Propagating Ranking Functions on a Graph: Algorithms and Applications

Buyue Qian
IBM T. J. Watson Research
Yorktown Heights, NY 10598
bqian@us.ibm.com

Xiang Wang
IBM T. J. Watson Research
Yorktown Heights, NY 10598
wangxi@us.ibm.com

Ian Davidson
University of California
Davis, Davis, CA 95616
davidson@cs.ucdavis.edu

Abstract

Learning to rank is an emerging learning task that opens up a diverse set of applications. However, most existing work focuses on learning a single ranking function whilst in many real world applications, there can be many ranking functions to fulfill various retrieval tasks on the same data set. How to train many ranking functions is challenging due to the limited availability of training data which is further compounded when plentiful training data is available for a small subset of the ranking functions. This is particularly true in settings, such as personalized ranking/retrieval, where each person requires a unique ranking function according to their preference, but only the functions of the persons who provide sufficient ratings (of objects, such as movies and music) can be well trained. To address this, we propose to construct a graph where each node corresponds to a retrieval task, and then propagate ranking functions on the graph. We illustrate the usefulness of the idea of propagating ranking functions and our method by exploring two real world applications.

Introduction

Any system that presents ordered results to users is performing ranking (Burges et al. 2005). This has led to extensive interest in applying machine learning techniques to learn ranking functions. Such techniques are called learning to rank, whose goal is to automatically build a retrieval function from training data, and has been successfully applied to various information retrieval problems. In many real world applications, we often require many ranking functions to fulfill various retrieval tasks on a single dataset. It is often the case that the training data is limited and imbalanced hence, only a few ranking functions can be trained adequately (deemed as *strong*), while the majority ranking functions are completely *unknown* or learnt with little training data (deemed as *weak*). Consider a personalized movie ranking problem, each person has different taste of movies and thus requires a unique ranking function. However, for a particular user, a *strong* ranking function can be learned only if that user provides sufficient movie ratings, which is often not the case in practice. In this study we consider using graph diffusion methods to estimate the *unknown* or improve the *weak* ranking functions by those *strong* ones. The purpose of this work is to propagate linear ranking functions over the nodes on a graph, where each node corresponds to a retrieval task/user.

Problem Setting. There are three key concepts in this work, (i) linear ranking function, (ii) ordinal distribution features, and (iii) ranking function propagation. We shall now explain each briefly. A *linear ranking function* is a real-valued vector, whose inner product with a data point is usually called a ranking score. In a retrieval task using a linear ranking function, we first calculate ranking scores, the inner product between the ranking function vector and all data points, and then sort the data points based on the ranking scores. Data points with higher ranking scores are more relevant to the retrieval topic. *Ordinal distribution features* are a powerful way of representing complex objects by the counts of features that are ordinal, that is the features represent an ordered set of values such as `small, medium, large`. Consider our experiments on creating a ranking function for each type of bird based on their song. The songs are represented using spectrograms which contains the counts of each audio frequency and analogous to histograms. The spectrograms capture not only the counts of each frequency but also that adjacent (in the spectrogram) frequencies are similar. *Propagation of ranking functions* implies that in our study a graph can be comprised of *strong*, *weak*, and *unknown* ranking functions, and we allow the *strong* ranking functions being propagated to the *weak* or *unknown* functions. In particular we can calculate the ranking functions after an infinite number of propagation steps on the network.

Proposal. To explore the graph propagation of linear ranking functions on distribution data, there are three main challenges that we address. (i) Since our ranking function is of ordinal (order sensitive features) regular label propagation methods will fare poorly with these ordinal distribution features. We adopt the Hellinger distance metric to calculate the similarity between two linear ranking functions (distributions). (ii) Propagating distributed based ranking functions requires a specialized objective function. We generalize the objective functions of several classic diffusion methods whose objectives have shown to be useful in solving real problems. (iii) The construction of a similarity graph to propagate the ranking functions will vary between applications. For example, in a personalized movie ranking problem, the graph can be the user co-rating matrix or constructed using side information such as user profiles.

Though the goals look similar, our work differs from multi-task learning (MTL) in two aspects. (i) MTL usually assumes just a few tasks whilst our method performs propagation on a large set of ranking functions (2000+ in one of our experiments). (ii) While MTL tries to learn a set of cooperative tasks together, our method views the learning of the initial set of ranking functions as a preprocessing step, and only focuses on the inference of unknown ranking functions.

