Collaborative Ordinal Regression

Shipeng Yu SPYU@DBS.IFI.LMU.DE
Institute for Computer Science, University of Munich, Germany

Kai Yu KAI.YU@SIEMENS.COM
Volker Tresp VOLKER.TRESP@SIEMENS.COM
Corporate Technology, Siemens AG, Information and Communications, Munich, Germany

Hans-Peter Kriegel KRIEGEL@DBS.IFI.LMU.DE
Institute for Computer Science, University of Munich, Germany

Abstract

Ordinal regression has become an effective way of learning user preferences, but most research focuses on single regression problems. In this paper we introduce *collaborative ordinal regression*, where multiple ordinal regression tasks are handled simultaneously. Rather than modeling each task individually, we explore the dependency between ranking functions through a hierarchical Bayesian model and assign a common Gaussian Process (GP) prior to all individual functions. Empirical studies show that our collaborative model outperforms the individual counterpart in preference learning applications.

1. Introduction

In recent years there has been quite some work on learning from ranking relations, which, in the literature, is often referred to as ordinal regression (McCullagh, 1980; Cohen et al., 1999; Herbrich et al., 2000; Crammer & Singer, 2002). In this paper we are interested in probabilistic conditional models with a generative process for ranking data. Some recent work in this direction include (Chu & Ghahramani, 2005a) in which Gaussian Processes (GP) are applied to ordinal regression, and (Burges et al., 2005; Chu & Ghahramani, 2005b) where a probabilistic model for pairwise preferences is employed.

Although ordinal regression research has often been motivated from user preference learning using the features of items or products (e.g., movies or books), a unified framework is still missing which can explore the correlation among different ranking functions and reflect the social effects of user preferences. The problem differs from traditional collaborative filtering (see, e.g., Rennie & Srebro, 2005), because the item features should be elegantly explored in the new framework. Web page ranking can also be viewed as an ordinal regression problem, since a ranked list of web pages should be returned for a given query (Burges et al., 2005). In this context it is interesting to explore the correlations across different queries, which can also be viewed as the problem of learning a set of related ranking functions. Instead of treating the ranking functions individually, we would like to model them *jointly* to uncover the dependency between them. We call this problem *collaborative ordinal regression*.

In this paper we propose a Bayesian approach to collaborative ordinal regression. The preference labels for one regression task are assumed to be generated from a latent function, and all the latent functions share a common GP prior to account for the interdependencies. Learning in this model is based on expectation propagation (EP) (Minka, 2001) along with variational method to update the model parameters. Our experimental results demonstrate that collaborative ordinal regression shows better performance than individual regressions in the presence of dependencies between ranking functions.

The rest of paper is organized as follows. In Section 2 we formally introduce collaborative ordinal regression with two example models. Then in Section 3 and 4 we consider learning and inference, respectively. Some empirical results are shown in Section 5, followed by Section 6 with conclusion and some future directions.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

2. Model Formulation

In this paper we use preference learning as a working example to describe our model. We consider a set of n items, and each item has a d-dimensional feature representation $\mathbf{x} \in \mathbb{R}^d$. In ordinal regression we observe a preference label y for each item, which is an integer from 1 (lowest preference) to r (highest preference).

2.1. Ordinal Regression for Single Function

When a single ordinal regression function is considered, the data consist of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where y_i denotes the preference label for item $\mathbf{x}_i$. One natural assumption in ordinal regression is that there is an unobserved latent function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ which maps items into a real line, and the preference labels are then generated from the latent values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$. Put in a probabilistic way, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ be the training items, $\mathbf{y} = [y_1, \ldots, y_n]^\top$ be the labels, and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^\top$ be the vector of latent values, the likelihood of preference labels is written as

$$P(\mathbf{y}|\mathbf{X}, f, \theta) = P(\mathbf{y}|f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n), \theta) = P(\mathbf{y}|\mathbf{f}, \theta),$$

where θ denote some likelihood parameters. We further assume that the labels are mutually independent given the latent values, which leads to a factorized form $P(\mathbf{y}|\mathbf{f}, \theta) = \prod_{i=1}^n P(y_i|f(\mathbf{x}_i), \theta)$.

Since $\mathbf{f}$ are not observable, we need to assign a prior $P(\mathbf{f})$ such that we can integrate them out in a Bayesian framework. In this paper we take a nonparametric approach and assume that $\mathbf{f}$ are a realization of random variables in a Gaussian Process (GP) (Rasmussen & Williams, 2006). The GP can be fully specified by a *mean function* $h(\cdot)$ and a *covariance matrix* $\mathbf{K}$. The (s,t)-th entry of $\mathbf{K}$, i.e., the covariance between the function values $f(\mathbf{x}_s)$ and $f(\mathbf{x}_t)$, is defined by any Mercer kernel function $\kappa(\mathbf{x}_s, \mathbf{x}_t)$. One simple example is the Gaussian kernel $\kappa(\mathbf{x}_s, \mathbf{x}_t) = \exp\left(-\alpha \sum_{l=1}^d (x_s^l - x_t^l)^2\right)$, where $\alpha > 0$ and x_s^l denotes the l-th element of $\mathbf{x}_s$. Then the prior for latent values $\mathbf{f}$ is a multivariate Gaussian,

$$P(\mathbf{f}|\mathbf{h}, \mathbf{K}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{K}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{f}-\mathbf{h})^\top \mathbf{K}^{-1}(\mathbf{f}-\mathbf{h})\right)$$

with $\mathbf{h} = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^\top$. In this paper we use the notation $\mathcal{N}(\mathbf{f}; \mathbf{h}, \mathbf{K})$ to denote a (multivariate) Gaussian distribution for $\mathbf{f}$ with mean $\mathbf{h}$ and covariance $\mathbf{K}$.

