Collaborative Ordinal Regression

Shipeng Yu[1,2]**, Kai Yu**[2]**, Volker Tresp**[2]
[1]Institute for Computer Science, University of Munich, Munich 80538, Germany
[2]Siemens Corporate Technology, Munich 81739, Germany
spyu@dbs.ifi.lmu.de
kai.yu@siemens.com, volker.tresp@siemens.com

Abstract

Ordinal regression has become an effective way of learning user preferences, but most of research only focuses on single regression problem. In this paper we introduce *collaborative ordinal regression*, where multiple ordinal regression tasks need to be handled simultaneously. Rather than modelling each task individually, we build a hierarchical Bayesian model and assign a common Gaussian Process (GP) prior to all individual latent functions. This very general model allows us to formally model the inter-dependencies between regression functions. We derive a very general learning scheme for this type of models, and in particular we evaluate two example models with collaborative effect. Empirical studies show that collaborative model outperforms the individual counterpart.

1 Introduction

Recent years have seen many works on preference learning, which aims to learn a preference model for the user and make predictions for newly arriving items. This is in general called ordinal regression in the literature [5]. In this paper we are interested in probabilistic conditional models for rankings (preference labels), for which we need to define some generative process for ranking data. Some recent work in this direction include [3] in which Chu and Ghahramani applied Gaussian Processes (GP) to ordinal regression, and [2] where Burges et al. derived a generative model for pairs of objects and trained the model using neural networks.

Up to now most of the research in ordinal regression focuses on a single regression problem, but in reality many ranking applications have multiple regression tasks. For instance, in user preference prediction, one user can be modelled as an ordinal regression problem, but in many cases we need to predict user preferences for many users at the same time. Intuitively, we should not model each user individually, but model all the users *jointly* to uncover the dependency between them. We call this problem *collaborative ordinal regression*, since it is more general than the well-known collaborative filtering problem. Another example of multiple regression is in web page ranking, where each query can be taken as an ordinal regression function (with possibly more than 2 labels) on all the web pages. The problem here is how to rank the web pages for a new query.

In this paper we propose a very general Bayesian framework for 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 inter-dependencies. Learning in this model is via an EM-like algorithm, where the E-step is an Expectation-Propagation (EP) [6] iteration in the general case. The model is shown to cover many existing models in the literature, and we empirically compare two of them in a collaborative manner. The collaborative models are shown to give better performance than the individual models.

The rest of the paper is organized as follows. In Section 2 we formally introduce the collaborative framework and give some example models. Then in Section 3 we derive a learning algorithm for collaborative ordinal regression. Some empirical results are shown in Section 4, followed by Section 5 with some discussions and future directions.

2 Model Formulation

In this section we use the language of collaborative filtering to explain the model. We assume we have an item set of m items, with the ith item denoted as $\mathbf{x}_i \in \mathbb{R}^d$. Each user gives a preference label y to each item, which is an integer between 1 (lowest preference) and r (highest preference).

2.1 Ordinal Regression for Single User

In the single user case, the data consist of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where y_i gives the preference label for item $\mathbf{x}_i$. One natural assumption 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 ranking outputs are then generated from the latent values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)$. More formally, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^\top$, $\mathbf{y} = [y_1, \ldots, y_m]$ and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)]$, the likelihood of preference labels is written as

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

where θ is some model parameter.

To complete the Bayesian formulation, we assign a Gaussian prior to the function f, $f \sim \mathcal{N}(f; h, \mathbf{\Sigma})$, which leads to a GP model for preference learning (see [3] for more explanations). h is the *prior function* for f, and $\mathbf{\Sigma}$ is the *kernel matrix* which is generated from a *covariance function* $\kappa(\cdot, \cdot)$ (also known as *kernel function*). The (i, k)-th entry in the kernel $\mathbf{\Sigma}$ is calculated as $\kappa(\mathbf{x}_i, \mathbf{x}_k)$, which measures the covariance between the functions corresponding to inputs $\mathbf{x}_i$ and $\mathbf{x}_k$. The final conditional likelihood is written as

$$P(\mathbf{y}|\mathbf{X}, \theta, \phi) = \int P(\mathbf{y}|\mathbf{X}, f, \theta) P(f|\phi)\, df,$$

where $\phi \equiv \{h, \mathbf{\Sigma}\}$ denote all the parameters for prior of f.