Contribution and Claims. Our work makes the following three technical contributions. (i) We explore a new method to **efficiently** learn thousands of ranking functions from insufficient and imbalanced training data. (ii) We propose a generic diffusion method to propagate linear ranking functions modeled as distributions on a graph. Our approach allows for interactions between adjacent histogram bins making propagation more useful. (iii) The proposed method can be naturally deployed to network settings, such as personalized ranking and social network advertising.

Related Work

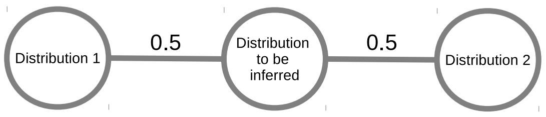
Our work relates to two topics: label propagation and learning to rank. We briefly review related work in both areas.

Graph-based label propagation mainly aims to address the issue of transduction. Blum and Chawla have proposed label propagation methods on graphs as a mincut problem. The Gaussian fields and harmonic function (GFHF) method in (2003) is a continuous relaxation to the difficult discrete Markov random fields. The local and global consistency (LGC) method (Zhou et al. 2003) extends GFHF with the normalized Laplacian. Another manifold regularization framework (2006) employs two regularization terms, i.e., a base kernel and a L_2 regularizer. Lawrence and Jordan propose to learn with unlabeled data in the context of Gaussian process. Comprehensive surveys on semi-supervised learning can be found in (Zhu 2005) and (Chapelle et al. 2006).

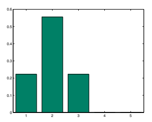
Learning to rank algorithms fall into three categories which differ in the form of training data. (i) *Pointwise* methods approximate a ranking problem as ordinal regression with large margin formulations (Shashua and Levin 2002) (Crammer and Singer 2001), and subset ranking formulations(Cossock and Zhang 2006). (ii) *Pairwise* methods learn from pairs of instances ordered by their relevance to the ranking problem (iii) *Listwise* use a fully ordered rank list as an instance, e.g., ListNet (Cao et al. 2007), AdaRank (Xu and Li 2007) and SVM Map (Yue et al. 2007). Learning to rank approaches have shown to be useful in many information retrieval applications. RankSVM was applied Joachims to document retrieval, RankNet (Burges et al. 2005) has shown to be useful on large scale web search. A brief survey of learning to rank approaches exists (Hang 2011).

The Importance of Propagating Distributions

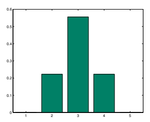
Feature representations of data objects fall into two categories. (i) Independent features, where the ordering of features is permutable without affecting the learning result. (ii) Ordinal distribution/histogram features, **where the ordering of features matters** to learning algorithms. The latter type of ordinal distribution features is commonly used in learning to rank problems and also covers a large portion of data representations, such as the color histogram used in computer vision, the spectrogram used in natural language processing, and various transformations used in signal processing. While on the data of independent features, ranking functions can be readily propagated using typical label propagation methods, there is no available method, to our knowledge, to perform ranking function propagation on distribution data. Existing graph diffusion methods perform propagation in a bin-by-bin fashion and ignore the interactions between adjacent histogram bins. We shall in the following example show that sometimes shifting and preserving shape are preferred.



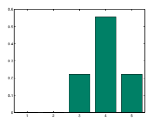
(a) A graph with each node correspondes to a distribution/ranking



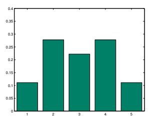
(b) Distirbution 1 (three bars on the left)



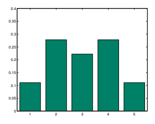
(c) Ideal propagation (three bars in the mid)



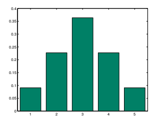
(d) Distribution 2 (three bars on the right)



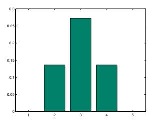
(e) GFHF propagation



(f) LGC propagation



(g) Our propagation



(h) Our propagation minus a constant

Figure 1: Distribution propagation using different methods. Note that Figure 1(h) is equivalent to Figure 1(g), since each distribution here is viewed as a linear ranking function

To illustrate the importance of distribution propagation, let us consider a toy graph with only three nodes. Figure 1(a) presents a simple graph with three nodes, each of which corresponds to a distribution (linear ranking function). We see that in the given Distribution 1 (Figure 1(b)) there are three bars on the left side of the 5-bin histogram with the middle bar higher than the other two, and the Distribution 2 (Figure 1(d)) follows the same shape as Distribution 1 but locates on the right side of the 5-bin histogram. It is clear that shifting Distribution 1 to the right by two bins is equivalent to Distribution 2, and vice versa. Therefore, we can infer the ideal propagation on distribution data as in Figure 1(c).

