2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

A Harmonic Extension Approach for Collaborative Ranking

Da Kuang†, Zuoqiang Shi‡, Stanley Osher†, and Andrea L. Bertozzi†

†Department of Mathematics, University of California, Los Angeles (UCLA), Los Angeles, CA 90095, United States
‡Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, China
Email: {dakuang,sjo,bertozzi}@math.ucla.edu†, zqshi@math.tsinghua.edu.cn‡

Abstract—We present a new perspective on graph-based methods for collaborative ranking for recommender systems. Unlike user-based or item-based methods that compute a weighted average of ratings given by the nearest neighbors, or low-rank approximation methods using convex optimization and the nuclear norm, we formulate matrix completion as a series of semi-supervised learning problems, and propagate the known ratings to the missing ones on the user-user or item-item graph globally. The semi-supervised learning problems are expressed as Laplace-Beltrami equations on a manifold, or namely, *harmonic extension*, and can be discretized by a point integral method. Our approach, named *LDM (low dimensional manifold)*, does not impose a low-rank Euclidean subspace on the data points, but instead minimizes the dimension of the underlying manifold. It turns out to be particularly effective in generating rankings of items, showing decent computational efficiency and robust ranking quality compared to state-of-the-art methods.

1. Introduction

Recommender systems are crucial components in contemporary e-commerce platforms (Amazon, eBay, Netflix, etc.), Recommendation algorithms are commonly based on collaborative filtering, or "crowd of wisdom", and can be categorized into memory-based and model-based approaches. Memory-based approaches include user-based and item-based recommendation For example, for a user u, we retrieve the highly-rated items from the nearest neighbors of u, and recommend those items that have not been consumed by u. Memory-based methods are actually based on a graph, where a user-user or item-item similarity matrix defines the nearest neighbors of each user or item. In contrast, model-based methods are formulated as matrix completion problems which assume that the entire user-by-item rating matrix is low-rank,and the goal is to predict the missing ratings given the observed ratings.

Popular model-based methods such as regularized SVD minimize the sum-of-squares error over all the observed ratings. When evaluating the predictive accuracy of these algorithms, after obtaining a model on a training set, we evaluate the sum-of-squares error of its prediction on a separate test set in order to see how well it generalizes to unseen data. However, the measure for evaluating success in a practical recommender system is very different. What we care more about is whether the top recommended items for a user u will actually be "liked" by u. A more relevant evaluation measure in this context is the resemblance between the ranked list of top recommended items and the ranked list of observed ratings in the test set. This problem that places priority on the top recommended items rather than the accuracy of predicted absolute ratings is referred to as top-N recommendation, or more recently *collaborative ranking* [2, 4], and is our focus in this paper.

Here are some notations we will use. For a vector $x = [x_1, \cdots, x_m]^T$, we call $y = [x_{i_1}, x_{i_2}, \cdots, x_{i_r}]^T$ a *subvector* of length r by extracting the elements of x in the index set $\{i_1, \cdots, i_r\}$, where $i_1 < i_2 < \cdots < i_r$. For a matrix M, a vector x, integers i, j, and sets of row and column indices S, S', we use $M_{i,j}, M_{S,S'}, M_{:,j}, M_{S,j}, x_S$ to denote an entry of M, a submatrix of M, the j-th column of M, a subvector of the j-th column of M, and a subvector of x, respectively.

2. Harmonic Extension

We start with the matrix completion problem and formulate it as a series of semi-supervised learning problems, or in particular, harmonic extension problems on a manifold that can be solved by label propagation [6]. For each item, we want to know the ratings by all the users, and the goal of the semi-supervised learning problem is to propagate the known labels for this item (observed ratings) to the unknown labels on the user-user graph; and reversely, for each user, to propagate the known labels given by this user to the unknown labels on the item-item graph.

Consider a user-by-item rating matrix $A = (a_{ij}) \in \mathbb{R}^{m\times n}$, where rows correspond to m users, and columns correspond to n items. The observed ratings are indexed by the set $\Omega = \{(i, j) : \text{user } i \text{ rated item } j\}$. Let $\Omega_i = \{1 \le j \le m : (i, j) \in \Omega\}$, $1 \le i \le n$. Suppose there exists a "true" rating matrix A^* given by an oracle with no missing entries, which is not known to us, and $A|_\Omega = A^*|_\Omega$.