The final likelihood of the preference labels is obtained by integrating over the latent function values $\mathbf{f}$,

$$P(\mathbf{y}|\mathbf{X}, \theta, \mathbf{h}, \mathbf{K}) = \int P(\mathbf{y}|\mathbf{f}, \theta) P(\mathbf{f}|\mathbf{h}, \mathbf{K})\, d\mathbf{f}. \quad (1)$$

In general the model has three parameters $\theta, \mathbf{h}$ and $\mathbf{K}$. θ are the parameters for the likelihood model and can assume various forms. We will illustrate two example models in Section 2.3. $\mathbf{h}$ and $\mathbf{K}$ are the parameters for the GP prior and are respectively defined by the mean function h and kernel function κ. In the context of ordinal regression, h defines a *prior preference* for each item, and κ specifies the *smoothness* of the latent function f, i.e., how likely two similar items get the same preference label. In typical GP models, h is assumed to be the zero function, and κ takes a parametric form with some kernel parameters, e.g., α in the Gaussian kernel. Therefore, in model fitting one only needs to optimize likelihood parameters θ and the kernel parameters.

2.2. Collaborative Ordinal Regression

As introduced in the first section, in many real-world problems we need to learn multiple correlated ordinal regression functions for the same set of items. One can of course treat each function separately and apply the model just introduced, but then we lose the *collaborative effects* of these correlated functions, which include (i) *common preferences*: they may share similar preference labels on some items; and (ii) *similar variabilities*: they tend to have the same predictability on very similar items. For example, if items are movies and functions are users who give ratings, the first effect means the users may have common interests on some movies because of, e.g., popularities, famous directors or topics. The second effect is related to the correlation of items in terms of associated user interests, i.e., a user who likes item A tends to like (or dislike) item B. This correlation reflects some intrinsic properties of items, expressed by their feature vectors and user's opinions.

In this subsection we introduce *collaborative ordinal regression* in which the multiple functions are modeled *jointly* in a hierarchical Bayesian framework. For a bit of notations, assume there are m functions, and function j has preference labels $\mathbf{y}_j$ on an item set $\mathbf{X}_j$ of size n_j.[1] Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ be the labels of all functions, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \supset \bigcup_{j=1}^m \mathbf{X}_j$ be the total item set. The item indices of $\mathbf{X}$ that $\mathbf{X}_j$ contains are denoted as $\mathbb{I}_j$. Note that we allow "untouched" items which are not labeled by any function.

In collaborative ordinal regression, we model the preference labels of each function j as an ordinal regression model $P(\mathbf{y}_j|\mathbf{f}_j, \theta_j)$ as in Section 2.1, where $f_j(\cdot)$ is the

[1] With a little abuse of notation, we use bold symbol $\mathbf{y}_j$ to denote the whole label vector for function j, and normal form y_i to denote the label for item i for one function.

latent function for j, and θ_j are the corresponding likelihood parameters. We then connect these models by assigning a *common* GP prior $(\mathbf{h}, \mathbf{K})$ to all the latent functions, following recent works on multi-task learning (Schwaighofer et al., 2005; Yu et al., 2005). Here $\mathbf{h} = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^\top$ are specified by a mean function $h(\cdot)$, and $\mathbf{K}$ denote the $n \times n$ covariance matrix for $\mathbf{X}$. Let $\mathbf{h}_j = \mathbf{h}(\mathbb{I}_j)$ and $\mathbf{K}_{j,j} = \mathbf{K}(\mathbb{I}_j, \mathbb{I}_j)$ be the sub-matrices of $\mathbf{h}$ and $\mathbf{K}$ with respect to $\mathbf{X}_j$, the label likelihood for function j is written as

$$P(\mathbf{y}_j|\mathbf{X}, \theta_j, \mathbf{h}, \mathbf{K}) = \int P(\mathbf{y}_j|\mathbf{f}_j, \theta_j) P(\mathbf{f}_j|\mathbf{h}_j, \mathbf{K}_{j,j})\, d\mathbf{f}_j.$$

By assuming that labels for different functions are independent given the GP prior, we obtain the following likelihood for the whole label set $\mathbf{Y}$:

$$P(\mathbf{Y}|\mathbf{X}, \mathbf{\Theta}, \mathbf{h}, \mathbf{K}) = \prod_{j=1}^{m} P(\mathbf{y}_j|\mathbf{X}, \theta_j, \mathbf{h}, \mathbf{K}), \quad (2)$$

where $\mathbf{\Theta} = \{\theta_1, \ldots, \theta_m\}$ denote the set of all likelihood parameters.

In this framework, the collaborative effects among different functions are modeled via a GP prior $(\mathbf{h}, \mathbf{K})$. In particular, mean function h defines the common preferences of all functions, and covariance matrix $\mathbf{K}$ specifies the smoothness of these functions over all items. Unlike the single function case where we fix $\mathbf{h}$ to zero and take a parametric form for $\mathbf{K}$, we can effectively *learn* the common preferences $\mathbf{h}$ and the (non-stationary) covariance matrix $\mathbf{K}$ from the data. This is described in the next section.

One can assign hyperpriors to $\mathbf{\Theta}$ and to $(\mathbf{h}, \mathbf{K})$. For $\mathbf{\Theta}$ the hyperprior depends on the likelihood model and should be *i.i.d.* for each θ_j. For the GP prior $(\mathbf{h}, \mathbf{K})$, we can assign a conjugate prior which takes a Normal-Inverse-Wishart distribution: $P(\mathbf{h}, \mathbf{K}) = \mathcal{N}(\mathbf{h}; \mathbf{h}_0, \frac{1}{\pi}\mathbf{K})\, \mathcal{IW}(\mathbf{K}; \tau, \mathbf{K}_0)$, where the parameters $\mathbf{h}_0$ and $\mathbf{K}_0$ are respectively the *prior mean* and *base kernel*, and π, τ correspond to the equivalent sample sizes before we observe any data (Schwaighofer et al., 2005; Yu et al., 2005). For the *maximum a posteriori* (MAP) estimate of $\mathbf{h}$ and $\mathbf{K}$, they correspond to a smooth term in the learning process (cf. Section 3).

