Cross-domain Ranking via Latent Space Learning

Jie Tang
Department of Computer Science and Technology, Tsinghua University
]Tsinghua National Laboratory for Information Science and Technology (TNList)
jietang@tsinghua.edu.cn

Abstract

We study the problem of cross-domain ranking, which addresses learning to rank objects from multiple interrelated domains. In many applications, we may have multiple interrelated domains, some of them with a large amount of training data and others with very little. We often wish to utilize the training data from all these related domains to help improve ranking performance. In this paper, we present a unified model: BayCDR for cross-domain ranking. BayCDR uses a latent space to measure the correlation between different domains, and learns the ranking functions from the interrelated domains via the latent space by a Bayesian model, where each ranking function is based on a weighted average model. An efficient learning algorithm based on variational inference and a generalization bound has been developed. To scale up to handle real large data, we also present a learning algorithm under the Map-Reduce programming model. Finally, we demonstrate the effectiveness and efficiency of BayCDR on large datasets.

Introduction

Cross-domain ranking addresses a common situation that arises when applying machine learning to many emerging applications over interrelated domains on the Web. In such applications, it may be possible to collect ample training data (e.g., user-rated reviews) for some domains, but at the same time little or no data for the other domains. This leads to a situation where some ranking functions perform well for some domains, but poorly for the other domains. Therefore, it becomes highly desirable to find a way to pool together the training data across the interrelated domains, in order to improve the ranking accuracy over all domains.

While much related work has been conducted—for example, transfer learning (Bickel, Brückner, and Scheffer 2007; Dai et al. 2007), domain adaptation (Ben-David et al. 2006), multi-task learning (Argyriou, Evgeniou, and Pontil 2006; Crammer, Kearns, and Wortman 2006), learning to rank (Herbrich, Graepel, and Obermayer 2000)—there are only a few theoretical studies of the cross-domain ranking problem. The major difference between the cross-domain ranking problem and the learning to rank (Herbrich, Graepel, and Obermayer 2000) is that (1) cross-domain ranking needs consider all the interrelated domains, which may have a different feature space or different feature distributions; (2) the numbers of labeled training examples in the interrelated domains may be very unbalanced (e.g., thousands versus a few). Cross-domain ranking is also different from transfer learning (Bickel, Brückner, and Scheffer 2007; Dai et al. 2007), whose goal is to utilize the training data from a source domain to help learn a model for the target domain.

The problem of cross-domain ranking is non-trivial and poses challenges, because the pooled training data may improve the performance of a ranking function or may make the ranking performance worse. A fundamental issue is how to measure the correlation between different domains and how to learn all the ranking functions by taking advantage of the correlation.

To address these challenges, we propose a unified cross-domain ranking model: BayCDR. BayCDR uses a latent feature space defined across multiple interrelated domains to measure the correlation between domains. Examples from different domains are mapped onto the latent space via a projection matrix, where a common feature set is discovered by using a regularization method. Given a prior distribution of the ranking model and a ranking rule based on a weighted average over an ensemble of ranking models, BayCDR adopts an EM-style algorithm to learn a posterior distribution of the ranking model in the latent space. We show that learning multiple ranking functions simultaneously can reduce the ranking errors. In addition, to scale to large datasets, BayCDR is designed with an efficient distributed learning algorithm that is implemented and tested under the Map-Reduce framework. Experimental results on several real ranking tasks show the effectiveness and efficiency of BayCDR.

The rest of the paper is organized as follows: Section formally formulates the problem; Section explains the proposed approach. Section presents experimental results. Finally, Section concludes.

Problem Formulation

The cross-domain ranking problem can be formalized as follows.

Input: Let T be the number of domains, and for each domain $t \in \{1, \dots, T\}$, we are given n_t retrieved objects for query q: $\{(x^q_{t1}, y^q_{t1}), \ldots, (x^q_{t1}, y^q_{tn_t})\} \in \mathbb{R}^K \times \mathbb{R}$, where x^q_{ti} is expressed as a K-dimensional feature vector, and $y^q_{ti} \in \{r_1, \ldots, r_{l_t}\}$ denotes the ranking level assigned to the object for the query q. Notation l_t denotes the number of ranking levels in domain t. The ranking levels for domain t have: $r_1 \succ r_2 \succ \cdots \succ r_{l_t}$, where $\succ$ denotes the preference relationship.

The underlying assumption of the cross-domain ranking problem is that the multiple input domains are related, but allow examples from different domains to have different feature distributions.

Learning: The goal of cross-domain ranking is to learn T ranking functions $\{f_t\}_{t=1}^T$, with $f_t : \mathbb{R}^K \rightarrow \mathbb{R}$ for predicting the rank levels of unlabeled examples. A commonly used ranking function is linear—that is, $f_t(x^q_{ti}) = w_t^\top x^q_{ti} + b$—where w_t are parameters to be estimated from the training data and b is a bias term, which can be further absorbed by adding one dimension (as b) to w_t and one dimension (of value 1) to x_{ti}. Thus we have a simple form: $f_t(x^q_{ti}) = w_t^\top x^q_{ti}$. Extensions to nonlinear functions can be done, for example, by using kernel tricks (Herbrich, Graepel, and Obermayer 2000). Let us denote by W the $K \times T$ matrix whose columns are the vector w_t.

