of an observation $(j, \mathbf{c})$:

$$m_i(j, \mathbf{c}) = b_i + b_j + \sum_{m=1}^{M} b_{c_m}, \tag{9}$$

Replacing the standard zero-mean GP prior over the utility functions (Equation 4) with $m_i(\cdot)$ we get the new GPFM prior that accounts for bias:

$$p(\mathbf{f}_1, \dots, \mathbf{f}_P | \mathbf{X}, \mathbf{X}^{bias}, \boldsymbol{\theta}) = \prod_{i=1}^{P} \mathcal{N}(\mathbf{f}_i; \mathbf{m}_i, \mathbf{K}^i), \tag{10}$$

where $\mathbf{X}^{bias} = \{\mathbf{X}_i^{bias}\}_{i=1}^{P}$ with $\mathbf{X}_i^{bias}$ the latent bias corresponding to the observations of user i (similar to $\mathbf{X}_i$) and $\mathbf{m}_i$ is the values of $m_i(\cdot)$ evaluated at $\mathbf{X}_i^{bias}$.

3.4.3 Relation to other Models

The most closely related model to GPFM is the probabilistic matrix factorization (NPMF) [15], which is a non-linear generalization of matrix factorization [28] based on GPs. Indeed, GPFM subsumes NPMF as a special case when there is no context, no bias and only explicit feedback data. Another class of popular factorization models is the Factorization Machines [19]. Using the same notations as in our problem setting, FM defines the utility function of user i given item-context $(j, \mathbf{c})$ as,

$$\begin{aligned} f_i(t(j, \mathbf{c})) =& b_i + b_j + \sum_{m=1}^{M} b_{c_m} + \mathbf{v}_i^T(\mathbf{v}_j + \sum_{m=1}^{M} \mathbf{v}_{c_m}) \\ &+ \mathbf{v}_j^T \sum_{m=1}^{M} \mathbf{v}_{c_m} + \sum_{m=1}^{M-1} \sum_{m'=m+1}^{M} \mathbf{v}_{c_m}^T\mathbf{v}_{c_{m'}}, \end{aligned}$$

where $\mathbf{v}_i \in \mathcal{R}^d$ is the latent vector of user i. The term involving the latent biases is called the unary interaction in FM. This unary interaction is exactly the mean function $m_i(j, \mathbf{c})$ of the GP prior of user i. The remaining pairwise linear combinations of the user, item, and context latent features are known as the 2-way interactions in FM. GPFM replaces this linear interaction with a non-linear function by using a GP prior with covariance function over the latent feature space. Hence GPFM can be seen as the non-linear generalization of FM models of order 2.

Notice that the utility functions in FM are parametric where the latent features $\mathbf{v}_i$ can be seen as the *weights* in a linear regression model. In contrast, the utility functions in GPFM are non-parametric (i.e. having no parametric formula).

3.5 Pairwise Comparison for Implicit Feedback

We now present a variant of GPFM for personalized ranking with implicit feedback. The utility here is not explicitly expressed by users but is in the form of implicit behaviour (e.g. user opening a website or purchasing an item). A typical approach to implicit feedback has been to cast the observed interactions as having positive utility, say 1, and all non-observed interactions as having negative utility, say -1. However, as pointed out in [20], this causes an underestimation problem e.g. for ranking, due to the much larger number of irrelevant (negative) items used for training. In [20], Rendle et al. addressed this problem by optimizing a pairwise comparison (context-agnostic) model. We take a similar approach to build a GPFM-based pairwise preference model which we call GPPW.

3.5.1 Latent Representation of Paired Comparisons

We start by formulating a latent representation of a paired comparison similar to section 3.2. A paired comparison for a given user i is denoted as $(j1, \mathbf{c1}) >_i (j_2, \mathbf{c2})$, which says the user has higher utility for item $j1$ in context $\mathbf{c1}$ than item $j2$ in context $\mathbf{c2}$[2]. This comparison is expressed in the latent space via the transformation,

$$\begin{aligned} t2 &: V \times C \times V \times C \to \mathcal{R}^{2D} \\ t2(jc_1, jc_2) &= [t(jc_1)^T, t(jc_2)^T]^T \end{aligned} \tag{11}$$

where jc is the short notation for $(j, \mathbf{c})$, e.g $jc_1 = (j1, \mathbf{c}1)$.

3.5.2 Pairwise Preference Function as a GP

Following the work in [7, 20] which define the paired comparison as the difference in utility, we define the pairwise preference function of user i, $g_i : \mathcal{R}^{2D} \to R$ as,

$$g_i(\mathbf{x}_1, \mathbf{x}_2) = f_i(\mathbf{x}_1) - f_i(\mathbf{x}_2), \tag{12}$$

[1] Again we abuse the notations to avoid using different symbols.