2.3. Example Models

The proposed framework for ordinal regression is general and connected to many existing models. One can define different likelihood $P(y_i|f(\mathbf{x}_i), \theta)$ to obtain different models, and all such models can be extended in a collaborative manner. In this paper we illustrate two example models in detail, and briefly discuss other likelihood forms in Section 6.

Gaussian Process Regression (GPR): This model grants the ordinal regression problem simply as a GP regression problem. The preference label y_i is assumed to be sampled from a Gaussian with mean $f(\mathbf{x}_i)$ and variance σ^2, i.e., $(\theta \equiv \sigma^2)$

$$P(y_i|f(\mathbf{x}_i), \theta) = \mathcal{N}(y_i; f(\mathbf{x}_i), \sigma^2).$$

The treatment is close to Rankprop (Caruana et al., 1996), which learns ranks by least squares.

Gaussian Process Ordinal Regression (GPOR): This model is discussed in (Chu & Ghahramani, 2005a) and specifies boundary parameters $b_0 < b_1 < \ldots < b_r$ for the r labels. Given the latent function value $f(\mathbf{x}_i)$, the likelihood of a particular label y_i is defined by how likely a corrupted value $z_i \sim \mathcal{N}(z_i; f(\mathbf{x}_i), \sigma^2)$ is in the corresponding interval (b_{y_i-1}, b_{y_i}):

$$\begin{aligned} P(y_i|f(\mathbf{x}_i), \theta) &= \int_{b_{y_i-1}}^{b_{y_i}} \mathcal{N}(z_i; f(\mathbf{x}_i), \sigma^2)\, dz_i \\ &= \Phi\left(z_i^+\right) - \Phi\left(z_i^-\right), \end{aligned}$$

where $z_i^+ = \frac{b_{y_i} - f(\mathbf{x}_i)}{\sigma}$, $z_i^- = \frac{b_{y_i-1} - f(\mathbf{x}_i)}{\sigma}$ and $\Phi(t) = \int_{-\infty}^{t} \mathcal{N}(z; 0, 1)\, dz$. We can define $b_0 = -\infty$ and $b_r = +\infty$ without loss of generality, and the model takes parameters $\theta \equiv \{b_1, \ldots, b_{r-1}, \sigma\}$.

Both of these two models can be extended to collaborative models. In the following we call their collaborative versions as CGPR and CGPOR, respectively.

3. Learning

We derive a general learning scheme based on expectation propagation (EP) (Minka, 2001) for collaborative ordinal regression model, where in principal we can use any likelihood model for $P(y_i|f(\mathbf{x}_i), \theta)$. We apply EP along with a variational method for parameter optimization, which has been applied for, e.g., GP classification (Seeger, 2002; Kim & Ghahramani, 2003). In our setting, EP is applied for each function j and attempts to approximate the *a posteriori* distribution of $\mathbf{f}_j$, i.e. $P(\mathbf{f}_j|\mathbf{y}_j, \theta_j, \mathbf{h}, \mathbf{K})$, as a multivariate Gaussian $Q(\mathbf{f}_j) = \mathcal{N}(\mathbf{f}_j; \hat{\mathbf{f}}_j, \hat{\mathbf{K}}_j)$. Note that here we use $\mathbf{f}_j$ to denote the j-th function values on *all* the n items in $\mathbf{X}$, because the EP algorithm is able to predict means and covariances for *missing data*, which in our case are the unlabeled items for function j. This also means that $\hat{\mathbf{f}}_j$ is a length-n vector, and $\hat{\mathbf{K}}_j$ is a $n \times n$ matrix.

The EP approximation can be done by taking a product form $Q(\mathbf{f}_j) = \prod_{i \in \mathbb{I}_j} t_i(f_j(\mathbf{x}_i)) P(\mathbf{f}_j|\mathbf{h}, \mathbf{K})$ with

Algorithm 1 Collaborative Ordinal Regression

Require: A size-n item set, with features $\mathbf{X} \in \mathbb{R}^{n\times d}$ and preference labels $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ of m functions.
1: Initialize mean vector $\mathbf{h}$, covariance matrix $\mathbf{K}$ and possible hyperparameters $\pi, \tau, \mathbf{h}_0, \mathbf{K}_0$.
2: Initialize likelihood parameters θ_j for function j.
3: **repeat**
4: **for** $j = 1, \ldots, m$ **do**
5: Obtain $Q(\mathbf{f}_j) = \mathcal{N}(\mathbf{f}_j; \hat{\mathbf{f}}_j, \hat{\mathbf{K}}_j)$ by running EP until convergence.
6: Optimize parameter θ_j using (3).
7: **end for**
8: Update GP prior $\mathbf{h}$ and $\mathbf{K}$ using (4).
9: **until** the improvement is smaller than a threshold.

$t_i(f_j(\mathbf{x}_i)) = s_i \exp(-\frac{1}{2}p_i(f_j(\mathbf{x}_i) - m_i)^2)$. We then optimize parameters $\{s_i, m_i, p_i\}$ in $\{t_i\}$ successively by minimizing the Kullback-Leibler divergence,

$$t_i^{\text{new}} = \arg\min_{t_i} \mathrm{KL}\left(\frac{Q(\mathbf{f}_j)}{t_i^{\text{old}}} P(\mathbf{y}_j(i)|f_j(\mathbf{x}_i)) \middle\| \frac{Q(\mathbf{f}_j)}{t_i^{\text{old}}} t_i\right),$$

where $\mathbf{y}_j(i)$ denote the observed label for item i in function j. (Minka, 2001) points out that we can do this approximation by moment matching, and efficient algorithms exist for our setting which do not require any matrix inversion (Rasmussen & Williams, 2006; Herbrich, 2005).