We next infer the missing distribution (in the middle) using different graph propagation methods. Figure 1(e) and 1(f) show the inferred distributions using two classic graph diffusion methods, GFHF and LGC, respectively. We observe that traditional methods do not work well on distribution data, since their result is essentially a weighted average of the given distributions. Figure 1(g) shows the result of our method, which captured the key structure of the two given distributions (higher in the middle and lower on both sides). Since our study concerns ranking function propagation, each distribution here can be viewed as a linear ranking function. As adding a constant to a linear ranking function on distribution data (nonnegative and sum-to-one) does not change the ranking order, we can remove the redundant constant and have an equivalent ranking function as shown in Figure 1(h). The result is very close to the ideal propagation.

The Method

Preliminaries

Formally, the problem to address is described as follows. In a retrieval problem, we are given a dataset containing n instances $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$, which are represented as distributions in $\mathbb{R}^d$ space. Since ranking functions of non-distribution data can be readily handled by traditional graph propagation methods, this paper only concerns ordinal distribution based feature representations, i.e., we assume the features of data are (i) nonnegative, (ii) sum-to-one, and (iii) the order of features is not permutable. Such representations have been shown to be useful for ranking problems of images (Datta et al. 2008). On the set of instances $\mathcal{X}$, there are m ranking tasks/categories $\mathcal{T} = \{t_1, t_2, \cdots, t_m\}$, which correspond to a set of ranking functions $\mathcal{W} = \{w_1, w_2, \cdots, w_m\}$ respectively. Each ranking function is defined with respect to an external ordering (retrieval task) that we are trying to learn from the data. Since in our approach we adopt a linear ranking function, i.e. w_i is a real-valued vector with the same dimension as x_i, the ranking score of an instance x_i w.r.t. a retrieval tasks t_j is simply calculated from the inner product between x_i and w_j.

$$\gamma_i^j = w_j^T x_i \tag{1}$$

We construct a graph over ranking functions using a measure of similarity that is application dependent and allow the interactions amongst themselves. In particular, we assume that among the m ranking functions, there are only a small portion of ranking functions that are trained with sufficient data (deemed as *strong* functions), and the majority remaining ones are unknown or learned with little training data (deemed as *unknown* or *weak* functions). In this paper we present a general approach to propagate linear ranking functions, and the input of our method is a set of ranking functions. Any learning to rank methods that produce linear ranking functions can be served as a preprocessing component for our approach.

Objective Function

To perform ranking function propagation, we first construct a graph of ranking functions, which is defined by a symmetric similarity/affinity matrix S, $S \in \mathbb{R}^{m \times m}$. Note that the graph contains both *strong* and *unknown/weak* ranking functions to enable the propagation of functions. An entry S_{ij} in the matrix S indicates the similarity between the retrieval tasks t_i and t_j, $S_{ij} = 1$ if t_i and t_j are exactly the same and $S_{ij} = 0$ if t_i and t_j are completely irrelevant, and we have $S_{ij} = S_{ji}$. Let D denote the degree matrix of S, which is a diagonal matrix and $D_{ii} = \sum_{j=1}^{m} S_{ij}$. Before propagating ranking functions on graph, we are given a initial set (possibly incomplete) of ranking functions obtained from some learning to rank algorithm, which is denoted using Y, $Y \in \mathbb{R}^{m \times d}$, and $Y_{i\cdot}$ (i-th row of Y) is the given ranking function of retrieval task t_i. We set $Y_{i\cdot}$ to a vector comprised of zeros if the ranking function for t_i is missing. The target value in our formulation is the propagation result of ranking functions, which is denoted by W, $W \in \mathbb{R}^{m \times d}$, and $W_{i\cdot}$ (i-th row of W) is the resulting (propagated) ranking function for retrieval task t_i.