2.2 Collaborative Ordinal Regression

In the multiple user case, we have n users, and user j has preference labels $\mathbf{y}_j$. Following the notation in previous subsection, we can model each user j as an ordinal regression model $P(\mathbf{y}_j|\mathbf{f}_j, \theta_j)$. However if we model them separately, label prediction for one user will only depend on preference labels of this particular user, which means we will lose the *collaborative effect* between users. To cope with this user dependency, we follow recent works for multi-task learning [7, 8] and assign a *common* Gaussian Process prior ϕ to all the functions f_j's in the GP model. In this nonparametric hierarchical Bayesian framework, different users are constrained to have similar preference patterns on the items. In particular, the prior function h defines the mean preference of these users, and kernel matrix $\mathbf{\Sigma}$ constrains the smoothness of these functions over all items. To better reflect the "common

interest" between users, both h and $\mathbf{\Sigma}$ can be adapted to the training data. The conditional likelihood for the whole observations $\mathcal{D} \equiv \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ is then written as

$$P(\mathcal{D}|\mathbf{X}, \mathbf{\Theta}, \phi) = \prod_{j=1}^{n} P(\mathbf{y}_j|\mathbf{X}, \theta_j, \phi) = \prod_{j=1}^{n} \int P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)\, df_j. \quad (1)$$

To be more Bayesian, one can assign priors for θ_j's and for ϕ, which results in a more flexible yet more complicated model. In particular, we can assign a Normal-Inverse-Wishart prior for ϕ, as done in [7, 8]. For MAP estimate of ϕ, this corresponds to a smooth term in the learning process. In this paper we do not consider this prior for simplicity.

2.3 Example Models

In the literature, different ranking models differ in defining the likelihood term $P(\mathbf{y}|\mathbf{f}, \theta)$. In this paper we discuss and compare two models, in which the likelihood term can be factorized with respect to items, i.e., $P(\mathbf{y}|\mathbf{f}, \theta) = \prod_{i=1}^{m} P(y_i|f(\mathbf{x}_i), \theta)$. Other forms of likelihood is discussed briefly in Section 5.

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

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

Gaussian Process Ordinal Regression (GPOR) This model is discussed in [3] and assumes there are some "pins" $b_0, b_1, \ldots, b_r$ on the real line. Each label y_i is assigned according to which area a disturbed value $z_i \sim \mathcal{N}(z_i; f(\mathbf{x}_i), \sigma^2)$ is in:

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

We can define $b_0 = -\infty$, $b_r = +\infty$, and the model parameter $\theta \equiv \{b_1, \ldots, b_{r-1}, \sigma\}$.

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

3 Learning

In this section we derive a very general framework for learning in collaborative ordinal regression model, where in principal we could use any likelihood function for $P(\mathbf{y}|\mathbf{f}, \theta)$. Since for collaborative filtering it is very likely to have missing data in the $\mathbf{y}$'s, we also consider this situation in the learning procedure. Starting from the likelihood function (1), we apply Jensen's inequality and obtain

$$\begin{aligned} \log P(\mathcal{D}|\mathbf{X}, \mathbf{\Theta}, \phi) &= \sum_{j=1}^{n} \log \int P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)\, df_j \\ &\geq \sum_{j=1}^{n} \int Q(f_j) \log \frac{P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)}{Q(f_j)}\, df_j, \end{aligned} \quad (2)$$

where $Q(f_j)$ is some function of f_j. Therefore an EM-like learning procedure can be derived by iteratively maximizing this lower bound (2) with respect to $Q(f_j)$ and model parameters $\mathbf{\Theta}$ and ϕ. In the E-step, we approximate $Q(f_j)$ to the *a posteriori* distribution of f_j, and we take a Gaussian form for Q, i.e., $Q(f_j) = \mathcal{N}(f_j; \hat{f}_j, \hat{\mathbf{\Sigma}}_j)$. While directly

maximizing the lower bound with respect to Q is difficult and may be intractable[1], we can apply Expectation-Propagation (EP) [6] to do this approximation, since $P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j)$ takes a factorized form for each j (see Appendix for details). This corresponds to Assumed Density Filtering (ADF) as an approximation to the posterior. In the M-step, new model parameters are obtained by directly maximizing the lower bound (2) with $Q(f_j)$ known from E-step. It is seen that θ_j's and ϕ are not coupled in this optimization problem. For θ_j we need to solve $\hat{\theta}_j = \arg\min_{\theta_j} \int Q(f_j) \log P(\mathbf{y}_j|\mathbf{X}, f_j, \theta)\, df_j$ analytically or numerically. The update for ϕ can be easily obtained as

$$\hat{h} = \frac{1}{n}\sum_j \hat{f}_j, \qquad \hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_j \left[(\hat{f}_j - \hat{h})(\hat{f}_j - \hat{h})^\top + \hat{\mathbf{\Sigma}}_j\right], \tag{3}$$

which can be intuitively understood as an average over all the functions. We emphasize that this update is *independent* of the likelihood function form, and that different users are connected and influenced *only* via this update. Note here that we can learn both the mean function h and the (non-stationary) kernel matrix $\mathbf{\Sigma}$, because we have multiple samples of f_j's from the GP prior (see [8] for a detailed discussion). This is in contrast to model fitting in [3] where only a parameter for a stationary kernel is updated.