The model parameters $\mathbf{\Theta}$ and $(\mathbf{h}, \mathbf{K})$ can be optimized by variational methods. We apply Jensen's inequality to the likelihood (2) and obtain

$$\log P(\mathbf{Y}|\mathbf{X}, \mathbf{\Theta}, \mathbf{h}, \mathbf{K}) = \sum_{j=1}^{m} \log P(\mathbf{y}_j|\mathbf{X}, \theta_j, \mathbf{h}, \mathbf{K})$$
$$\geq \sum_{j=1}^{m} \int Q(\mathbf{f}_j) \log \frac{P(\mathbf{y}_j|\mathbf{f}_j, \theta_j) P(\mathbf{f}_j|\mathbf{h}, \mathbf{K})}{Q(\mathbf{f}_j)} \, d\mathbf{f}_j.$$

We can then maximize this lower bound with $Q(\mathbf{f}_j)$ fixed as $\mathcal{N}(\mathbf{f}_j; \hat{\mathbf{f}}_j, \hat{\mathbf{K}}_j)$ to get new model parameters. It is seen that $\theta_1, \ldots, \theta_m$ and $(\mathbf{h}, \mathbf{K})$ are not coupled in this optimization problem. For θ_j we need to solve

$$\hat{\theta}_j = \arg\min_{\theta_j} \int Q(\mathbf{f}_j) \log P(\mathbf{y}_j|\mathbf{f}_j, \theta_j) \, d\mathbf{f}_j \qquad (3)$$

analytically or numerically. The updates for $(\mathbf{h}, \mathbf{K})$ can be easily obtained as $\hat{\mathbf{h}} = \frac{1}{\pi+m}\left(\sum_{j=1}^{m} \hat{\mathbf{f}}_j + \pi\mathbf{h}_0\right)$ and

$$\hat{\mathbf{K}} = \frac{1}{\tau+m}\Bigg(\pi(\hat{\mathbf{h}} - \mathbf{h}_0)(\hat{\mathbf{h}} - \mathbf{h}_0)^\top + \tau\mathbf{K}_0 + \sum_{j=1}^{m}\left[(\hat{\mathbf{f}}_j - \hat{\mathbf{h}})(\hat{\mathbf{f}}_j - \hat{\mathbf{h}})^\top + \hat{\mathbf{K}}_j\right]\Bigg), \qquad (4)$$

which can be seen as averaged over the sufficient statistics of all functions and the hyperpriors. Then we perform EP approximations again with the updated parameters, and the whole process is repeated until convergence. Algorithm 1 illustrates the learning algorithm.

The GP prior updates are the *keys* to connect all the individual ordinal regression functions and make them collaborative. On one hand, one can model each task separately and optimize for each task the parameters of a pre-chosen kernel function. But then we lose the collaborative effect among the tasks. On the other hand, one can take a parametric kernel function and optimize the kernel parameters with regards to all the tasks, but then the model can be *restrictive*, because it is highly non-trivial to design a parametric kernel family that is sufficiently expressive. In our solution, the (non-stationary) covariance matrix $\mathbf{K}$ is directly estimated from the repeated random samples (namely, functions in our case), which has sufficient flexibilities. By assigning a prior to $\mathbf{K}$ we can also compromise the degrees of freedom and avoid overfitting.

3.1. Learning in CGPR

In CGPR model the likelihood $P(\mathbf{y}_j|\mathbf{f}_j, \theta_j)$ already takes a Gaussian form, so the EP approximation is exact and turns out to be

$$\hat{\mathbf{f}}_j = \mathbf{K}_{n,j}(\mathbf{K}_{j,j} + \sigma^2\mathbf{I})^{-1}(\mathbf{y}_j - \mathbf{h}_j) + \mathbf{h},$$
$$\hat{\mathbf{K}}_j = \mathbf{K} - \mathbf{K}_{n,j}(\mathbf{K}_{j,j} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{n,j}^\top,$$

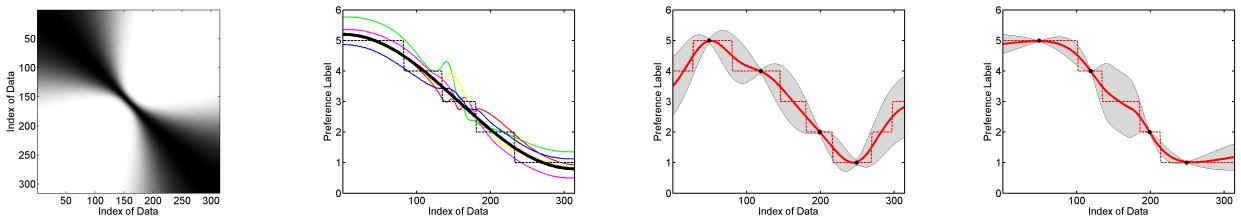
where $\mathbf{K}_{n,j} = \mathbf{K}(:, \mathbb{I}_j)$ denote the $n \times n_j$ rectangle sub-matrix of $\mathbf{K}$, and $\mathbf{I}$ is the identity matrix. The parameter update can also be done analytically as

$$\hat{\sigma}_j^2 = \frac{1}{n_j}\left(\|\mathbf{y}_j - \hat{\mathbf{f}}_j\|^2 + \mathrm{tr}[\hat{\mathbf{K}}_j]\right),$$

where $\mathrm{tr}[\cdot]$ denote matrix trace.

3.2. Learning in CGPOR

Given model parameters, the EP approximation for one function in CGPOR model is very similar to the EP solution in GPOR (Chu & Ghahramani, 2005a), except that we allow unlabeled items to be considered and return an approximated Gaussian $Q(\mathbf{f}_j)$ which is defined on the function values of all items. Then we update the variance and boundary parameters for each task using the same gradient descent methods as in GPOR, and instead of optimizing a kernel parameter we update the GP priors directly using (4). Due to lack of space we do not include the gradient formula in this paper and refer interested readers to (Chu & Ghahramani, 2005a) for details.