The ranking function can be learned independently from each domain, but learning all the ranking functions simultaneously is often desirable, and it can improve the ranking performance (Argyriou, Evgeniou, and Pontil 2006; Crammer, Kearns, and Wortman 2006).

Ranking: Based on the learned ranking functions $\{f_t\}$, we can predict the ranking level of a new object from, for example, domain t, for the query q by $f_t(x^q_{ti})$.

BayCDR: Cross-domain Ranking via Latent Space and Bayesian Learning

The goal of cross-domain ranking is to capture the following information: correlation between different domains and ranking preference relationships in each domain. In addition, the approach should be able to scale up to a large dataset. Following this thread, we propose a BayCDR model to incorporate all the information into a unified Bayesian model. Second, we propose a variational inference algorithm for model learning with Laplace priors. Third, we discuss how to do distributed learning in the Map-Reduce framework.

The BayCDR Model

Let us first consider only one domain (say domain t). To learn the parameter w_t, we can define the objective function:

$$\min_{w_t} \frac{1}{2}\|w_t\|_2^2$$
$$s.t. \quad \forall q, y^q_{ti} \succ y^q_{tj} : w_t^\top(\phi(x^q_{ti}) - \phi(x^q_{tj})) \geq 1, \quad (1)$$

where the constraint corresponds to the empirical loss—i.e., we have the loss $L(x^q_{ti}, y^q_{ti}) = \mathbb{I}(y^q_{ti} \not\succ y^q_{tj}) w_t^\top(\phi(x^q_{ti}) - \phi(x^q_{tj})) - 1$; here $\mathbb{I}(y^q_{ti} \not\succ y^q_{tj})$ is an indicator function that equals to 1 if the argument is true and 0 otherwise; $\phi(x)$ is a transformation function that projects the example x onto a new space.

To generalize the problem to multiple interrelated domains, we have two main ideas: first, we use a $K \times K$ matrix U to describe the correlation between features. The inner product of the examples is then defined as $x_i^\top U U^\top x_j$, using the matrix. Such parameterization is equivalent to projecting every example x onto a latent space spanned by $U : \; x \rightarrow U^\top x$. Second, we take a Bayesian approach to learn a parameter distribution $p(w_t)$, rather than a point estimate of w_t. The Bayesian approach results in a weighted averaging model for ranking, which enjoys a desirable smoothing effect. Thus the empirical loss can be redefined as $L(x^q_{ti}, x^q_{tj}) = \langle \mathbb{I}(y^q_{ti} \not\succ y^q_{tj}) w_t^\top(\phi(x^q_{ti}) - \phi(x^q_{tj}))\rangle_{p(w_t)} - 1$, where $\langle . \rangle_p = E_p[.]$ denotes the expectation operator w.r.t. distribution p. Now, the problem is to find the best distribution $p(w_t)$ from the possible space. In Bayesian learning, a common way, according to the maximum entropy principle (Jaakkola, Meila, and Jebara 1999), to solving this problem is to minimize the relative entropy of the distribution w.r.t. some chosen prior p_0. The relative entropy can be measured by Kullback-Leibler divergence, $KL(p||p_0) = \langle log(p/p_0)\rangle_p$. Therefore, for the cross-domain ranking problem, we can define the objective function as:

$$\min_{\{p(w_t)\}} \sum_t KL(p(w_t)||p_0(w_t))$$
$$s.t. \quad \forall t, q, y^q_{ti} \succ y^q_{tj} : \langle U^\top(x^q_{ti} - x^q_{tj})\rangle_{p(w_t)} \geq 1$$
$$\forall t : \int p(w_t) dw_t = 1, \quad (2)$$

where $KL(p||p_0) = \langle log(p/p_0)\rangle_p$ represents the KL divergence between the learned distribution p and the prior distribution p_0. To choose the best distribution p, we minimize its relative entropy w.r.t. the prior p_0. We can add a slack variable ξ_{ij} to absorb errors in the training data; thus, the optimization problem in Eq. 1 is extended as $\frac{1}{2}\|w_t\|_2^2 + C\sum\xi$ and Eq. 2 as $\sum_t KL(p(w_t)||p_0(w_t)) + B(\xi)$, where C is a constant and $B(\xi)$ can be defined as any closed proper convex function over the slack variables (also known as an additional "potential" term in the maximum entropy principle).

From the above discussion, we can see that we use an average model weighted by the distribution $p(w_t)$ as the ranking function; each example x is transformed into a latent space by the projection matrix U, which is shared across all domains. This elegant combination of the latent space and the Bayesian learning offers a number of important advantages. For example, the Bayesian learning can be interpreted as posterior model estimates by assuming a Laplace prior on the model parameters, which introduces useful sparsity effects; the ranking based on the weighted averaging model also enjoys a desirable smoothing effect; the sparse and latent space also offer a principled way to interpret the correlation between different domains.