Without loss of generality, we assume that there exists a user manifold, denoted as $\mathcal{M}$, which consists of an infinite number of users. Each user is identified by an n-dimensional vector that consists of the ratings to n items. Thus, $\mathcal{M}$ is a submanifold embedded in $\mathbb{R}^n$. For the i-th item, we define the rating function $f_i : \mathcal{M} \to \mathbb{R}$ that maps a user into the rating of this item.

One basic observation is that for a fixed item, *similar users should give similar ratings*. This implies that the

function f_i, $1 \le i \le n$ is a *smooth* function on $\mathcal{M}$. Therefore, it is natural to find the rating function f_i by minimizing the following energy functional:

$$E(f_i) = \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f_i(u)\|^2 du, \tag{1}$$

where $\nabla_{\mathcal{M}} f(u)$ is the gradient at u defined on the manifold $\mathcal{M}$.

Let the set of m users in our user-by-item rating system be $U = \{u_j, 1 \le j \le m\}$, where u_j is the j-th row of A^* and $U \subset \mathcal{M}$ is a sample of $\mathcal{M}$. Let $U_i = \{u_j \in U : j \in \Omega_i\}$ be the collection of users who rate the i-th item. Then we compute the rating function f_i for all the users by minimizing the energy functional in (1):

$$\min_{f_i \in H^1(\mathcal{M})} \quad E(f_i) \quad \text{subject to:} \quad f_i(u_j)|_{U_i} = a_{ij}, \tag{2}$$

where H^1 is the Sobolev space. Hence, we need to solve the following type of optimization problem for n times.

$$\min_{f \in H^1(\mathcal{M})} \quad E(f) \quad \text{subject to:} \quad f(u)|_{\Lambda} = g(u), \tag{3}$$

where $\Lambda \subset \mathcal{M}$ is a point set.

To solve the constrained optimization problem (3), we use the Bregman iteration to enforce the constraint.
- Solve $f^{k+1} = \arg\min_f \; E(f) + \mu \|f - g + d^k\|^2_{L^2(\Lambda)}$, where $\|f\|^2_{L^2(\Lambda)} = \sum_{u \in \Lambda} |f(u)|^2$, and d^n is a function defined on Λ.
- Update d^k, $d^{k+1}(u) = d^k(u) + f^{k+1}(u) - g(u), \; \forall u \in \Lambda$.
- Repeat above process until convergence.

Using a standard variational approach, the solution to the above unconstrained optimization problem can be reduced to the following Laplace-Beltrami equation:

$$\begin{cases} \Delta_{\mathcal{M}} f(\mathbf{x}) - \mu \sum_{\mathbf{y} \in \Lambda} \delta(\mathbf{x} - \mathbf{y})(f(\mathbf{y}) - h(\mathbf{y})) = 0, \; \mathbf{x} \in \mathcal{M}, \\ \dfrac{\partial f}{\partial \mathbf{n}}(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial \mathcal{M}, \end{cases} \tag{4}$$

where δ is the Dirac-δ function in $\mathcal{M}$, $h = g - d^n$ is a given function on Λ, and $\mathbf{n}$ is the outer normal vector. That is to say, the function f that minimizes (3) is a harmonic function on $\mathcal{M} \backslash \partial \mathcal{M}$, and (4) is called the *harmonic extension* problem in the continuous setting.

2.1. Point Integral Method (PIM)

The Laplace-Beltrami operator in (4) can be approximated by an integral form with provable bound (see details in [5]), leading to an integral equation:

$$\frac{1}{t} \int_{\mathcal{M}} (f(\mathbf{x}) - f(\mathbf{y})) w_t(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} + \mu \sum_{\mathbf{y} \in \Lambda} w_t(\mathbf{x}, \mathbf{y})(f(\mathbf{y}) - h(\mathbf{y})) = 0, \tag{5}$$

where no derivatives appear and therefore it is easy to discretize over the point cloud. We notice that the closed form of the user manifold $\mathcal{M}$ is not known, and we only have a sample of $\mathcal{M}$, i.e. U. Assume that the point set U is uniformly distributed over $\mathcal{M}$. The integral equation can be discretized as follows:

$$\frac{|\mathcal{M}|}{m} \sum_{j=1}^{m} w_t(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j)) + \mu t \sum_{\mathbf{y} \in \Lambda} w_t(\mathbf{x}_i, \mathbf{y})(f(\mathbf{y}) - h(\mathbf{y})) = 0 \tag{6}$$

where $|\mathcal{M}|$ is the volume of the manifold $\mathcal{M}$.