(a) (b) (c) (d)

Figure 1. Collaborative ordinal regression (using GPR and CGPR models) on a 1D toy data with 315 data points (0 to π with 0.01 space) and 5 preference labels. (b) shows the mean function h (wide black one in cosine shape), mean preferences (black dashed line) and 5 latent functions sampled from the GP with covariance matrix defined in (a). (c) and (d) show the predicted means and variances (in shade) for a new function learned using GPR and CGPR, respectively. The 4 training data are marked in black. GPR uses Gaussian kernel with $\alpha = 5$, and CGPR uses the kernel in (a).

4. Inference

The proposed model allows us to do two types of inference: label prediction of unlabeled items for each function, and label predictions for new functions.

4.1. Inference for Unlabeled Items

Let us fix a function j and omit the subindex j for simplicity. Given an unlabeled item $\mathbf{x}_*$, we want to infer the preference label y_*. We distinguish two scenarios based on the availability of $\mathbf{x}_*$ before learning.

4.1.1. Transductive Inference

When the item is known before learning, i.e., $\mathbf{x}_* \equiv \mathbf{x}_i \in \mathbf{X}$, we can include it in the learning procedure and obtain the *a posteriori* Gaussian $Q(f_*) = \mathcal{N}(f_*; \mu_*, \sigma_*^2)$ from EP, where $f_* = f(\mathbf{x}_*) = f(\mathbf{x}_i)$, and $\mu_* = \hat{\mathbf{f}}(i)$, $\sigma_*^2 = \hat{\mathbf{K}}(i,i)$ take the entries corresponding to item i from the mean and covariance of $Q(\mathbf{f})$. Then the likelihood of a given label y is

$$P(y|\mathbf{x}_*, \mathcal{D}, \theta, \mathbf{h}, \mathbf{K}) = \int P(y|f_*, \theta) Q(f_*)\, df_*, \qquad (5)$$

where $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ and $P(y|f_*, \theta)$ depends on the likelihood model. This is a one-dimensional integral and can be analytically calculated for CGPR and CGPOR models. Finally the predicted label $y_* = \arg\max_y P(y|\mathbf{x}_*, \mathcal{D}, \theta, \mathbf{h}, \mathbf{K})$.

4.1.2. Inductive Inference

When $\mathbf{x}_*$ is not in $\mathbf{X}$, this is a new item and we cannot directly obtain its *a posteriori* distribution from EP. The only information available is the vector $\mathbf{k}_* = [\kappa(\mathbf{x}_*, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}_*, \mathbf{x}_n)]^\top$, which is defined via base kernel function κ. We can solve this problem by optimizing in the *dual space*: Instead of taking a Gaussian form for $Q(\mathbf{f}_j)$ in EP, we take a Gaussian form for $Q(\boldsymbol{\alpha}_j)$ where $\boldsymbol{\alpha}_j = \mathbf{K}^{-1}\mathbf{f}_j$. Then after EP approximation we obtain $Q(\boldsymbol{\alpha}_j) = \mathcal{N}(\boldsymbol{\alpha}_j; \mathbf{K}^{-1}\hat{\mathbf{f}}_j, \mathbf{K}^{-1}\hat{\mathbf{K}}_j\mathbf{K}^{-1})$ which is equivalent to $Q(\mathbf{f}_j)$. Given the vector $\mathbf{k}_*$ for new item $\mathbf{x}_*$, for function j we have the latent value $f_* = \mathbf{k}_*^\top \boldsymbol{\alpha}_j$, which also takes a Gaussian form $Q(f_*) = \mathcal{N}(f_*; \mu_*, \sigma_*^2)$ with $\mu_* = \mathbf{k}_*^\top \mathbf{K}^{-1}\hat{\mathbf{f}}_j$ and $\sigma_*^2 = \mathbf{k}_*^\top \mathbf{K}^{-1}\hat{\mathbf{K}}_j\mathbf{K}^{-1}\mathbf{k}_*$. Then (5) follows and we are able to predict the label for $\mathbf{x}_*$. For more details on inductive inference for multi-task learning please refer to (Yu et al., 2005).

4.2. Inference for New Functions

In our model all the latent functions are sampled from the GP prior $(\mathbf{h}, \mathbf{K})$. Therefore a new function will by default take $\mathbf{h}$ as the predicted mean preferences, and take $\mathbf{K}$ as the covariances of latent functions corresponding to every two items. By using (5) we can already make default predictions, and after observing labels for some items we need to iteratively run EP and variational updates with fixed GP prior until convergence.

In Figure 1 we show a toy problem for inference with GPR and CGPR models. It is seen from (a) that the covariance is non-stationary such that all the functions sampled from this GP prior have a highly non-smooth part in the middle (see (b)). This corresponds to normal preference learning case where people tend to agree on very good (around 5) and very bad (around 1) items, but have large diversity for items in between. Then when a new function comes with 4 labeled items (see (c)), model fitting simply using GPR will totally ignore this collaborative effect and end up with similar variances for all the test items. CGPR, on the other hand, considers this prior in inference and thus obtains higher variances for test items in the middle (see (d)). The predicted labels (red dashed line) are also more reasonable.

5. Empirical Study

We evaluate the effectiveness of collaborative ordinal regression by comparing the two example models GPR and GPOR with their collaborative counterparts CGPR and CGPOR. In GPR and CGPR, we initialize the model with $\sigma^2 = 0.01$ and estimate the mean preference of each function empirically from the sample mean. For GPOR and CGPOR models, we take the same initializations as suggested in (Chu & Ghahramani, 2005a), i.e., $\sigma = 1$ and $b_k = -1 + \frac{(k-1)r}{2}$ for $k = 1, \ldots, r-1$. These model parameters are optimized using scaled conjugate gradient methods.

We apply these models on two data sets MovieLens and EachMovie, both of which have a large number of users who gave discrete ratings on a set of movies. It is known that there are strong collaborative effects among users, and each movie has its own features such as genre, directors and keywords. Therefore, they are the ideal test beds for collaborative ordinal regression since we can take one movie as one item, and one user as one ordinal regression function.