Learning Algorithm

In BayCDR, we do not simply want to learn the ranking function f_t for each domain, but also to learn the matrix U. The matrix reflects a projection to the latent space, in which it is desirable to leave out features that are not relevant. Traditionally, this has been done by using the L_1 norm. However, in some applications, the L_1 will also leave out features that are not completely irrelevant, which hurts the regularization ability (Zhu, Xing, and Zhang 2008).

In Bayesian learning, to favor a sparse estimate, a Laplace prior can be adopted for w_t. We apply a common prior λ to parameters $\{w_t\}_t$ of all domains; that is, $p(w_t|\lambda) = \prod_{k=0}^{K} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w_{tk}|} = (\frac{\sqrt{\lambda}}{2})^k e^{-\sqrt{\lambda}\|w_t\|_1}$. The Laplace density is heavy-tailed and peaked at zero; hence, it encodes prior belief that the distribution of all w_t is strongly peaked around zero. In addition, the Laplace density is log-convex, and can be used to get convexity estimates of the posterior density (Kabán 2007; Park and Casella 2008).

Generally, the Laplacian prior is equivalent to a two-level hierarchical-Bayes model: zero-mean Gaussian priors with independent, exponentially distributed variances. This equivalence has been previously used to derive EM algorithms for discriminative learning (Figueiredo 2003) and discrimination Markov networks (Zhu, Xing, and Zhang 2008). More specifically, let us consider that each w_{tk} has a zero-mean Gaussian prior $p(w_{tk}|\tau_{tk}) = \mathcal{N}(w_{tk}|0, \tau_{tk})$, with its own variance τ_{tk}, and that each τ_{tk} has an exponential (hyper)prior

$$p(\tau_{tk}|\lambda) = \frac{\lambda}{2} e^{-\frac{\lambda}{2}\tau_{tk}}, \quad \text{for } \tau_{tk} \geq 0. \tag{3}$$

Then, by integrating out τ_{tk}, we have

$$\begin{aligned} p_0(w_t|\lambda) &= \prod_{k=1}^{K} \int p(w_{tk}|\tau_{tk}) p(\tau_{tk}|\lambda) d\tau_{tk} \\ &= \int p(w_t|\tau_t) p(\tau_t|\lambda) d\tau_t, \end{aligned} \tag{4}$$

where $p(w_t|\tau_t) = \prod_{k=1}^{K} p(w_{tk}|\tau_{tk})$ and $p(\tau_t|\lambda) = \prod_{k=1}^{K} p(\tau_{tk}|\lambda)$ are joint distributions and $d\tau_t \triangleq d\tau_{t1} \cdots \tau_{tK}$. Using the hierarchical representation of the Laplace prior and applying the Jensen's inequality, we get an upper bound of the KL-divergence,

$$\begin{aligned} KL(p\|p_0) &= -H(p) - \langle log \int p(w|\tau) p(\tau|\lambda) \rangle_p \\ &\leq -H(p) - \langle \int q(\tau) log \frac{p(w|\tau)p(\tau|\lambda)}{q(\tau)} \rangle_p \\ &\triangleq \mathcal{E}(p(w), q(\tau)), \end{aligned} \tag{5}$$

where $q(\tau)$ is a variational distribution which is used to approximate $p(\tau|\lambda)$.

Substituting this upper bound for KL into Eq. 2, we can obtain a new objective function with the same constraints:

$$\min_{\{p(w_t)\},\{q(\tau_t)\}} \sum_t \mathcal{E}(p(w_t), q(\tau_t)). \tag{6}$$

Directly solving the above objective function (involving solving parameters $\{p(w_t)\}, \{q(\tau_t)\}, U$ is intractable. But as both the KL-divergence and the projection matrix U are convex, we can obtain a dual form of the problem, and solve the dual problem with an iterative minimization algorithm, as outlined in Algorithm 1, and detailed as follows.

```
Input: training data from T domains $\{(x_{ti}^q, y_{ti}^q)\}_{t,i,q}$ and
       constant $\gamma$
Output: posterior mean for each domain: $\{\langle w_t \rangle_{p(w_t)}\}$,
        and the projection matrix $U$
1.1  Initialize $\{\langle w_t \rangle_{p(w_t)}\} \leftarrow 0$, $\Sigma_{w_t} \leftarrow 0$, $D = \frac{I_{K\times K}}{K}$;
1.2  repeat
1.3      Step E: Calculate $\langle w_t \rangle_{p(w_t)}$ and projection matrix $U$;
1.4      repeat
1.5          Step 1: foreach domain $t \in \{1,\dots,T\}$ do
1.6              Compute
1.7              $w_t = \arg\min \sum_{q, y_{ti}^q \succ y_{tj}^q} [1 - U^\top (x_{ti}^q - x_{tj}^q)\rangle_{p(w_t)}]_+ + \gamma\|A\|_{2,1}$; s.t. $A \in \mathbb{R}^{K\times K}$;
1.8          end
1.9          Step 2: Compute $D = \frac{(AA^\top)^{1/2}}{trace(AA^\top)^{1/2}}$;
1.10     until convergence;
1.11     Apply SVD decomposition on $D$, $D = U\Sigma V^\top$;
1.12     Construct $U$ with the biggest eigenvalues of $D$;
1.13     Step M: Update $\Sigma_{w_t} \leftarrow \mathrm{diag}(\sqrt{\frac{\langle w_{tk}^2 \rangle_{p(w_t)}}{\lambda}})$;
1.14 until convergence;
```