[2] We can formulate the comparison also as the preference of i for $j1$ over $j2$ *given* a same context, but it is less general.

where $\mathbf{x}_1 = t(jc_1)$, $\mathbf{x}_2 = t(jc_2)$, and $(\mathbf{x}_1, \mathbf{x}_2) = t2(jc1, jc2)$. From the above and $f_i \sim \mathcal{GP}(0, k(\cdot,\cdot))$ we can show that g_i is also a GP with covariance function:

$$\begin{aligned} k_{pref}((\mathbf{x}_1,\mathbf{x}_2),(\mathbf{x}_1',\mathbf{x}_2')) &= Cov[g_i(\mathbf{x}_1,\mathbf{x}_2), g_i(\mathbf{x}_1',\mathbf{x}_2')] \\ &= Cov[f_i(\mathbf{x}_1) - f_i(\mathbf{x}_2), f_i(\mathbf{x}_1') - f_i(\mathbf{x}_2')] \\ &= k(\mathbf{x}_1,\mathbf{x}_1') + k(\mathbf{x}_2,\mathbf{x}_2') \\ &\quad - k(\mathbf{x}_1,\mathbf{x}_2') - k(\mathbf{x}_2,\mathbf{x}_1') \end{aligned}$$

This preference kernel has desirable properties for preference learning, including anti-symmetry and transitivity [7]. Specifically, this means that the functions generated from the kernel satisfy: $g_i(\mathbf{x}_1,\mathbf{x}_2) = -g_i(\mathbf{x}_2,\mathbf{x}_1)$ and $g_i(\mathbf{x}_1,\mathbf{x}_3) > 0$ if $g_i(\mathbf{x}_1,\mathbf{x}_2) > 0$ and $g_i(\mathbf{x}_2,\mathbf{x}_3) > 0$.

3.5.3 The Pairwise Preference Model (GPPW)

Having derived the preference function and its kernel based on the utility function over items, we now describe the pairwise preference model. Since the user utility functions f_i are independent GPs a priori, the pairwise preference functions are also independent GPs. The *prior* of the pairwise preference model is thus

$$p(\mathbf{g}_1,\ldots,\mathbf{g}_P|\mathbf{X}^{pair},\boldsymbol{\theta}) = \prod_{i=1}^{P} \mathcal{N}(\mathbf{g}_i;\mathbf{0},\mathbf{K}^i_{pref}), \tag{13}$$

where $\mathbf{X}^{pair} = \{\mathbf{X}_i^{pair}\}_{i=1}^P$ and each $\mathbf{X}_i^{pair}$ is the latent representations of the paired comparisons of user i; $\mathbf{g}_i = g(\mathbf{X}_i^{pair})$ is value of g_i evaluated at all points in $\mathbf{X}_i^{pair}$; $\mathbf{K}^i_{pref}$ is the covariance matrix of the preference kernel $k_{pref}(\cdot,\cdot)$ evaluated at $\mathbf{X}_i^{pair}$. Note that the GPPW preference kernel is induced from the GPFM utility kernel and thus they share the same set of hyperparameters.

For the *likelihood*, we use standard iid Gaussian noise model leading to

$$p(\mathbf{y}_1^{pair},\ldots,\mathbf{y}_P^{pair}|\mathbf{g}_1,\ldots,\mathbf{g}_P,\mathbf{X},\tilde{\boldsymbol{\sigma}}) = \prod_{i=1}^{P} \mathcal{N}(\mathbf{y}_i^{pair};\mathbf{g}_i,\tilde{\sigma}_i^2\mathbf{I}), \tag{14}$$

where $\mathbf{y}_i^{pair}$ is the set of observed paired comparisons corresponding to $\mathbf{X}_i^{pair}$. Note that the noise hyperparameter $\tilde{\boldsymbol{\sigma}} = \{\sigma_i\}$ is unrelated to that of the utility model.

The analogy between the GPFM and the GPPW can be seen by inspecting their prior and likelihood definitions in the equations 4, 6, 13, and 14. Although sharing the same set of underlying latent utility functions, GPFM models item-based observations whereas GPPW models pair-based observations. Effectively, in terms of learning, GPFM fits a model which aims to score individual items right. GPPW on the other hand fits a model which seeks to order items correctly. As will be seen in the experiments, these different learning goals can lead to substantial difference in performance, for example in learning to rank.

While common in many ranking models our application of the pairwise preference model in this paper is novel with respect to previous applications in the GP literature. In particular, we use GPPW to learn the utility functions whose values can be used to produce an ordered list efficiently. In contrast, traditional paired comparison GP models (e.g. [5, 7]) are used to predict or classify, given two items, which one is preferred by a user. Such comparisons cannot be used to trivially create a ranked list of items for recommendations. Furthermore, these conventional approaches require observed item features and thus do not belong to the class of latent factors model like GPPW.

3.5.4 GPPW for Implicit Feedback

To apply GPPW to the implicit feedback setting, we need to convert the implicit feedback by each user to his/her set of pairwise comparisons. This can be done simply by sampling the negative/irrelevant feedback and creating a pair $jc_1 >_i jc_2$ for every jc_1 in the positive / relevant feedback and every jc_2 from the rest (i.e. the negative/irrelevant feedback). Although this may lead to quadratic number of pairs (per user), our experiments in Section 5.3 suggest that GPPW can be effective using only the same number of observations as GPFM.