We assume the initial set of ranking functions Y consists of a small portion of *strong* (sufficiently trained) ranking functions and a large portion of *weak/unknown* (insufficiently/not trained) ranking functions. Our goal is to fill in the *unknown* or improve the *weak* ranking functions using the *strong* functions based on the similarity amongst retrieval tasks, i.e., to propagate the ranking functions in Y via the similarity matrix S of the graph. We follow the classic consistency assumption of graph-based semi-supervised learning, nearby ranking functions are likely to be similar, and generalize the objective functions from previous studies. Formally, the generalized objective is written as follows.

$$\arg\min_{W} \sum_{i,j=1}^{m} S_{ij} d^2(W_{i\cdot}, W_{j\cdot}) + \sum_{i=1}^{m} \mu_i d^2(W_{i\cdot}, Y_{i\cdot}) \tag{2}$$

where $d()$ denotes some distance metric, and $d^2()$ simply denotes the square of a distance. The first term measures the *smoothness* of graph, i.e., two similar ranking functions should have small distance. The second term *penalizes* any changes to the initial ranking functions, i.e., the given *strong* ranking functions should not be overwritten significantly since these functions are deemed as correct. μ is a tuning parameter that balances the *smoothness* and *penalty*. In practice, μ can be a m-length vector, such that we can set a large overwriting penalty to the locations of *strong* ranking functions, and a zero/small penalty for the overwriting of *unknown/weak* ranking functions respectively. We shall later in this paper experimentally demonstrate that the performance of our proposed approach is not sensitive to the scale of μ.

The generalized objective is closely related to several classic graph-based semi-supervised learning methods. If $d(W_{i\cdot}, W_{j\cdot}) = |W_{i\cdot} - W_{j\cdot}|$ the objective reduces to GFHF (graph Laplacian). If $d(W_{i\cdot}, W_{j\cdot}) = |\frac{W_{i\cdot}}{\sqrt{D_{ii}}} - \frac{W_{j\cdot}}{\sqrt{D_{jj}}}|$ the objective reduces to LGC (normalized graph Laplacian).

With the objective function, the problem turns to selecting an appropriate distance metric. Though propagation on non-distribution data can be naturally handled by typical graph diffusion approaches, the propagation of distributions is difficult to solve. Similar problems involving probability interpolation have been encountered in mathematical literatures (McCann 1997; Agueh and Carlier 2011), topic modeling (Cai, Wang, and He 2009), clustering problem (Applegate et al. 2011), and also graphics applications (Bonneel et al. 2011). To address this issue, our algorithm was designed with a few criteria in mind. (1) The distance metric is preferred to be within the range from 0 to 1. (2) It allows interactions between adjacent histogram bins. (3) From the distance metric, we should be able to derive a simple or closed-form solution for the objective. A comparison of statistical distances found the Hellinger distance metric satisfies all the aforementioned criteria. The Hellinger distance, a type of f-divergence, is used to quantify the similarity between two probability distributions. Let a and b denote two distributions, the Hellinger metric in discrete format is defined as

$$H(a,b) = \frac{1}{\sqrt{2}}\sqrt{\sum_i \left(\sqrt{a_i} - \sqrt{b_i}\right)^2} \tag{3}$$

In addition there are other benefits brought by the Hellinger metric: (4) Unlike other statistical distances, the Hellinger metric is applicable to any distributions without pre-processing corrections, e.g., KL divergence prohibits zeros on the same location of both distribution which is corrected using the Laplace correction, (5) The Hellinger distance has shown to be robust to noise and skew-insensitive to imbalanced data (Cieslak et al. 2012; Goldberg et al. 2009). Substituting the Hellinger metric into the Eq.(2), our objective function is written as

$$\begin{aligned} \arg\min_{W} \quad & \frac{1}{2}\sum_{i,j=1}^{m} S_{ij} \sum_{k=1}^{d} \left(\sqrt{w_{ik}} - \sqrt{w_{jk}}\right)^2 \\ & + \frac{1}{2}\sum_{i=1}^{m} \mu_i \sum_{j=1}^{d} \left(\sqrt{w_{ij}} - \sqrt{y_{ij}}\right)^2 \end{aligned} \tag{4}$$