Algorithm 1: The learning algorithm for BayCDR.

E-step. Keep $\{q(\tau_t)\}$ fixed we can optimize Eq. 6 with respect to $p(w_t)$. In general, we update $\{p(w_t)\}$ as follows: (Derivation is omitted due to space limitation.)

$$\begin{aligned} p(w_t) &\propto \exp\{\int q(\tau) \mathrm{log} p(w_t|\tau_t) d\tau_t - b + w_t^\top \eta - d\} \\ &\propto \exp\{-\frac{1}{2} w_t^\top \langle A_t^{-1} \rangle_{q(\tau_t)} w_t - b + w_t^\top \eta - d\} \\ &= \mathcal{N}(w_t|\mu_{w_t}, \Sigma_{w_t}), \end{aligned} \tag{7}$$

where $b = \langle \mathrm{log} p(\tau_t|\lambda) \rangle_{q(\tau_t)} - H(q(\tau_t))$; $\eta = \sum_{q, y_{ti}^q \succ y_{tj}^q} \langle \mathbb{I}(y_{ti}^q \not\succ y_{tj}^q) U^\top (x_{ti}^q - x_{tj}^q) \rangle_{p(w_t)} - 1$; $d = \sum_{ti} \alpha_{ti}$; and $A_t = \mathrm{diag}\tau_{tk}$. The posterior mean and variance are $\mu_{w_t} = \langle w_t \rangle_{p(w_t)}$ and $\Sigma_{w_t} = (\langle A_t \rangle_{q(\tau_t)}^{-1})^{-1} = \langle w_t U^\top w_t^\top \rangle_{p(w_t)} - \langle w_t U \rangle_{p(w_t)} \langle w_t U \rangle_{p(w_t)}^\top$. Now the problem is how to estimate the dual parameters α and the projection matrix U. Because the hinge-loss style (Bishop 2006; Gentile and Warmuth 1999) objective in the latent space (by U) is convex, we can solve the following problem:

$$\begin{aligned} \min_{w,U} \sum_{t=1}^{T} \sum_{q, y_{ti}^q \succ y_{tj}^q} & [1 - \langle U^\top (x_{ti}^q - x_{tj}^q) \rangle_{p(w_t)}]_+ + \gamma\|A\|_{2,1} \\ \text{s.t.} \quad & A \in \mathbb{R}^{K\times K}, \end{aligned} \tag{8}$$

where the subscript "+" indicates the positive part; γ is a parameter that controls the tradeoff between the empirical loss (the first term) and the penalty (the second term) of the model complexity; $A = W^\top U$, which implies that we penalize the model complexity in the projected space instead of the original space W. The problem can be solved using two sub-steps: Step 1 and 2. In Step 1, we fix U and update w_t for each domain t; and in Step 2, we fix W and update the matrix U. Specifically, in Step 2, we first calculate an intermedia matrix $D = \frac{(AA^\top)^{1/2}}{trace(AA^\top)^{1/2}}$, where $trace(A) = \sum_{i=1}^{|A|} A_{ii}$; and then apply a SVD decomposition (Wall, Rechtsteiner, and Rocha 2003) on D; i.e., $D = U\Sigma V^\top$; then the matrix U is constructed with the biggest eigenvalues of D.

After obtaining α and U, we calculate the posterior mean of $\{w_t\}$ under $p(w_t)$. As $p(w_t)$ is a normal distribution, the posterior mean is the only parameter that is needed for ranking.

M-step. Keep $\{p(w_t)\}$ fixed, we optimize Eq. 6 w.r.t. $q(\tau_t)$. Taking the derivative of Eq. 6 w.r.t. $q(\tau_t)$ and setting it to zero, then we get

$$q(\tau_t) \propto p(\tau_t|\lambda) \cdot \exp\{\langle \log p(w_t|\tau)\rangle_{p(w_t)}\}. \quad (9)$$