4. INFERENCE

In this section we derive inference for GPFM, which includes learning of the hyperparameters and latent features and making predictions for unseen items. Inference for GPPW can be done similarly thanks to the analogy of the two models.

4.1 Learning Latent Features and Covariance Hyperparameters

Although a Bayesian model, fully Bayesian inference of GPFM is not feasible for large-scale data. The standard approach in GP is to use the empirical Bayes (also known as type-II maximum likelihood) approach. This means optimizing the *marginal likelihood* of the model with respect to the latent features and covariance hyperparameters.

The marginal likelihood is obtained by integrating out (hence the term *marginal*) the utility function values $\mathbf{f}_i$, which is given by:

$$p(\mathbf{y}|\mathbf{X},\mathbf{X}^{bias},\boldsymbol{\theta},\boldsymbol{\sigma}) = \int p(\mathbf{y}|\mathbf{f},\boldsymbol{\sigma}) p(\mathbf{f}|\mathbf{X},\mathbf{X}^{bias},\boldsymbol{\theta}) \mathrm{d}\mathbf{f},$$

where $\mathbf{f} = \{\mathbf{f}_i\}_{i=1}^P$ and $\mathbf{y} = \{\mathbf{y}_i\}_{i=1}^P$. Substituting the prior $p(\mathbf{f}|\mathbf{X},\mathbf{X}^{bias},\boldsymbol{\theta})$ (equation 10) and the likelihood $p(\mathbf{y}|\mathbf{f})$ (equation 6) into the above we get,

$$\begin{aligned} p(\mathbf{y}|\mathbf{X},\mathbf{X}^{bias},\boldsymbol{\theta},\boldsymbol{\sigma}) &= \int \prod_{i=1}^{P} \mathcal{N}(\mathbf{y}_i;\mathbf{f}_i,\sigma_i^2\mathbf{I}) \mathcal{N}(\mathbf{f}_i;\mathbf{m}_i,\mathbf{K}^i) \mathrm{d}\mathbf{f}_i \\ &= \prod_{i=1}^{P} \int \mathcal{N}(\mathbf{y}_i;\mathbf{f}_i,\sigma_i^2\mathbf{I}) \mathcal{N}(\mathbf{f}_i;\mathbf{m}_i,\mathbf{K}^i) \mathrm{d}\mathbf{f}_i \\ &= \prod_{i=1}^{P} \mathcal{N}(\mathbf{y}_i;\mathbf{m}_i,\sigma_i^2\mathbf{I}+\mathbf{K}^i). \end{aligned}$$

This gives the *negative log marginal* (with the conditioned variables omitted for brevity):

$$-\log p(\mathbf{y}) = -\sum_{i=1}^{P} \log \mathcal{N}(\mathbf{y}_i;\mathbf{m}_i,\sigma_i^2\mathbf{I}+\mathbf{K}^i), \tag{15}$$

which is the sum of the (negative log) marginals of all users. Each user marginal likelihood is given by:

$$\begin{aligned} -\log \mathcal{N}(\mathbf{y}_i;\mathbf{m}_i,\mathbf{K}^i_y) =& \frac{1}{2}(\mathbf{y}_i-\mathbf{m}_i)^T(\mathbf{K}^i_y)^{-1}(\mathbf{y}_i-\mathbf{m}_i) \\ &+ \frac{1}{2}\log|\mathbf{K}^i_y| + N_i \log 2\pi \end{aligned} \tag{16}$$

where $\mathbf{K}_y^i = \sigma_i^2\mathbf{I} + \mathbf{K}^i$ and N_i is the cardinality of $\mathbf{y}_i$.

4.1.1 Stochastic Gradient Descent (SGD) Learning

With the (negative) log marginal given in equation 15, learning becomes an optimization problem with the optimization variables being the set $\{\mathbf{X}, \mathbf{X}^{bias}, \boldsymbol{\theta}, \boldsymbol{\sigma}\}$. Since the objective $-\log p(\mathbf{y})$ decomposes into the sum of the negative log marginals, we can use *stochastic gradient descent* with respect to users for training with GPFM. In recommendations, the number of observations for a user is relatively small. Thus, this decomposition across users makes GPFM feasible for large-scale datasets as we will see in Section 4.1.3.

We iterate over each user and update its parameters $\{\mathbf{X}_i, \mathbf{X}_i^{bias}, \boldsymbol{\theta}_i, \sigma_i\}$ according to the following update rule:

$$u_{n+1} = u_n + \Delta_n \tag{17}$$
$$\text{where}\Delta_n = h\Delta_{n-1} + \alpha\frac{\mathrm{d}\log\mathcal{N}(\mathbf{y}_i; \mathbf{m}_i, \mathbf{K}_y^i)}{\mathrm{d}u_n},$$

for $u \in \{\mathbf{X}_i, \mathbf{X}_i^{bias}, \boldsymbol{\theta}_i, \sigma_i\}$ at the n^{th} iteration. In the above, α is the learning rate and h is the momentum term which allows a large range of α to be used with SGD.