The MovieLens data we use consists of 100,000 ratings for 1682 movies from 943 users.[2] Ratings are made on a five-star scale, and we remove the movies with less than 50 ratings and have 591 movies left. To build a feature vector for each movie, we use the "genres" part of the movie information which is a 19-dimensional binary-valued vector. A 1 in the vector means the movie takes the corresponding genre.

EachMovie contains 2,811,983 ratings entered by 72,916 users for 1628 movies. The ratings are zero-to-five stars, and we change them to 1-to-6 scores to comply with our notations. To show the model performance on different features, we write a script to download movie information from IMDB[3] including directors, genres, keywords, casts and languages. We then extract all the words and build a 23,753-dimensional *tf-idf* vector for each movie. After ignoring the not-found movies and the movies which are rated less than 50 times, we end up with 1075 movies.

5.1. Evaluation Metrics

In ordinal regression we usually consider the following two evaluation metrics given the target labels R and the predicted labels $\hat{R}$: *mean absolute error* (MAE) which is the average deviation of the prediction from the target, i.e., $\mathrm{MAE}(\hat{R}) = \frac{1}{t}\sum_{i=1}^{t} |\hat{R}(i) - R(i)|$; *mean zero-one error* (MZOE) which gives an error 1 to every incorrect prediction and then averages, as $\mathrm{MZOE}(\hat{R}) = \frac{1}{t}\sum_{i=1}^{t} \mathbf{1}_{\hat{R}(i)\neq R(i)}$. In the collaborative case, we need to average these errors over all the functions to give a single metric, but sometimes this is not straightforward since the number of test items for each function may be very different. Therefore we propose the *macro-averaged* and *micro-averaged* versions of these metrics for collaborative ordinal regression. Macro average is simply the algebraic average of the metrics over all the functions, and micro average is a weighted average with the weights given by the number of test items for each function. We then have four metrics which we denote as MAE-Macro, MAE-Micro, MZOE-Macro and MZOE-Micro, respectively.

Besides evaluating every individual label, we are also interested in the ranking quality of test items. This is sometimes more informative since for a recommendation system the top-ranked items are considered more important than the bottom ones. Here we use the normalized discounted cumulative gain (NDCG) (Jarvelin & Kekalainen, 2000) to evaluate a predicted ranking, which is calculated by summing over all the "gains" along the rank list with a log discount factor as $\mathrm{NDCG}(\hat{R}) = Z\sum_k (2^{r(k)} - 1)/\log(1+k)$, where $r(k)$ denote the target label for the k-th ranked item in $\hat{R}$, and Z is chosen such that a perfect ranking obtains value 1. To focus more on the top-ranked items, we also consider the NDCG@10 which only counts the top 10 items in the rank list. These two scores are averaged over all functions for comparison.

5.2. Performance Comparison

5.2.1. Label Prediction for New Items

For label prediction of new items we fix 100 users with the most number of ratings as the ordinal regression functions. Then for each user, we randomly pick up 10, 20 and 50 labeled items for training and test on the rest of labeled items. The whole process is repeated 10 times independently. We do not explicitly distinguish between transductive inference and inductive inference because we can handle both of them very well. For GPR and GPOR, we train for each user separately and there is no information sharing among users. For CGPR and CGPOR we apply the collaborative learning algorithm and let all the latent functions share the same GP prior. The base kernel for both data sets are linear kernel $\kappa(\mathbf{x}_s, \mathbf{x}_t) = \langle \mathbf{x}_s, \mathbf{x}_t \rangle$, and the mean function is initialized as zero function. Other hyperparameters are initialized as $\pi = 1$ and $\tau = 1$, and they are not sensitive to the performance metrics. For comparison we also show the results of maximum margin matrix factorization (MMMF) which is recently

[2] The data is available at http://www.grouplens.org/.

[3] Internet Movie Database at http://www.imdb.com/.

Table 1. Preference label prediction results for MovieLens (a) and EachMovie (b) with mean and standard deviation. "N" denote the number of training items for each user. We use bold face to indicate the lowest error rate or highest NDCG score for each N. Symbols $\star$ indicate that the collaborative model is significantly better than the individual counterpart (p-value 0.01 in Wilcoxon rank sum test). All the experiments are repeated 10 times independently.