By exploring the factorization forms of $p(w_t|\tau_t)$ and $p(\tau_t|\lambda)$, we can get an induced factorization $q(\tau_t) = \prod_{k=1}^{K} q(\tau_{tk})$, and each $q(\tau_{tk})$ is computed as follows:

$$\begin{aligned} q(\tau_{tk}) &\propto p(\tau_{tk}|\lambda) \cdot \exp\{\langle \log p(w_{tk}|\tau_{tk})\rangle_{p(w_t)}\} \\ &\propto \mathcal{N}(\sqrt{\langle w_{tk}^2\rangle_{p(w_t)}}|0, \tau_{tk}) \cdot \exp(-\frac{1}{2}\lambda\tau_{tk}). \end{aligned} \quad (10)$$

Such a derivation has been previously used for a maximum margin Markov network (Zhu, Xing, and Zhang 2008) and Bayesian classification (Kabán 2007). Similarly we can get the normalization factor: $\int \mathcal{N}(\sqrt{\langle w_{tk}^2\rangle_{p(w_t)}}|0, \tau_{tk}) \exp(-\frac{1}{2}\lambda\tau_{tk}) d\tau_{tk} = \frac{\sqrt{\lambda}}{2}\exp(-\sqrt{\lambda\langle w_{tk}^2\rangle_{p(w_t)}})$. Furthermore, in an analogous way to (Kabán 2007), we can calculate the expectations $\langle \tau_{tk}^{-1}\rangle_{q(\tau_t)}$ which are required in calculating $\langle A_t^{-1}\rangle_{q(\tau_t)}$ as follows,

$$\langle \frac{1}{\tau_{tk}}\rangle_{q(\tau_t)} = \int \frac{1}{\tau_k} q(\tau_{tk}) d\tau_{tk} = \sqrt{\frac{\lambda}{\langle w_{tk}^2\rangle_{p(w_t)}}}. \quad (11)$$

We run the above E and M steps iteratively until convergence. Then we use the posterior distribution $p(w_t)$ to predict the ranking level for domain t. For irrelevant features, the variances should converge to zero and thus lead to a sparse estimation. The generalization bound of the algorithm is given in the supplementary file.

Distributed Learning

BayCDR has a complexity of $O(M(K^2 + K + (2L + 1)KNlog(N)))$, where M is the number of EM iterations, L is the number of the two sub-step iterations, and N is the total number of examples from all domains. The complexity is high and it is impractical to scale up to large datasets. To address this challenge, we deploy the learning task on a distributed system under the map-reduce programming model (Dean and Ghemawat 2004). *Map-Reduce* is a programming model for distributed processing of large datasets. In the *map* stage, each machine (called a process node) receives a subset of data as input and produces a set of intermediate key/value pairs. In the *reduce* stage, each process node merges all intermediate values associated with the same intermediate key and outputs the final computation results.

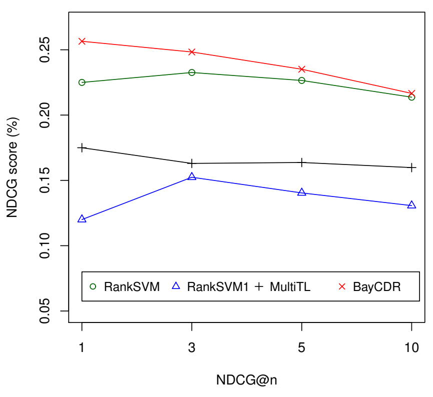
Generally, BayCDR consists of two steps—i.e., E and M steps. In the E step, the learning algorithm needs to solve a quadratic optimization problem, which is actually the major computational cost in BayCDR. The quadratic optimization problem is similar to Support Vector Machine (Cortes and Vapnik 1995), which has been parallelized by (Chu et al. 2006). Therefore, we define the map stage and the reduce stage as follows. In the map stage, each process node calculates the gradient descent for w_t on a subset of the example pairs: $\nabla = 2w_t + 2\frac{1}{\lambda}\sum_{sub(y_{ti} \succ y_{tj})} \langle (x_{ti} - x_{tj})^\top UU^\top (x_{ti} - x_{tj})\rangle_{p(w_t)}$. Thus the key/value pair corresponds to the instance pair (i, j) and the partial gradient. In the reduce stage, each process node collects all values associated with each intermediate key (i, j) and sums up the partial gradient to update w_t for each domain. Next it calculates the matrix D and U and finally updates Σ_{w_t}.

Experiments

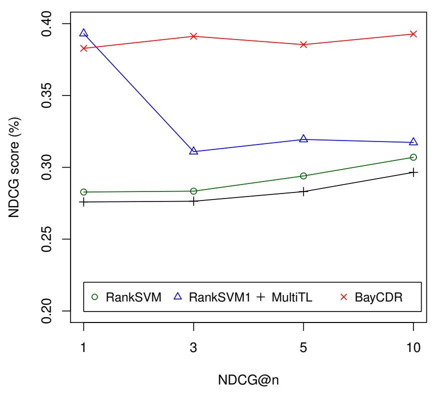
We evaluate BayCDR on three different genres of datasets: homogeneous data, heterogeneous data, and heterogeneous tasks; and compare it with RankSVM (Joachims 2002), RankSVM$_1$ (learning only one ranking function by combining training examples from all domains), and multi-task learning (MultiTL) (Argyriou, Evgeniou, and Pontil 2006), which tries to learn the classification/regression models for multi-tasks.