From the definition of the Hellinger distance metric, we see that it is required that all entries in a linear ranking function are (i) non-negative ($w_{ij} \geq 0, \forall i, j$) in order to keep the metric valid, and the entries in a function are (ii) sum-to-one ($\sum_{j=1}^{d} w_{ij} = 1, \forall i$) so as to preserve the nice properties of the Hellinger metric. However, in practice these two requirements rarely hold in a linear ranking function learned by some learn to rank algorithm. To address this issue, we adopt a simple scheme to preprocess linear ranking functions: (1) adding/subtracting a constant to make the entries range from zero to a positive number, and then (2) multiplying/dividing a constants so that the entries sum to one. Since our work only deals with distribution data, the above two manipulations do not affect the ranking orders. Recall that a ranking score is calculated using $r_i^j = w_j^T x_i$. For adding/subtracting a constant we have $r_i^j = (w_j + c)^T x_i = w_j^T x_i + c$, since $\sum_{j=1}^{d} w_{ij} = 1$. As for multiplying/dividing a constant, we have $r_i^j = c w_j^T x_i$. We see that both the shifting and normalization make an uniform change to all ranking scores, therefore, the ranking orders remain exactly the same.

Closed-form Solution

In the proposed learning objective, the cost function involves only one variable W to be optimized, which contains the ranking functions to be inferred. To recover W in closed form, we write the objective in Eq.(4) into matrix format.

$$\begin{aligned} \mathcal{Q}(W) \quad = \quad & \mathrm{tr}\{\frac{1}{2} W^{\circ\frac{1}{2}^T} (D - S) W^{\circ\frac{1}{2}} \\ & + \frac{1}{2}\mu (W^{\circ\frac{1}{2}} - Y^{\circ\frac{1}{2}})^T (W^{\circ\frac{1}{2}} - Y^{\circ\frac{1}{2}})\} \end{aligned} \tag{5}$$

where $\mathrm{tr}\{\}$ denotes the trace operation, and $()^{\circ\frac{1}{2}}$ the denotes element-wise square root of a matrix.

Since $W^{\circ\frac{1}{2}}$ and W have one-to-one correspondence, the optimization of $\mathcal{Q}$ with respect to W is equivalent to minimizing $\mathcal{Q}$ with respect to $W^{\circ\frac{1}{2}}$. Since $W^{\circ\frac{1}{2}}$ is continuous and our objective function is convex, we recover the minimum by simply setting the derivative w.r.t. $W^{\circ\frac{1}{2}}$ to zero.

$$\frac{\partial \mathcal{Q}}{\partial W^{\circ\frac{1}{2}}} = (D - S) W^{\circ\frac{1}{2}} + \mu (W^{\circ\frac{1}{2}} - Y^{\circ\frac{1}{2}}) = 0 \tag{6}$$

With simple linear algebra we have the global optimal W.

$$W = \left(\left(\frac{1}{\mu}(D - S) + I \right)^{-1} Y^{\circ\frac{1}{2}} \right)^{\circ 2} \tag{7}$$

where I denotes the identity matrix, and $()^{\circ 2}$ denotes the element-wise square of a matrix. The propagation of ranking functions is closely related to random walk on a graph, but here the *strong* ranking functions are viewed as "absorbing boundary" for the random walk. Our method differs from typical random walk in two main aspects, (i) we penalize the changes on the *strong* ranking functions, and (ii) our solution is an equilibrium state in terms of hitting time.

Empirical Evaluation

We in this section attempt to understand the strengths and relative performance of our distribution based ranking function propagation method, which we refer to in this section as `RFP` (Ranking Function Propagation). It is important to note that, to the best of our knowledge, our work is the first on ranking function propagation on distribution data. Therefore, we in our experiment can only compare with non-distribution propagation methods. In particular we wish to answer how well our method compares to the following state-of-the-art **baseline methods**:

1. `GFHF` (2003): A state-of-the-art label propagation method based on the harmonic function and the graph Laplacian.
2. `LGC` (2003): A label propagation method based on the graph smoothness and the normalized graph Laplacian.

The result shows that our propagation method significantly outperforms the two baseline methods. Given this, the next question naturally raised would be "*How good are the inferred ranking functions comparing to the ranking functions that are trained with sufficient data?*". To investigate this we explore the following two extreme scenarios:

3. `Lower Bound`: performance of the initial ranking functions without propagation. For *unknown* ranking functions, we simply use a random vector. An inferred ranking function should be at least better than a random guess.

4. Upper Bound: performance of all the ranking functions fully trained, which can be deemed as $1-$training errors, and is the upper bound for any propagation methods.