N	Model	MAE-Macro	MAE-Micro	MZOE-Macro	MZOE-Micro	NDCG	NDCG@10
10	GPR	1.1258 ± 0.0235	1.1264 ± 0.0281	0.7082 ± 0.0048	0.7083 ± 0.0054	0.8528 ± 0.0020	0.4937 ± 0.0108
	CGPR	$\star\mathbf{0.8820 \pm 0.0086}$	$\star\mathbf{0.8822 \pm 0.0094}$	$\star 0.6572 \pm 0.0026$	$\star 0.6569 \pm 0.0025$	$\star 0.8579 \pm 0.0016$	$\star 0.5109 \pm 0.0081$
	GPOR	0.9307 ± 0.0290	0.9296 ± 0.0308	0.6545 ± 0.0073	0.6542 ± 0.0074	0.8540 ± 0.0005	0.4988 ± 0.0035
	CGPOR	0.9294 ± 0.0226	0.9283 ± 0.0258	$\mathbf{0.6537 \pm 0.0059}$	$\mathbf{0.6531 \pm 0.0063}$	$\star 0.8553 \pm 0.0008$	$\star 0.5053 \pm 0.0047$
	MMMF	0.9886 ± 0.0224	0.9897 ± 0.0221	0.6853 ± 0.0074	0.6859 ± 0.0073	$\mathbf{0.8717 \pm 0.0026}$	$\mathbf{0.5521 \pm 0.0183}$
20	GPR	1.0813 ± 0.0150	1.0826 ± 0.0131	0.7000 ± 0.0061	0.6993 ± 0.0053	0.8543 ± 0.0015	0.5020 ± 0.0089
	CGPR	$\star\mathbf{0.8222 \pm 0.0063}$	$\star\mathbf{0.8217 \pm 0.0067}$	$\star 0.6379 \pm 0.0040$	$\star 0.6373 \pm 0.0044$	$\star 0.8619 \pm 0.0014$	$\star 0.5249 \pm 0.0073$
	GPOR	0.8901 ± 0.0214	0.8898 ± 0.0212	0.6358 ± 0.0046	0.6350 ± 0.0046	0.8530 ± 0.0008	0.5004 ± 0.0046
	CGPOR	0.8770 ± 0.0177	0.8763 ± 0.0176	$\mathbf{0.6326 \pm 0.0042}$	$\mathbf{0.6314 \pm 0.0044}$	$\star 0.8549 \pm 0.0007$	$\star 0.5089 \pm 0.0044$
	MMMF	0.8989 ± 0.0122	0.9000 ± 0.0121	0.6580 ± 0.0047	0.6589 ± 0.0044	$\mathbf{0.8830 \pm 0.0035}$	$\mathbf{0.6133 \pm 0.0180}$
50	GPR	0.8879 ± 0.0046	0.8880 ± 0.0049	0.6551 ± 0.0037	0.6548 ± 0.0036	0.8546 ± 0.0026	0.5088 ± 0.0141
	CGPR	$\star\mathbf{0.7830 \pm 0.0032}$	$\star\mathbf{0.7822 \pm 0.0035}$	$\star 0.6251 \pm 0.0032$	$\star 0.6245 \pm 0.0032$	$\star 0.8643 \pm 0.0010$	$\star 0.5438 \pm 0.0063$
	GPOR	0.8548 ± 0.0227	0.8522 ± 0.0224	0.6229 ± 0.0075	0.6215 ± 0.0082	0.8500 ± 0.0010	0.5011 ± 0.0051
	CGPOR	0.8385 ± 0.0120	0.8370 ± 0.0104	$\mathbf{0.6162 \pm 0.0053}$	$\mathbf{0.6148 \pm 0.0056}$	0.8514 ± 0.0005	0.5049 ± 0.0035
	MMMF	0.7932 ± 0.0073	0.7928 ± 0.0075	0.6198 ± 0.0050	0.6199 ± 0.0050	$\mathbf{0.8907 \pm 0.0028}$	$\mathbf{0.6651 \pm 0.0190}$

(a) Results for MovieLens

N	Model	MAE-Macro	MAE-Micro	MZOE-Macro	MZOE-Micro	NDCG	NDCG@10
10	GPR	0.9793 ± 0.0092	0.9915 ± 0.0092	0.6843 ± 0.0059	0.6877 ± 0.0060	0.8341 ± 0.0022	0.4558 ± 0.0151
	CGPR	$\star\mathbf{0.9650 \pm 0.0085}$	$\star\mathbf{0.9776 \pm 0.0087}$	$\mathbf{0.6792 \pm 0.0056}$	$\mathbf{0.6828 \pm 0.0057}$	$\star\mathbf{0.8639 \pm 0.0024}$	$\star\mathbf{0.5734 \pm 0.0144}$
	GPOR	1.1186 ± 0.0301	1.1383 ± 0.0316	0.6843 ± 0.0085	0.6863 ± 0.0085	0.8059 ± 0.0003	0.3692 ± 0.0025
	CGPOR	1.1086 ± 0.0299	1.1324 ± 0.0320	0.6820 ± 0.0077	0.6843 ± 0.0079	$\star 0.8083 \pm 0.0011$	0.3789 ± 0.0105
	MMMF	1.2982 ± 0.0127	1.3124 ± 0.0125	0.7506 ± 0.0036	0.7530 ± 0.0036	0.8434 ± 0.0048	0.4746 ± 0.0342
20	GPR	0.9557 ± 0.0073	0.9681 ± 0.0076	0.6798 ± 0.0036	0.6836 ± 0.0042	0.8412 ± 0.0015	0.4849 ± 0.0066
	CGPR	$\star\mathbf{0.9291 \pm 0.0083}$	$\star\mathbf{0.9421 \pm 0.0082}$	$\star 0.6694 \pm 0.0047$	$\star 0.6736 \pm 0.0051$	$\star\mathbf{0.8698 \pm 0.0016}$	$\star\mathbf{0.5989 \pm 0.0118}$
	GPOR	1.0658 ± 0.0180	1.0853 ± 0.0173	0.6682 ± 0.0058	0.6702 ± 0.0057	0.8048 ± 0.0005	0.3678 ± 0.0030
	CGPOR	1.0510 ± 0.0166	1.0721 ± 0.0147	$\mathbf{0.6649 \pm 0.0062}$	$\mathbf{0.6673 \pm 0.0063}$	$\star 0.8078 \pm 0.0013$	$\star 0.3781 \pm 0.0056$
	MMMF	1.1935 ± 0.0120	1.2111 ± 0.0119	0.7257 ± 0.0044	0.7290 ± 0.0042	0.8485 ± 0.0028	0.4786 ± 0.0139
50	GPR	0.9301 ± 0.0044	0.9437 ± 0.0048	0.6718 ± 0.0033	0.6762 ± 0.0036	0.8515 ± 0.0023	0.5375 ± 0.0089
	CGPR	$\star\mathbf{0.8839 \pm 0.0034}$	$\star\mathbf{0.8989 \pm 0.0039}$	$\star 0.6520 \pm 0.0028$	$\star 0.6575 \pm 0.0031$	$\star\mathbf{0.8782 \pm 0.0021}$	$\star\mathbf{0.6341 \pm 0.0114}$
	GPOR	1.0243 ± 0.0190	1.0403 ± 0.0152	0.6549 ± 0.0062	0.6585 ± 0.0052	0.8010 ± 0.0004	0.3663 ± 0.0024
	CGPOR	1.0133 ± 0.0153	1.0327 ± 0.0179	$\mathbf{0.6501 \pm 0.0053}$	$\mathbf{0.6527 \pm 0.0058}$	$\star 0.8045 \pm 0.0006$	$\star 0.3774 \pm 0.0041$
	MMMF	1.0395 ± 0.0072	1.0606 ± 0.0070	0.6866 ± 0.0035	0.6913 ± 0.0034	0.8613 ± 0.0038	0.5478 ± 0.0211

(b) Results for EachMovie

proposed for collaborative filtering (Rennie & Srebro, 2005). MMMF directly optimizes a rank related cost function and is thus very related to collaborative ordinal regression. However, it cannot use the low level features of the data and is purely based on the collaborative effects among users. In MMMF we set the dimensionality to 100 and regularization factor to 10.