Datasets and Evaluation Criteria

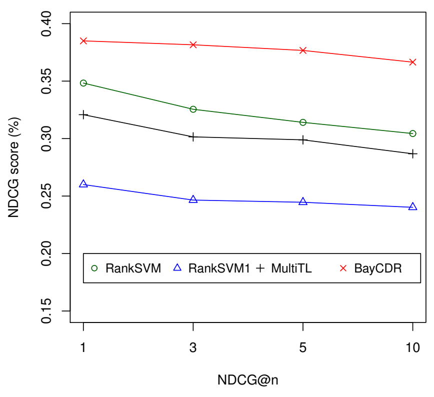
The first (homogeneous) dataset is LETOR 2.0 (Liu et al. 2007), a public dataset for learning to rank research. LETOR consists of three sub datasets (i.e., TREC2003, TREC2004, and OHSUMED), with 50, 75, and 106 queries, respectively. Given a query, the task is to identify which documents are relevant to the query. There are three rank levels in LETOR—i.e., relevant $\succ$ partially relevant $\succ$ non-relevant. The second (heterogeneous) dataset is an academic dataset, which is also publicly available. The dataset contains 14,134 authors, 10,716 papers, and 1,434 conferences. Given a query, the goal is to find experts, top conferences, and authoritative papers for the query. There are 44 queries and four rank levels in the dataset. The third dataset is the same as the second one, but it has two different ranking tasks: finding experts and finding best supervisors. An expert can be a good supervisor, but not necessarily. Therefore, the two tasks are related but different. In the first dataset, there are six domains (two for each data subset), with each data subset consisting of 5,726-13,761 training examples. In the second dataset, the numbers of training examples from different domains vary from 312 to 3,717. In the third dataset,



(a) TREC2003
KL divergence=2.42

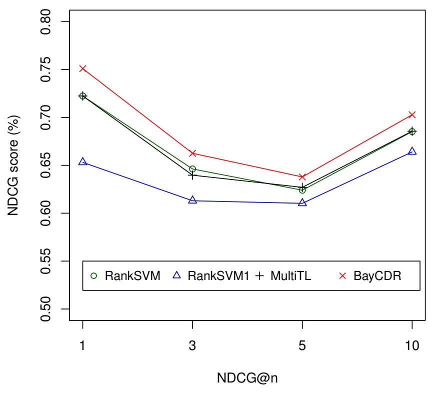


(b) TREC2004
KL divergence=0.69

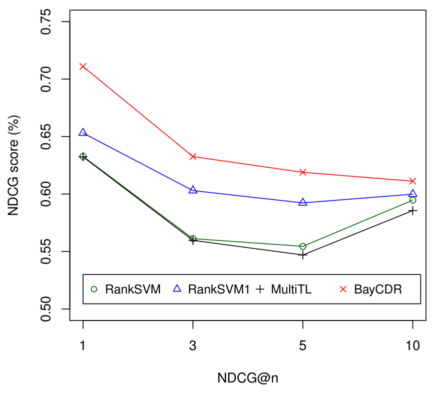


(c) OHSUMED
KL divergence=2.20

Figure 1: Ranking accuracy on the three tasks of the LETOR data.



(a) Expert finding
KL divergence=5.72



(b) Multi-task ranking
KL divergence=1.26

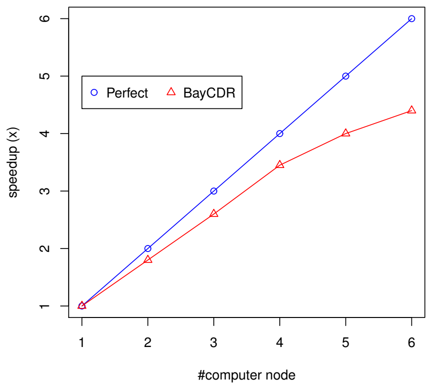
Figure 2: Ranking accuracy on the academic data. (a) finding experts, top conferences, authoritative papers; (b) finding experts and best supervisors.

we sent 732 emails to ask related researchers for the ground truth of "best supervisors". Finally, we obtained 270 training examples. The difference of feature distribution on the first dataset is relative small (2.42, 0.69, and 2.20 in terms of KL-divergence, respectively for the three data subsets), while the difference in the other two datasets is relative large (5.72 and 1.26 in terms of KL-divergence). As an evaluation measure, we utilize NDCG@n (Normalized Discounted Cumulative Gain) (Järvelin and Kekäläinen 2002). In all the experiments, we conduct five-fold cross-validation for all the methods.