3.1 Learning in CGPR

In CGPR model, since the likelihood term $P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j)$ already takes a Gaussian form, the approximation in the E-step is exact. Denote m_j the length of $\mathbf{y}_j$, the E-step turns out to be

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

where $\mathbf{h}_j$ is the length-m_j sub-vector of $\mathbf{h}$, and $\mathbf{\Sigma}_{m,j}$ and $\mathbf{\Sigma}_{j,j}$ denote the $m \times m_j$ and $m_j \times m_j$ sub-matrix of $\mathbf{\Sigma}$, respectively. The M-step for θ is given analytically as $\hat{\sigma}^2 = \frac{1}{\sum_j n_j}\sum_j \left(\|\mathbf{y}_j - \hat{f}_j\|^2 + \mathrm{tr}[\hat{\mathbf{\Sigma}}_j]\right)$. The whole algorithm turns out to be the same as in [7] and [8], if we assign a Normal-Inverse-Wishart hyperprior to ϕ and consider MAP estimates for ϕ.

3.2 Learning in CGPOR

The learning algorithm for CGPOR model is a natural extension of the EP algorithm given in [3]. In the E-step, we fit a Gaussian $Q(f_j)$ for the *a posteriori* distribution of each function f_j. Since there may be items without labels, we only need to consider the likelihood terms for labelled items. But after EP learning, the obtained Gaussian $Q(f_j)$ is defined not on labelled items, but on *all* the items (see Appendix). These Gaussian parameters will be used to update prior ϕ in the M-step as given in (3), and we update "pins" for each user separately via the same gradient method as in [3].

4 Empirical Study

We report some preliminary results on the EachMovie data set, which is popular for collaborative filtering research. For preprocessing we remove movies that are rated less than 50 times, and users who give less than 20 ratings. Then we end up with 809 movies, 4842 users and 334251 ratings. Since we don't have a feature representation for each movie, we select the 100 users with the most ratings as the true "users" in our model (i.e., regression functions), and treat the other users as features for each movie. A correlation kernel [1] is

[1] This is known as variational EM in the literature.

# Training Items	GPR	CGPR	GPOR	CGPOR
100	1.1971	1.1763	1.2366	1.2225
200	1.1283	1.1260	1.2126	1.1711
300	1.1042	1.1005	1.0907	1.0557
400	1.0970	1.0923	1.0454	1.0203

Table 1: Mean absolute errors for all the four models.

# Training Items	GPR	CGPR	GPOR	CGPOR
100	0.7432	0.7372	0.6239	0.6168
200	0.7224	0.7213	0.6087	0.5999
300	0.7155	0.7142	0.5862	0.5782
400	0.7125	0.7085	0.5760	0.5723

Table 2: Mean zero-one errors for all the four models.

defined for movies and used as the prior $\mathbf{\Sigma}$, in which we subtract the user mean and calculate cosine distance between two movie vectors. All the missing values for features are filled in as the corresponding user means. For the target values, about 60% of the entries are missing.

For learning the 100 functions (for the 100 users), we randomly select a subset of $\{100, 200, 300, 400\}$ movies for training, and prediction performance over the rest movies is averaged over all the users. Two comparison metrics are used: *mean absolute error* is the average deviation of the prediction from the target; *mean zero-one error* gives an error 1 to every incorrect prediction and then averages over all predictions. Prior function h is set to zero function initially. σ is initialized as 0.1 for GPR models and 1 for GPOR models. Table 1&2 give the performance for each approach. It can be seen that both the collaborative models outperform the corresponding individual models (in which we learn a GP model for each user separately). This means ranking can be improved if we model all the tasks jointly. If the prior kernel $\mathbf{\Sigma}$ is worse (e.g., calculated with less users), the differences will be larger. GPOR models are in general better than GPR models, except for mean absolute error with small training sets. This may be because of poor model fitting due to lack of data.

5 Discussion

This paper goes beyond ordinal regression for a single function and introduces a very general framework for modelling collaborative ordinal regression. A generative process of all the ordinal entries is presented, and learning can be done via a mixed EM and EP algorithm. Single-output ordinal regression is a special case of the proposed model.