Learning to rank model. We use RankSVM (Chapelle 2007) to provide the initial ranking functions for the propagation methods. The pairwise constraints used for training in RankSVM are generated using labels (in the bird species dataset) or the ratings (in the movie poster dataset). The parameters in RankSVM are set as follows: the penalty constant C is set to 1, and the two options for Newton's method, i.e., the maximum number of linear conjugate gradients is 20, the stopping criterion for conjugate gradients is 10^{-3}.

Parameters. GFHF is nonparametric. For LGC and RFP, there is a parameter μ that balances the graph smoothness and overwriting penalty. In LGC, μ is set using cross validation. In RFP, we set the vector μ with the following values, 10 at the locations corresponding to *strong* (trained with abundant data) ranking functions, 0 at the locations corresponding to *unknown* (no training data) ranking functions, and 1 at the locations corresponding to *weak* (trained with little data) functions. We will also experimentally show that our propagation method is not sensitive to the scale of μ.

Evaluation measure. In our evaluation, ranking accuracy was assessed using a standard measure - normalized discounted cumulative gain (**NDCG**). We use binary ratings to calculate NDCG, i.e., rating is 1 if relevant, and 0 otherwise. To further investigate the properties of inferred ranking functions, since a linear ranking function can be viewed as a vector in the space, we calculate the **cosine similarity** between an inferred ranking function and the corresponding ranking functions that is fully trained using all data (upper bound).

Experiment 1. Acoustic Retrieval of Bird Species

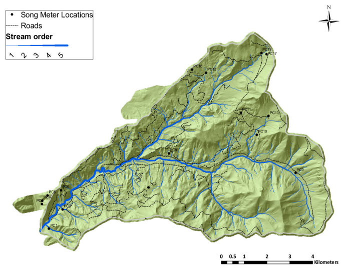
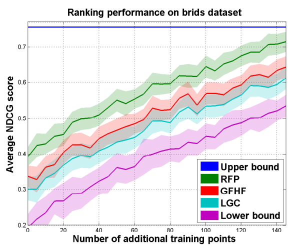
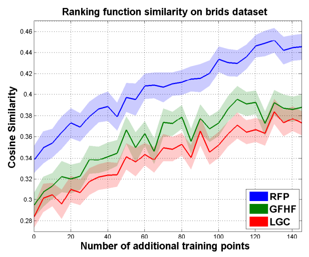


Figure 2: Song meter data collection locations in the H. J. Andrews Forest.

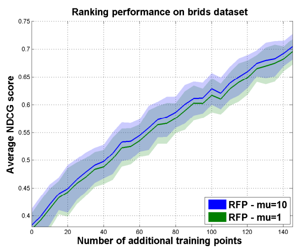
Datasets and Experimental Settings. The birds dataset (Briggs et al. 2013) consists of 645 ten-second audio recordings collected from the Cascade mountain range of Oregon (as shown in Figure 2). The raw audio signal is converted into a spectrogram (an image representation of the sound), by dividing it into frames, and applying the FFT to each frame. To extract features, the spectrogram is divided into 24 frequency bands, and we then summarize each band with statistics, including the mean, variance, skewness, kurtosis, min, max, and median. Finally, each audio recoding is represented using a 168-dimensional statistical spectrogram descriptor (SSD), which can be viewed as distribution data. There are 19 species of bird presented in the audio recordings, such as Brown Creeper, Pacific Wren. Each audio recording is paired with a set of species that are present. These species labels were obtained by listening to the audio and looking at spectrograms. Our goal is to learn 19 ranking functions to retrieve the 19 species of bird. We first construct a graph of the 19 nodes (species). The affinity matrix S is constructed by calculating the Hellinger similarity (i.e., one minus the Hellinger distance) amongst the prototypes (averaged audio spectrogram of a species) of the 19 bird species. We then propagate ranking functions on the graph.



(a) Average NDCG at top 20



(b) Cosine similarity between inferred ranking functions and the fully trained (deemed as the ground truth) functions



(c) Average NDCG w.r.t. different μ

Figure 3: Evaluation on birds dataset (standard deviation is denoted by shade).