In Table 1 we show the results for the two data sets MovieLens and EachMovie. It can be seen that both the collaborative models outperform the corresponding individual models, and the better performance is achieved for all the metrics. This means that the quality of each ordinal regression task can be improved if we consider the collaborative effects and model all the tasks jointly. The difference is especially big for MovieLens data, which probably indicates that MovieLens has stronger collaborative effects than EachMovie.

GPR and GPOR show different behaviors on the two data sets. In MovieLens GPOR is consistently better than GPR, but in EachMovie is worse except for mean zero-one errors. While a theoretical analysis is still missing, we suspect this is related to the feature representation and kernel function. For MovieLens we only use the "genres" for low level features, and thus for GP regression with linear kernel the performance is very poor. GPOR, on the other hand, does a good job because it is modeling ranking more elegantly. When we use the more informative term features for EachMovie, GPR is able to make reasonable predictions, but GPOR does not work very well because of the high dimensionality of features which makes it more likely to overfit. GPOR is learning a pair of boundary values for each label, so when the training data is small the local minima problem occurs.

MMMF does show a good performance on MovieLens, especially for the NDCG scores which are higher than all other methods. This means by directly minimizing a rank related cost, MMMF can obtain very good ranking results. On EachMovie the performance is similar to GPOR, which may also suffer from overfitting.

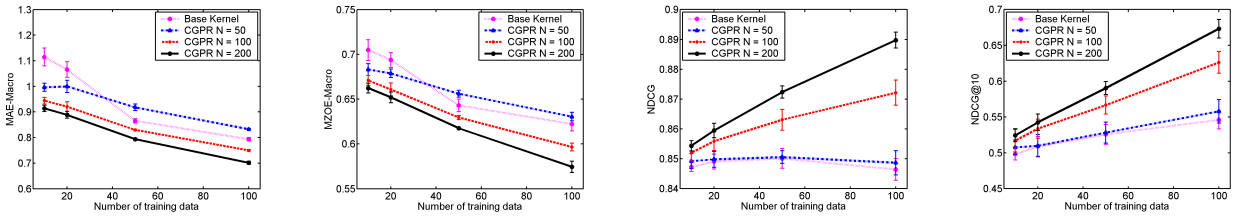


Figure 2. Learning curves of GPR for label predictions of new users on MovieLens. We show the means and standard deviations of MAE-Macro, MZOE-Macro, NDCG, and NDCG@10 over 20 trials from left to right. We vary the number of training items for each test user as 10, 20, 50 and 100. The pink dotted lines are training with base linear kernel (without collaborative effects), and other lines use the learned GP priors trained in CGPR models with the corresponding N users.

5.2.2. Label Prediction for New Functions

As discussed in Section 4.2, inference for new functions in collaborative ordinal regression simply means we fix the learned GP prior for the new latent functions. So our goal is to investigate whether or not we can improve label predictions for new users using the new prior. In this experiment we use GPR as the baseline, and train different CGPR models to get different GP priors for testing. To show the collaborative effectiveness on new users, we vary the number of training users for CGPR models from 50 to 200, and test on the rest users. For training in CGPR models we fix the number of labeled items to be 100. In Figure 2 we show the learning curves of 4 metrics on MovieLens data set and omit others since the results are similar. It can be seen that by using the learned priors, prediction performance of new users can be greatly improved. With more training users in CGPR model training, better performance can be achieved. This indicates a stronger collaborative effects for more users.

6. Conclusion and Future Work

We have proposed a Bayesian framework for collaborative ordinal regression and empirically evaluated two models. (C)GPOR is slow due to hard parameter optimization, so it is interesting to look for better likelihood models in future. To improve scalability, one can assume a linear form for each f_j, i.e., $f_j(\mathbf{x}_i) = \mathbf{w}_j^\top \mathbf{x}_i$, and learn the Gaussian prior jointly for all $\mathbf{w}_j$'s. Extending this framework to collaborative pairwise preference learning is also worth investigating.

References

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *ICML'05* (pp. 89–96).

Caruana, R., Baluja, S., & Mitchell, T. (1996). Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. *NIPS'8*.

Chu, W., & Ghahramani, Z. (2005a). Gaussian processes for ordinal regression. *JMLR*, *6*, 1019–1041.

Chu, W., & Ghahramani, Z. (2005b). Preference learning with Gaussian processes. *ICML'05*.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Crammer, K., & Singer, Y. (2002). Pranking with ranking. *NIPS'14*.

Herbrich, R. (2005). On Gaussian expectation propagation. http://research.microsoft.com/∼rherb/papers/EP.pdf.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (pp. 115–132). MIT Press.

Jarvelin, K., & Kekalainen, J. (2000). IR evaluation methods for retrieving highly relevant documents. *SIGIR'00*.

Kim, H., & Ghahramani, Z. (2003). The EM-EP algorithm for Gaussian process classification. *Proceedings of the Workshop on Probabilistic Graphical Models for Classification at ECML*.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society B*, *42*, 109–142.

Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. Doctoral dissertation, Massachusetts Institute of Technology.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. *ICML'05*.

Schwaighofer, A., Tresp, V., & Yu, K. (2005). Hierarchical bayesian modelling with gaussian processes. *NIPS'17*.

Seeger, M. (2002). *Notes on Minka's expectation propagation for Gaussian process classification* (Technical Report). University of Edinburgh.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. *ICML'05* (pp. 1017–1024).