4.1.2 Derivatives

In this section we find the derivatives of the individual marginal of each user with respect to its parameters. As can be seen from Equation 15, the marginal depends on the form of the covariance function. Here we derive for the RBF covariance function as it generates non-linear functions. To avoid notational clutter, we drop the dependent on subscript i, but the derivation applies to all users.

First, the derivatives of the RBF covariance with respect to the latent features $\mathbf{x}$ are given by:

$$\frac{\partial k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})}{\partial \mathbf{x}} = k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})\frac{\partial(-\frac{1}{2l^2}(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x}'))}{\partial \mathbf{x}}$$
$$= -\frac{1}{l^2}k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})(\mathbf{x}-\mathbf{x}'). \tag{18}$$

Derivatives of $k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$ with respect to the $\boldsymbol{\theta}$ can be similarly computed. Note that both of the latent features $\mathbf{x}$ and the covariance hyperparameters $\boldsymbol{\theta}$ are parameters of the kernel. This is contrary to standard GP where the inputs are observed and thus are constant in the kernel function.

The gradient of the marginal with respect to any $u \in \{\mathbf{x}, \boldsymbol{\theta}\}$ is composed of two parts (see Equation 16):

$$\frac{\partial \log|\mathbf{K}_y|}{\partial u} = \mathrm{tr}\left[\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}}{\partial u}\right] \tag{19}$$

$$\frac{\partial(\mathbf{y}-\mathbf{m})^T\mathbf{K}_y^{-1}(\mathbf{y}-\mathbf{m})}{\partial u} = -(\mathbf{y}-\mathbf{m})^T\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}}{\partial u}\mathbf{K}_y^{-1}(\mathbf{y}-\mathbf{m}), \tag{20}$$

where it should be emphasized again that $\mathbf{y}, \mathbf{m}, \mathbf{K}_y$ are that of user i with the subscript dropped.

The gradient with respect to any of the latent bias u is given by:

$$\frac{\partial(\mathbf{y}-\mathbf{m})^T\mathbf{K}_y^{-1}(\mathbf{y}-\mathbf{m})}{\partial b} = 2\mathbf{K}_y^{-1}(\mathbf{m}-\mathbf{y}). \tag{21}$$

4.1.3 Computational Complexity

To simplify the analysis, we make the crude assumption that each user has the same number of observations i.e. $N_i \approx B/P = a$, where B is the total number of observations. The main cost in computing the marginal and its derivatives (for each user) is the cost of matrix inversion which requires $\mathcal{O}(a^3)$. Once the inverse is calculated, the cost of taking derivatives of all parameters of i is $\mathcal{O}(a^2 \times a(Md + d + 2)$, where the a^2 factor is due to the matrix multiplication and $a(Md + d + 2)$ is the upperbound on the total parameters of the user. The total cost is thus $\mathcal{O}(Pa^3 + Pa^3Md) = \mathcal{O}(Ba^2Md)$. In real world recommendations, the data is very sparse while P is very large, hence a is typically small. The number of contextual factors M is usually less than 20. The latent dimension d can be much smaller than that used in linear latent factors methods (e.g. d=3) since the GP model allows a higher modeling capacity. Hence *the computational complexity of GPFM is linear in the number of total observations* B, as we later demonstrate empirically in Section 5.4. The complexity of GPPW is also linear in the number of paired comparisons as the computation is the same as GPFM, except that the GPPW kernel requires 4 evaluations of the GPFM kernel.

4.2 Predictive Distribution

Once the latent features $\mathbf{X}$ and covariance hyperparameters $\boldsymbol{\theta}$ are learned, they can be used to make prediction for unseen pairs of (item, context). Since GPFM and GPPW use the same underlying set of utility functions, the predictive distribution is the same with respect to the goal of predicting utility for items. Given a test observation $(j_*, \mathbf{c}_*)$, we first use the transformation to convert it to its latent representation $\mathbf{x}_* = t(j_*, \mathbf{c}_*)$. Prediction of the utility for $(j_*, \mathbf{c}_*)$ for user i then follows standard GP regression [18] which is,

$$p(f_i(\mathbf{x}_*)|\mathbf{X}_i, \boldsymbol{\theta}_i, \mathbf{y}_i) = \mathcal{N}(\mu_*, s_*) \tag{22}$$

where

$$\mu_* = k(\mathbf{x}_*, \mathbf{X}_i; \boldsymbol{\theta}_i)(\mathbf{K}_y^i)^{-1}\mathbf{y}_i = ((\mathbf{K}_y^i)^{-1}k(\mathbf{x}_*, \mathbf{X}_i; \boldsymbol{\theta}_i))^T\mathbf{y}_i$$
$$s_* = k(\mathbf{x}_*, \mathbf{x}_*; \boldsymbol{\theta}_i) - k(\mathbf{x}_*, \mathbf{X}; \boldsymbol{\theta}_i)(\mathbf{K}_y^i)^{-1}k(\mathbf{X}, \mathbf{x}_*; \boldsymbol{\theta}_i).$$

Notice that the prediction mean μ_* has the intuitive interpretation of being a weighted linear combination of all seen utilities of user i. The prediction variance s_* expresses the confidence of the model about the (item, context) being predicted. This is an additional advantage of GPFM and GPPW over the non-Bayesian counterpart as, for example, one can use the variance to decide whether or not to recommend an item to the user.