Result and discussion. In each trial we randomly select 5 species of birds, and train a *strong* ranking function for each of them using adequate training data. The 5 *strong* ranking functions are used as the initial set of ranking functions, and then additional 150 (randomly selected) labeled audio recordings are gradually added to the training set, which are used to build new ranking functions, or improve the existing *weak* ones. At each step, we infer the *unknown* or *weak* ranking functions using the five graph propagation methods. The experiment is repeated for 100 times.

The mean (averaged over both the 19 retrieval tasks and 100 random trials) NDCG scores (at rank 20) are reported in Figure 3(a). We see that both the GFHF and LGC achieve significantly higher accuracy than Lower bound. This confirms the motivation and usefulness of ranking function propagation since it improves the retrieval performance with no additional training data. It can be observed that our distribution propagation method significantly outperforms the competing techniques, and approaches the Upper bound as the training set increases. Figure 3(b) shows the cosine similarity between the inferred ranking functions and the ranking functions that are fully trained with all data (deemed as the ground truth). The result also verifies the superior performance of our method. This demonstrates the effectiveness and necessity of the proposed graph diffusion method since the propagation of distributions differs from traditional propagation methods that were derived to propagate vectors of independent labels. By observing the performance comparison of our method w.r.t different values of μ shown in Figure 3(c), we can conclude that our method is not sensitive to the scale of parameter μ, since the ranking accuracy is about the same even when μ is significantly different (setting `Our-mu=1` denotes $\mu = 1$ for strong ranking functions and $\mu = 0.1$ for weak ones, `Our-mu=10` is ten times bigger).

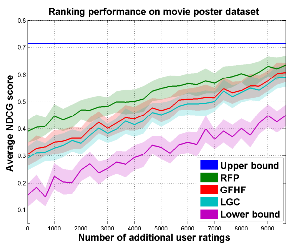
Experiment 2. Personalized Movie Ranking

Figure 4: Movie posters crawled from the web

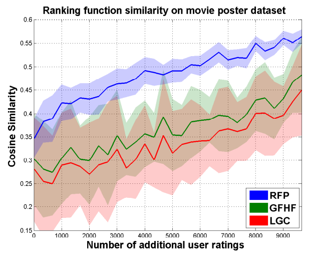
Datasets and Experimental Settings. The dataset used in this experiment is a collection of movie posters, which are crawled from the web using the links provided in HetRec2011-MovieLens-2K dataset (Cantador, Brusilovsky, and Kuflik 2011). Note that the goal of this experiment is not to evaluate our method as a recommender system, rather just to show the capability of our method in content-based personalized ranking. The HetRec-2K dataset contains 855,598 personal ratings on 10,197 movies from 2,113 users. On average, there are 404.921 ratings per user, and 84.637 ratings per movie. We explore an interesting scenario of personalized movie ranking – rank movies for each user based on the movie posters. This is inspired by a recent online article[1], which suggests that movie posters, as people's first impression of a movie, are closely correlated to people's taste of movies. Using the picture URLs provided in the dataset, we collected 9,893 movie posters from IMDb website (we will make the movie poster dataset publicly available soon). We calculate the color histogram ($4 \times 4 \times 4$ bins), such that each movie poster is represented by a 64-dimensional vector. According to the definition of color histogram, a representation of the distribution of colors in an image, the data can be viewed as distribution data. The 9,893 movie posters we crawled are associated with 835,189 ratings from 2,112 users. Our goal is to build a distinct ranking function for each user. The user affinity matrix S is constructed by simply normalizing the co-rating matrix of users, which consists of the counts that both users rated the same movies. In our evaluation, the user ratings (ranged from 0.5 to 5) are used to calculate the NDCG scores.

Result and discussion. We select 100 users in each random trial, and train a *strong* ranking function for each of them using all available ratings. These *strong* functions are used as the initial ranking functions to propagate, and then additional 10,000 randomly selected user ratings are added to the training data step by step, which are used either to train *unknown* ranking functions, or to improve the *weak* ones. At each step, we perform ranking functions propagation on the user graph, and the experiment is repeated for 50 times.

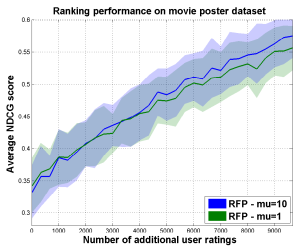
[1]http://www.boredpanda.com/movie-poster-cliches/



(a) Average NDCG at top 200



(b) Cosine similarity between inferred ranking functions and the fully trained (deemed as the ground truth) functions



(c) Average NDCG w.r.t. different μ

Figure 5: Evaluation on movie posters (standard deviation is denoted by shade).