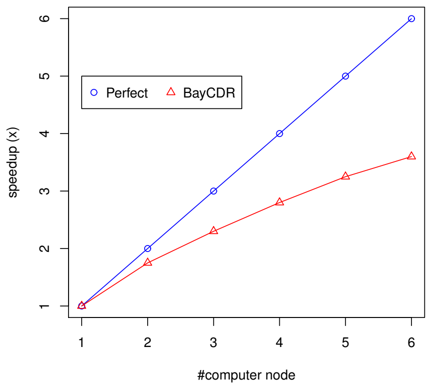
Accuracy Performance

Figure 1 shows the ranking accuracy of the different methods, in terms of NDCG@n on the three data subsets in LETOR. BayCDR clearly outperforms the comparison methods. Figure 2 (a) shows the average performance of ranking the three objects (experts, conferences, and papers) on the academic data, and Figure 2 (b) shows the average performance of the two different ranking tasks (finding experts and finding the best supervisors). BayCDR achieves higher performance than the three comparison methods.

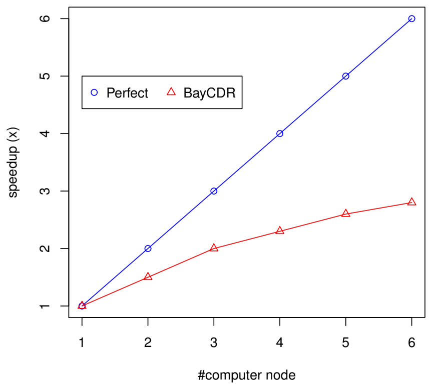
From the experiments, we can see an interesting pattern: when the differences of the interrelated domains are large (e.g., TREC2003 and OHSUMED), independent learning (RankSVM) and multi-task learning (MultiTL) perform better than learning only one ranking function (RankSVM$_1$). On the other hand, when the difference is relative small, RankSVM and MultiTL underperform RankSVM$_1$. BayCDR results in a balanced performance and consistently outperforms all the comparison methods. The result suggests that it is better to learn the ranking functions and domain correlation simultaneously. RankSVM$_1$ only learns one ranking function, thus with the domains' difference increasing, the ranking performance decreases. RankSVM learns a ranking function for each domain; however, it does not consider



(a) LETOR



(b) Expert finding



(c) Multi-task data

Figure 3: Speedup results on the three datasets.

the correlation between domains, and thus cannot utilize the knowledge from "related" domains.

Scalability Performance

We evaluate the speedup of the distributed learning algorithm on the three datasets using 1-6 computer nodes (since we did not have access to a large number of computer nodes). It can be seen from Figure 3 that on the large dataset (LETOR), the distributed learning shows a good parallel efficiency (speedup $> 4\times$, as Figure 3 (a)); on the other two relative small datasets, the speedup also shows reasonable parallel efficiency ($\geq 2.8\times$, as Figure 3 (b) and (c)). This confirms that the distributed learning algorithm is necessary on large-scale datasets.

Related Work

Considerable work has been conducted for supervised learning to rank. Existing approaches include pointwise approach, pairwise approach and listwise approach. In pointwise approaches, the ranking problem aims at predicting the rank level of an object. In pairwise approaches, the ranking problem can be reduced to a classification problem by comparing the rank levels of each instance pairs. Ranking SVM (Herbrich, Graepel, and Obermayer 2000), RankBoost and RankNet (Burges et al. 2005) are three state-of-the-art algorithms in this category. Other methods can be also found in (Wauthier, Jordan, and Jojic 2013). In listwise approaches, the ranking problem is formulated to directly optimize some listwise performance measures of information retrieval (Xu and Li 2007; Yue et al. 2007). There is also some work on ranking by semi-supervised learning and transductive learning. For example, Duh and Kirchhoff propose a framework for ranking problem in the transductive setting (Duh and Kirchhoff 2008). Amini et al. propose a semi-supervised rankboost algorithm (Amini, Truong, and Goutte 2008). Hoi and Jin propose a semi-supervised ensemble ranking with a SVM-like formulation (Hoi and Jin 2008). Some work takes the relations between objects to be ranked into consideration. Agarwal et al. propose a framework for ranking networked entities based on a maximum entropy flow algorithm with weight function for different types of edges (Agarwal, Chakrabarti, and Aggarwal 2006). There are also many applications of learning to rank such as ranking Paraphrases from query logs (Figueroa and Neumann 2013), recommendation (Pan, Xiang, and Yang 2012), and cross-domain recommendation (Tang et al. 2012). However, all these existing works mainly focus on single domain and do not consider the cross domain ranking problem.

Another related work is transfer learning, which aims to transfer knowledge from a source domain to a related target domain. Two fundamental issues in transfer learning are "what to transfer" and "when to transfer". Many approaches have been proposed by reweighting instances in source domain for the use in target domain (Dai et al. 2007). Gao et al. propose a locally weighted ensemble framework which can utilize different models for transferring labeled information from multiple training domains (Gao et al. 2008). Also many works have been done based on new feature representation (Jebara 2004; Lee et al. 2007). There are also other approaches which transfer information by shared parameters (Bonilla, Chai, and ChrisWilliams 2008) or relational knowledge.