5. EVALUATION

We evaluate the performance of GPFM against two state-of-the-art methods in context-aware recommendations. First we describe the datasets in details. We then evaluate GPFM for explicit feedback and GPFM pairwise for implicit feedback separately. We conclude the section with additional experiments demonstrating the linear scalability of our SGD-based learning algorithm. Our implementation is available at `http://trungngv.github.io/gpfm`.

5.1 Datasets

We use 5 contextual datasets, two of which are in the food domain, two in the movie domain, and one in the mobile applications domain. The statistics of all datasets are given in Table 1 where the first 4 datasets are explicit and the

Table 1: Dataset statistics (#obs is the number of observations and scale is the range of ratings in the datasets).

name	#users	#items	#contexts	#obs	scale
ADOM	84	192	5	1464	1 - 13
COMODA	121	1232	12	2296	1 - 5
FOOD	212	20	2	5554	1 - 5
SUSHI	5000	100	7	50000	0 - 4
FRAPPE	953	4073	4	61465	1

last dataset FRAPPE is implicit. For FRAPPE, the number of observations is the number of user-item-context interactions.

5.1.1 Explicit Feedback Datasets

The first explicit dataset is ADOM [1] which contains 1464 ratings by 84 college students for 192 movies. The students were asked to rate the movies on a scale from 1 (hate) to 13 (absolutely love) and they also filled out information about the context of the watching experience. Following the work in [10, 21] we use 5 contextual factors: companion, day of the week, if it was on the opening week- end, season, and year seen.

The second dataset is COMODA [14] which contains 2296 ratings of 1232 movies by 121 users. It is interesting to note that data acquisition occurred immediately after a user finished watching a movie. As a result, the ratings may be more reliably captured in this dataset compared to others. The user submitted a rating for the movie and also provided the context of the experience: time of the day, day type (working day, weekend, holiday), season, location, weather, social (companion watchers), ending and dominant emotions, mood, physical condition, discovery (self-selected or suggested by others), and interaction (first, n-th).

The third dataset is FOOD [17] which contains 5554 ratings by 212 users on 20 food menus. The users were asked to rate the menus while being in three different levels of hunger. They did so while being either in a real or supposed situation (i.e. the subjects imagined that they were in a specific state of hunger, which may differ from the actual state). We found that the original dataset is contaminated with conflicted ratings where a tuple of (user, item, context)) corresponds to two outputs (ratings) that differ by at least 2. There are 804 such inputs in total and for each of them we replace the conflicted ratings with their average. This results in a clean dataset of 5554 observations from the original 6360 observations.

The fourth dataset is SUSHI [9] which contains 50,000 ratings of 100 different types of sushi by 5000 Japanese users. Each user was asked to score a different set of 10 sushi on a scale from 0 to 4. We use the following contextual factors: style, major group (seafood or otherwise), minor group (12 in total), heaviness/oiliness in state, popularity (how frequently people eat the sushi), price, and availability in shops. Despite the contextual information coincides with the item attributes, we use this dataset to verify the scalability of our model due to the lack of large-scale explicit contextual datasets.

5.1.2 Implicit Feedback Datasets

The first implicit feedback dataset we use is from the mobile app recomender FRAPPE which contains 61,465 implicit feedback of 4073 Android applications by 953 users. The recommender logs the app usage counts by a user in different contexts. The observations with count of at least one is regarded as positive / relevant feedback. The following 4 contextual factors are used: time of the day, day of the week, location (unknown, home, workplace), and weather.

In addition to FRAPPE, we created two more datasets by converting FOOD and COMODA into implicit feedback. This is done by treating all observations with a rating higher than 3 (for FOOD) and 4 (for COMODA) as the positive feedback, i.e. users like the food or movie. This results in 2,882 implicit observations for FOOD and 1526 observations for COMODA. To avoid over-sparsity in the case of COMODA (due to the large number of contextual factors), we use only two, "the ending", "dominant emotions" and "mood factors" in the experiments.

5.2 Evaluation of GPFM for Explicit Feedback

In this section we describe our experimental protocols to evaluate GPFM for explicit feedback and discuss the results.

5.2.1 Experimental Setup

We use the 4 explicit feedback datasets detailed in the previous section. We compare GPFM (*gpfm*) with two state-of-the-art methods in context-aware recommendations namely factorization machines (*fm*[3]) [21] and tensor decomposition (*multiverse*) [10]. Similar to [21], we also compare with standard matrix factorization (*mf*) which does not use any contextual information. Finally, we use a naive predictor (*constant*) which predicts for every user the mean of his ratings.