The proposed EP algorithm allows any factorized form for the likelihood term $P(\mathbf{y}|\mathbf{X}, f, \theta)$. Recently Burges et al. proposed a neural network learner called RankNet [2], in which a logistic function is used to map pairwise outputs to probabilities. If we cast RankNet model into a Bayesian framework, it actually factorizes likelihood $P(\mathbf{y}|\mathbf{f}, \theta)$ with respect to pairs, i.e., $P(\mathbf{y}|\mathbf{f}, \theta) = \prod_{y_i \rhd y_k} P(y_i \rhd y_k | f(\mathbf{x}_i), f(\mathbf{x}_k), \theta) = \prod_{i,k} P_{ik}(o_{ik}, \theta)$.[2] Here $y_i \rhd y_k$ means "item i is preferred than item k", and $o_{ik} \equiv f(\mathbf{x}_i) - f(\mathbf{x}_k)$. P_{ik} is simply the logistic function $P_{ik}(o_{ik}, \theta) = \frac{\exp(o_{ik})}{1+\exp(o_{ik})}$. Note that we have an empty set for θ. This model is also ready to generalize to collaborative case, and the only difficulty for learning is that in E-step we need to calculate the first and second moments of the distri-

[2]This correspondence can be seen if we think $\max_\theta \mathcal{L}(\theta) = \min_\theta(-\log \mathcal{L}(\theta))$.

bution $z \sim \frac{\exp(z)}{1+\exp(z)}\mathcal{N}(z; \mu_z, \sigma_z^2)$. This involves one-dimensional integral and can be done via sampling method. One may argue that this is not a proper generative model since all the pairs should not be independent, but the experimental results in [2] somehow validate the model (with a neural network learner). An interesting future work is to investigate how this model behaves in a collaborative manner.

This paper focuses on a *transductive setting* of rank prediction, i.e., all the test items are known *a priori*. We just take their targets as missing data and come up with predicted means and variances after learning. For induction on previously unseen items, we need to map the learned kernel back using a similar approach as in [8]. It's interesting to empirically evaluate this approach.

Appendix: EP Learning

We describe EP learning for E-step. In the following we ignore the subindex j since we need to do EP for all the functions f_j's. To approximate the *a posteriori* distribution of f as a Gaussian $Q(f)$, in EP learning we take a factorized form for the likelihood term, i.e., $P(\mathbf{y}|\mathbf{X}, f, \theta) = \prod_k t_k(f)$, and approximate each term $t_k(f)$ as a Gaussian sequentially [6]. Gaussian EP has been carefully investigated recently in [4], and it turns out that the critical terms for calculating EP approximation are $Z_k(\mathbf{u}, \mathbf{C}) \equiv \int t_k(f)\mathcal{N}(f; \mathbf{u}, \mathbf{C})\, df$ and its gradients $\mathbf{g}_k(\mathbf{u}, \mathbf{C}) \equiv \partial \log Z_k(\mathbf{u}, \mathbf{C})/\partial \mathbf{u}$ and $\mathbf{G}_k(\mathbf{u}, \mathbf{C}) \equiv \partial \log Z_k(\mathbf{u}, \mathbf{C})/\partial \mathbf{C}$. A careful manipulation shows that

$$\mathbf{g}_k(\mathbf{u}, \mathbf{C}) = \mathbf{C}^{-1}(\langle f \rangle - \mathbf{u}), \mathbf{G}_k(\mathbf{u}, \mathbf{C}) = \frac{1}{2}\mathbf{C}^{-1}\left(\langle ff^\top \rangle - \mathbf{u}\langle f \rangle^\top - \langle f \rangle \mathbf{u}^\top + \mathbf{u}\mathbf{u}^\top - \mathbf{C}\right)\mathbf{C}^{-1},$$

where $\langle \cdot \rangle$ denotes the expectation under distribution $f \sim t_k(f)\mathcal{N}(f; \mathbf{u}, \mathbf{C})/Z_k(\mathbf{u}, \mathbf{C})$. Calculation of these moments may be problematic for arbitrary term $t_k(f)$, but in this paper all these terms are one-dimensional, therefore we can at least use numerical integration method (e.g. Simpson quadrature) to calculate them. In some special cases, however, we can solve these integrals analytically. For instance in CGPR model, $t_k(f)$ are already Gaussian, so all the moments can be very easily calculated and the approximation is exact. For CGPOR model, the moments can be obtained through special properties of the error function (see [3] or [4]).

References

[1] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML*, pages 65–72, 2004.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[3] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

[4] R. Herbrich. On Gaussian expectation propagation. 2005.

[5] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42(2):109–142, 1980.

[6] T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[7] A. Schwaighofer, V. Tresp, and K. Yu. Hierarchical bayesian modelling with gaussian processes. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[8] K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *ICML*, pages 1017–1024, 2005.