Conclusion

In this paper, we investigate a novel problem of cross-domain ranking. We propose a unified ranking model, named BayCDR, which offers a nice way to integrate the latent space extraction into the Bayesian learning. An efficient EM-style algorithm based on variational inference has been developed for BayCDR. We applied BayCDR to three real-world data sets and experimental results show that BayCDR performs better than the baseline methods. To scale up to real large data sets, a distributed learning algorithm has been developed and shows a good parallel efficiency in the experiments. In future, we plan to further integrate the link (relational) information into the BayCDR model and also try to apply BayCDR to other applications.

References

Agarwal, A.; Chakrabarti, S.; and Aggarwal, S. 2006. Learning to rank networked entities. In *KDD'06*, 14–23.

Amini, M.-R.; Truong, T.-V.; and Goutte, C. 2008. A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *SIGIR'08*, 99–106.

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2006. Multi-task feature learning. In *NIPS'06*, 41–48.

Ben-David, S.; Blitzer, J.; Crammer, K.; and Pereira, F. 2006. Analysis of representations for domain adaptation. In *NIPS'06*. 137–144.

Bickel, S.; Brückner, M.; and Scheffer, T. 2007. Discriminative learning for differing training and test distributions. In *ICML'07*, 81–88.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Bonilla, E.; Chai, K. M.; and ChrisWilliams. 2008. Multi-task gaussian process prediction. In *NIPS'08*, 153–160.

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

Chu, C.-T.; Kim, S. K.; Lin, Y.-A.; Yu, Y.; Bradski, G.; Ng, A. Y.; and Olukotun, K. 2006. Map-reduce for machine learning on multicore. In *NIPS'06*, 281–288.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20:273–297.

Crammer, K.; Kearns, M.; and Wortman, J. 2006. Learning from multiple sources. In *NIPS'06*. 321–328.

Dai, W.; Yang, Q.; Xue, G.-R.; and Yu, Y. 2007. Boosting for transfer learning. In *ICML'07*, 193–200.

Dean, J., and Ghemawat, S. 2004. Mapreduce: Simplified data processing on large clusters. In *OSDI'04*, 10–10.

Duh, K., and Kirchhoff, K. 2008. Learning to rank with partially-labeled data. In *SIGIR'08*, 251–258.

Figueiredo, M. A. T. 2003. Adaptive sparseness for supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 25(9):1150–1159.

Figueroa, A., and Neumann, G. 2013. Learning to rank effective paraphrases from query logs for community question answering. In *AAAI'13*, 1099–1105.

Gao, J.; Fan, W.; Jian, J.; and Han, J. 2008. Knowledge transfer via multiple model local structure mapping. In *KDD'08*, 283–291.

Gentile, C., and Warmuth, M. K. 1999. Linear hinge loss and average margin. In *NIPS'99*, 225–231.

Herbrich, R.; Graepel, T.; and Obermayer, K. 2000. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA.

Hoi, S. C., and Jin, R. 2008. Semi-supervised ensemble ranking. In *AAAI'08*, 634–639.

Jaakkola, T.; Meila, M.; and Jebara, T. 1999. Maximum entropy discrimination. In *NIPS'99*, 470–476.

Järvelin, K., and Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems* 20(4):422–446.

Jebara, T. 2004. Multi-task feature and kernel selection for svms. In *ICML'04*, 55–62.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *KDD'02*, 133–142.

Kabán, A. 2007. On bayesian classification with laplace priors. *Pattern Recognition Letters* 28(10):1271–1282.

Lee, S.-I.; Chatalbashev, V.; Vickrey, D.; and Koller, D. 2007. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML'07*, 489–496.

Liu, T.-Y.; Xu, J.; Qin, T.; Xiong, W.; and Li, H. 2007. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007, in conjunction with SIGIR 2007*.

Pan, W.; Xiang, E. W.; and Yang, Q. 2012. Transfer learning in collaborative filtering with uncertain ratings. In *AAAI'12*, 662–668.

Park, T., and Casella, G. 2008. The bayesian lasso. *Journal of the American Statistical Association* 103(482):681–686.

Tang, J.; Wu, S.; Sun, J.; and Su, H. 2012. Cross-domain collaboration recommendation. In *KDD'12*, 1285–1294.

Wall, M. E.; Rechtsteiner, A.; and Rocha, L. M. 2003. *Singular value decomposition and principal component analysis*. Kluwer: Norwell, MA. 91–109.

Wauthier, F. L.; Jordan, M. I.; and Jojic, N. 2013. Efficient ranking from pairwise comparisons. In *ICML'13*, 109–117.

Xu, J., and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR'07*, 391–398.

Yue, Y.; Finley, T.; Radlinski, F.; and Joachims, T. 2007. A support vector method for optimizing average precision. In *SIGIR'07*, 271–278.

Zhu, J.; Xing, E. P.; and Zhang, B. 2008. Laplace maximum margin markov networks. In *ICML'08*, 1256–1263.