We split each dataset into 5 folds and repeat the experiments 5 times using one fold as the test set and the remaining 4 folds as the training set. For all methods (except *constant*) we empirically tune the parameters using one of the 5 folds as the validation set. We then fix the tuned parameters when running experiments with the other 4 folds.

For *gpfm* with SGD training, we find that a momentum of 0.9 and a fixed learning rate of 0.5×10^{-4} seems to work well for most datasets[4]. The optimization procedure typically converges after 10 epochs (an epoch is a single pass through the dataset). The latent features are initialized by random sampling from the normal distribution $\mathcal{N}(0, 10^{-4})$, the covariance hyperparameters from the standard normal distribution, and the noise hyperparameter initialized to 1. We find that the RBF kernel always outperform the linear kernel so we use RBF in all of the experiments.

The task we are evaluating is prediction of utilities (ratings) of unseen items and contexts for users. We use four evaluation metrics: mean absolute error (MAE), root-mean-square error (RMSE), Normalized Discounted Cumulative Gain of top 10 items (NDCG@10), and Expected Reciprocal Rank also of top 10 items (ERR@10). Note that while MAE and RMSE measure the overall prediction quality of a model, ERR and NDCG are better suited for ranking as they place higher reward for the top items in a recommended list (typically consists of 5-10 items only). For all datasets, the

[3]We use the libfm implementation available at libfm.org. Note that it only implements a regression model.

[4]We also experiment with decaying the learn rate after every iteration but this seems to have marginal effects.

Table 3: Performance comparison on the 4 *explicit* datasets in terms of ERR@10, higher is better

method	ADOM	COMODA	FOOD	SUSHI
gpfm	0.0809	**0.1923**	**0.1317**	**0.0355**
fm	**0.0931**	0.1905	0.1055	0.0086
multiverse	0.0597	0.0214	0.0736	0.0008
mf	0.0436	0.1371	0.0900	0.0203

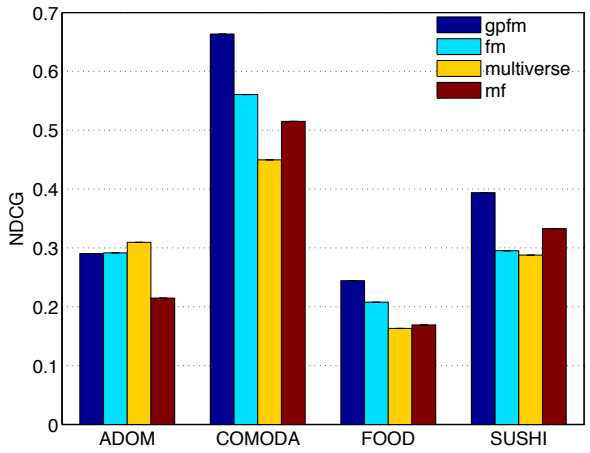

Figure 1: Performance comparison on the 4 *explicit* datasets in terms of NDCG@10, higher is better.

performance is averaged over the 5 different folds. The results are statistically significant and the variances are small so not reported.

5.2.2 Results

The performance comparison of all methods are shown in Table 2 for MAE and RMSE, Table 3 for ERR@10, and Figure 1 for NDCG@10.

First we compare the context-aware and context-agnostic methods. It is clear that both *gpfm* and *fm* significantly outperform *mf* on all datasets and on metrics (with the exception of *mf* slightly outperforms *fm* on the SUSHI dataset. This confirms the benefits of using contextual information in recommendations, as was previously demonstrated in CAR [10, 21]. *Multiverse* outperforms *mf* on the ADOM and FOOD but has poor performance on COMODA and SUSHI where it suffers because of the high dimensionality of context in those two datasets. Meanwhile, *mf* does only slightly better than the naive constant predictor. This can be explained by the fact that users may have different utility for the same item under two different contexts. Failing to account for this user-item-context interaction leads to the low predictive quality of standard matrix factorization.

Next we compare the two best context-aware methods: *gpfm* and *fm*. The overall performance measure in terms of MAE and RMSE (Table 2) shows that *gpfm* does significantly better than *fm* on all of the datasets except ADOM. For example, on COMODA, the MAE and RMSE by *gpfm* are smaller (better) than that of *fm* by 11% and 13%, respectively. Similarly, the MAE and RMSE differences are 6% and 2% for FOOD and 1% and 3% for SUSHI respectively. The superiority of *gpfm* over *fm* is further reflected in terms of ERR in Table 3 and NDCG in Figure 1. This also demonstrates that GPFM gives much better predictions for the items near the top of a ranked recommendation list compared to the state-of-the-art methods. Thus, the experiment results confirm that modeling user-item-context interaction nonlinearly with GPFM leads to substantial performance in context-aware recommendations.