The mean NDCG scores at rank 200, averaged over the 2,112 users and 50 random trials, are reported in Figure 5(a). It can be observed that the three propagation methods (i.e., `RFP`, `GFHF`, and `LGC`) significantly outperform the `Lower bound` baseline. This confirms our motivation of ranking function propagation, as it does increase the ranking accuracy without extra training data. Among the three propagation methods, we see that our distribution propagation achieves the highest ranking accuracy. The cosine similarity between the inferred ranking functions and the ground truth is presented in Figure 5(b), which shows that the ranking functions learned from our method are closer to the ground truth than the two competitors. It implies that distribution propagation differs from typical label propagation, which in turn demonstrates the necessity of our method when counts or frequencies are used as features. Figure 5(c) shows that the ranking performance only changes slightly though μ differs significantly. This verifies one of our claims that the proposed method is not sensitive to the scale of parameter μ.

Conclusion

In this paper we describe a method to propagate ranking functions on a graph and present an algorithm to perform distribution propagation. The method is used to address the issue caused by limited and imbalanced training data to propagate thousands of ranking functions on the same set of data. The proposed propagation approach is centered around distribution data and allows for the interactions between neighboring histogram bins. We derive a closed-form solution for the optimization, which makes our algorithm easy to implement. The experimental result on real world problems demonstrates the usefulness of our propagation method.

Acknowledgments

The authors gratefully acknowledge support of this research from ONR grants N00014-09-1-0712, N00014-11-1-0108 and NSF Grant NSF IIS-0801528.

References

Agueh, M., and Carlier, G. 2011. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis* 43(2):904–924.

Applegate, D.; Dasu, T.; Krishnan, S.; and Urbanek, S. 2011. Unsupervised clustering of multidimensional distributions using earth mover distance. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, 636–644.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research* 7:2399–2434.

Blum, A., and Chawla, S. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 19–26.

Bonneel, N.; van de Panne, M.; Paris, S.; and Heidrich, W. 2011. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph.* 30(6):158:1–158:12.

Briggs, F.; Huang, Y.; Raich, R.; Eftaxias, K.; Lei, Z.; Cukierski, W.; Hadley, S. F.; Hadley, A.; Betts, M.; Fern, X. Z.; et al. 2013. The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, 1–8. IEEE.

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

Cai, D.; Wang, X.; and He, X. 2009. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 105–112.

Cantador, I.; Brusilovsky, P.; and Kuflik, T. 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, 129–136.

Chapelle, O.; Schölkopf, B.; Zien, A.; et al. 2006. *Semi-supervised learning*, volume 2. MIT press Cambridge.

Chapelle, O. 2007. Training a support vector machine in the primal. *Neural Computation* 19(5):1155–1178.

Cieslak, D. A.; Hoens, T. R.; Chawla, N. V.; and Kegelmeyer, W. P. 2012. Hellinger distance decision trees are robust and skew-insensitive. *Data Min. Knowl. Discov.* 24(1):136–158.

Cossock, D., and Zhang, T. 2006. Subset ranking using regression. In *COLT*, 605–619.

Crammer, K., and Singer, Y. 2001. Pranking with ranking. In *NIPS*, 641–647.

Datta, R.; Joshi, D.; Li, J.; and Wang, J. Z. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.* 40(2):5:1–5:60.

Goldberg, A. B.; Zhu, X.; Singh, A.; Xu, Z.; and Nowak, R. 2009. Multi-manifold semi-supervised learning. In *AISTATS*, 169–176.

Hang, L. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94(10):1854–1862.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, 133–142.

Lawrence, N. D., and Jordan, M. I. 2005. Semi-supervised learning via gaussian processes. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press. 753–760.

McCann, R. J. 1997. A convexity principle for interacting gases. *advances in mathematics* 128(1):153–179.

Shashua, A., and Levin, A. 2002. Ranking with large margin principle: Two approaches. In *NIPS*, 937–944.

Xu, J., and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 391–398.

Yue, Y.; Finley, T.; Radlinski, F.; and Joachims, T. 2007. A support vector method for optimizing average precision. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 271–278.

Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; and Schölkopf, B. 2003. Learning with local and global consistency. In *NIPS*.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 912–919.

Zhu, X. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.