5.2.3 Context that Matters

In context-aware recommendations, different contextual factors are likely to influence the user-item-context interactions at varying degrees. Can we identify which context is most important or relevant to the users? To answer this question, we perform a qualitative analysis with the COMODA dataset. We chose COMODA for two reasons: it has the highest context dimension and also has the most natural, non-intrusive data acquisition process. We run experiments on this dataset using the same experimental setup aforementioned, except that only one of the contexts is used at a time. The results show that the two most influential contextual factors are *ending and dominant emotions* – *gpfm* using those two factors alone gives an MAE of 0.7119 and a RMSE of 0.8953. This turns out to be not too surprising: the ratings of user for movies are typically conditioned on their emotions triggered from watching the movies. Perhaps one implication is that movie recommendation can be improved by taking into account the mood/emotional state of a user, which was the focus of a recent challenge in CAR [22]. While it may not be practical to obtain the users' emotions in the same way as in [14], social medias such as Twitter, Facebook, or online movie review websites may be used to collect similar information.

5.3 Evaluation of GPPW for Implicit Feedback

In this section we describe our experimental protocols for evaluation of GPPW for the implicit feedback datasets and discuss the results.

5.3.1 Experimental Setup

We use the 3 implicit feedback datasets described in Section 5.1.2. Since we are comparing a pairwise preference model (GPPW) with item-based models (GPFM and FM), we must make sure that the comparison is fair.

We use 70% of the relevant feedback in each dataset for training. Let say there are N positive instances, we randomly sample N negative instances, one for each positive item of a user under a given context. The relevant observations are given a rating of 1 and the sampled irrelevant observations are given a rating of -1. These $2N$ observations are then used for training with GPFM and FM, just like in the explicit setting.

The $2N$ observations are also used to create the paired comparisons for GPPW training. Specifically, consider user i in context $\mathbf{c}$, let j^+ be one of the relevant items and j^- its sampled irrelevant counterpart. The items j^+ and j^- are used to create two pairs, $(j^+, \mathbf{c}) >_i (j^-, \mathbf{c})$ and symmetrically $(j^-, \mathbf{c}) <_i (j^+, \mathbf{c})$. *This leads to the exact training size of $2N$ for GPPW.* Note that if the number of relevant items is n for $(i, \mathbf{c})$, the actual number of comparisons is $2n^2$: every j^+ induces $2n$ pairs since j^+ is preferred over all of the n sampled irrelevant items. This can be a potential problem

Table 2: Performance comparison on the 4 *explicit* datasets in terms of MAE and RMSE, smaller is better.

	ADOM		COMODA		FOOD		SUSHI	
method	mae	rmse	mae	rmse	mae	rmse	mae	rmse
gpfm	1.3255	1.9136	**0.7000**	**0.8885**	**0.7358**	**0.9603**	**0.9103**	**1.1640**
fm	**1.2315**	**1.7434**	0.7836	1.0245	0.7609	0.9798	0.9195	1.1994
multiverse	1.4601	2.0412	1.4812	2.1174	0.8336	1.043	0.9544	1.2145
mf	2.2378	2.9836	0.8746	1.1316	0.8845	1.1116	0.9187	1.1970
const	2.2796	2.9206	0.8370	1.0360	0.8993	1.1247	1.0079	1.2498

for training GPPW. However, our experiments suggest that the training size needs only be the same as GPFM and FM for it to be effective.

We carry out the same sampling procedure to create validation sets (10%) and test sets (20%), except that the ratios of irrelevant to relevant are 5, 10, and 20 for FOOD, COMODA, and FRAPPE, respectively. The sampling is done such that the items in training, validation, and test sets are non-overlapped. We use the validation sets to empirically tune the parameters of all methods, afterward the experiments are run 5 times using the tuned parameters.

The task we are evaluating here is prediction of utilities (ratings) for items given a context for the users. We use two ranking evaluation metrics: ERR@10 and Mean Average Precision (MAP@10). For all methods, the predicted utility values are used to create ranked lists which are then scored by the metrics. Note that the ERR and MAP are averaged over user given context, as this was how the data is generated.

5.3.2 Results

The performance of all methods on the 3 implicit feedback datasets is shown in Table 4. It can be seen than GPPW significantly outperforms both GPFM and FM on the FOOD and COMODA dataset while having comparable ERR@10 and MAP@10 on the FRAPPE dataset. Recall that GPPW uses the same set of latent utility functions as GPFM, only optimizing a different likelihood model. The results therefore suggest that learning with paired comparisons can lead to substantial improvement for ranking compared to optimizing item-based scores. Furthermore, as stated in the previous section, GPPW achieves this performance using the same training size as GPFM. This suggests that GPPW can be more effective than GPFM in the implicit feedback setting with little overhead in computation.

5.4 Scalability

Finally we verify the linear computational complexity of the stochastic gradient descent learning for GPFM. To this end, we measure the running time against the amount of data used for training and the dimensionality of the latent features. We normalize the measured time by the time needed to train with 100% of the data and with the latent dimension $d = 8$. The normalized training time per epoch is shown in Figure 2, where the linear correlation between the training size and time can be readily observed. Since the number of optimization variables increases linearly with d, the training time also scales with d, but the overall computational complexity is still linear with respect to the total number of observations. Note that one iteration on the FRAPPE dataset on a MATLAB implementation took approximately one minute.

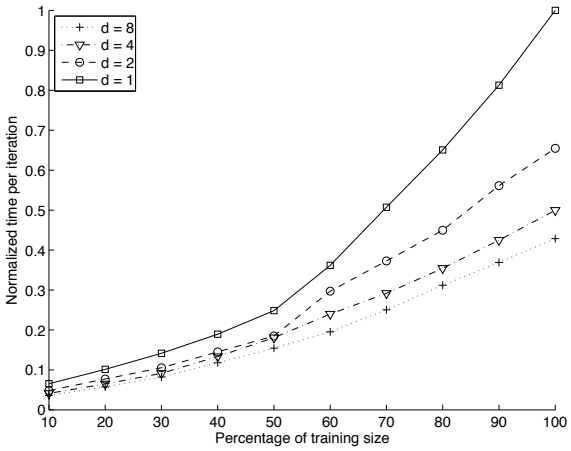

Figure 2: GPFM training time per iteration for the FRAPPE dataset as a function of training size and dimensionality of latent features.

6. CONCLUSIONS

We presented the Gaussian Process Factorization Machines, a novel latent factors based approach for context-aware recommendations. The utility of an item under a context is modeled as functions in the latent feature space of the item and context. By introducing Gaussian processes as priors for these utility functions, GPFM allows complex, nonlinear user-item-context interactions to be captured leading to powerful and flexible modeling capacity. Learning in GPFM is carried out with stochastic gradient descent that scales linearly with the total number of observations and thus making GPFM scalable to large datasets. We also derive a pairwise preference variant of GPFM by changing its covariance function, which can be used seamlessly to deal with implicit feedback.

7. ACKNOWLEDGMENTS

The work leading to these results has received partial funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 610594 (CrowdRec). We would like to thank Edwin V. Bonilla for helpful discussions and the anonymous reviewers for their constructive feedback.

Table 4: Performance comparison on the 3 *implicit* feedback datasets.

	FOOD		FRAPPE		COMODA	
method	ERR@10	MAP@10	ERR@10	MAP@10	ERR@10	MAP@10
gppw	**0.3888**	**0.5555**	0.3814	**0.6067**	**0.3580**	**0.6015**
gpfm	0.3745	0.5089	0.3831	**0.6067**	0.1021	0.0926
fm	0.3659	0.4638	**0.3937**	0.6024	0.1200	0.1224

References

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, New York, NY, USA, 2009. ACM.

[3] D. Agarwal, B.-C. Chen, and B. Long. Localized factor models for multi-context recommendation. In *KDD*, pages 609–617, New York, NY, USA, 2011. ACM.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, pages 245–248, New York, NY, USA, 2009. ACM.

[5] E. Bonilla, S. Guo, and S. Sanner. Gaussian process preference elicitation. In *NIPS*, volume 23, pages 262–270, 2010.

[6] W. Chu, V. Sindhwani, Z. Ghahramani, and S. Keerthi. Relational learning with gaussian processes. In *NIPS*, volume 19, page 289, 2007.

[7] N. Houlsby, J. M. Hernandez-Lobato, F. Huszar, and Z. Ghahramani. Collaborative gaussian processes for preference learning. In *NIPS*, pages 2105–2113, 2012.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[9] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *KDD*, pages 583–588. ACM, 2003.

[10] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, New York, NY, USA, 2010. ACM.

[11] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR*, SIGIR '09, pages 195–202, New York, NY, USA, 2009. ACM.

[12] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, August 2009.

[14] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic. Database for contextual personalization. *Elektrotehniski Vestnik*, 78(5):270–274, 2011.

[15] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *ICML*, pages 601–608. ACM, 2009.

[16] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 601–608, New York, NY, USA, 2009. ACM.

[17] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. In *User Modeling, Adaptation, and Personalization*, pages 102–113. Springer, 2009.

[18] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[21] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644. ACM, 2011.

[22] A. Said, S. Berkovsky, and E. W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 2–6, New York, NY, USA, 2010. ACM.

[23] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, 2008.

[24] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798. ACM, 2007.

[25] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 155–164, New York, NY, USA, 2012. ACM.

[26] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys*, pages 139–146, New York, NY, USA, 2012. ACM.

[27] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, New York, NY, USA, 2008. ACM.

[28] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, pages 1329–1336, 2004.

[29] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal Machine Learning Research*, 10:623–656, June 2009.

[30] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofirank - maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1593–1600, 2007.

[31] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM'10*, pages 211–222, 2010.

[32] Z. Xu, K. Kersting, and V. Tresp. Multi-relational learning with gaussian processes. In *IJCAI*, pages 1309–1314, San Francisco, CA, USA, 2009.

[33] Z. Xu, F. Yan, et al. Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. *arXiv preprint arXiv:1108.6296*, 2011.

[34] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *SIGIR*, SIGIR '11, pages 295–304, New York, NY, USA, 2011. ACM.

[35] K. Yu and W. Chu. Gaussian process models for link analysis and transfer learning. In *NIPS*, pages 1657–1664, 2007.