We can rewrite (6) in the matrix form:

$$\boldsymbol{L}\boldsymbol{f} + \bar{\mu} \boldsymbol{W}_{:,\Lambda} \boldsymbol{f}_{\Lambda} = \bar{\mu} \boldsymbol{W}_{:,\Lambda} \boldsymbol{h}. \tag{7}$$

where $\boldsymbol{h} = (h_1, \cdots, h_m)$ and $\bar{\mu} = \frac{\mu t m}{|\mathcal{M}|}$. $\boldsymbol{L}$ is a $m \times m$ matrix which is given as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \tag{8}$$

where $\boldsymbol{W} = (w_{ij}), \; i, j = 1, \cdots, m$ is the weight matrix and $\boldsymbol{D} = \mathrm{diag}(d_i)$ with $d_i = \sum_{j=1}^{m} w_{ij}$. Our harmonic extension algorithm is summarized in Algorithm 1.

Remark. In the harmonic extension approach, we use a continuous formulation based on the underlying user manifold. And the point integral method is used to solve the Laplace-Beltrami equation on the manifold. If a graph model were used at the beginning, the natural choice for harmonic extension would be the graph Laplacian. However, it has been observed that the graph Laplacian is not consisitent in solving the harmonic extension problem [3], and PIM gives much better results.

Algorithm 1 Harmonic Extension

Input: Initial rating matrix A.
Output: Rating matrix R.

```
Set R = A.
repeat
    Estimate the weight matrix W = (w_ij) from the user set U (Algorithm 2).
    Compute the graph Laplacian matrix: L = D − W
    for i = 1 to n do
        repeat
            Solve the following linear systems
                L f_i + μ̄ W_{:,U_i} (f_i)_{U_i} = μ̄ W_{:,U_i} h_{U_i},
            where h = g_i − d^k.
            Update d^k,
                d^{k+1} = d^k + f^{k+1} − g_i
        until some stopping criterion for the Bregman iterations is satisfied
    end for
    r_ij = f_i(u_j) and R = (r_ij).
until some stopping criterion is satisfied
```

Algorithm 2 Estimating the weight matrix

Input: Incomplete rating matrix $R \in \mathbb{R}^{m\times n}$, number of nearest neighbors K

1: Generate binary rating matrix $R_B \in \mathbb{R}^{m\times n}$:

$$(R_B)_{j,j'} = \begin{cases} 1, & R_{j,j'} \text{ is not missing} \\ 0, & R_{j,j'} \text{ is missing} \end{cases}$$

2: Normalize each row of R_B such that $\|(R_B)_{j,:}\|_2 = 1$, $\forall j, 1 \le j \le m$

3: Build a *kd*-tree on the data points (rows) in R_B

4: Initialize a sparse matrix $\boldsymbol{W} \leftarrow \mathbf{0}^{m\times m}$

5: **for** $j = 1$ to m **do**

6: $\mathcal{N}_B \leftarrow$ The set of K approximate nearest neighbors of $(R_B)_{j,:}$, found by querying the *kd*-tree

7: **for** $j' \in \mathcal{N}_B$ $(j' \neq j)$ **do**

8: Set of co-rated items $C \leftarrow$ $\{i : R_{j,i}$ is not missing, and $R_{j',i}$ is not missing$\}$

9: $\boldsymbol{W}_{j,j'} \leftarrow \text{cosine}(R_{j,C}, R_{j',C})$

10: **end for**

11: **end for**

Output: Sparse weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times m}$

2.2. Low Dimensional Manifold (LDM) Interpretation

Here we emphasize another interpretation of our method based on the low dimensionality of the user manifold. A user is represented by an n-dimensional vector that consists of the ratings to n items, and the user manifold is a manifold embedded in $\mathbb{R}^n$, where n is usually a large number in the order of $10^3 \sim 10^6$. The intrinsic dimension of the user manifold is typically much smaller than n. Based on this observation, we use the dimension of the user manifold as a regularization, in contrast to the dimension of the user subspace in a matrix factorization method, to recover the rating matrix, which implies the following optimization problem:

$$\min_{\substack{X\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^n}} \quad \dim(\mathcal{M}), \tag{9}$$
$$\text{subject to:} \quad P_\Omega(X) = P_\Omega(A), \quad \mathcal{U}(X) \subset \mathcal{M}.$$

where $\dim(\mathcal{M})$ is the dimension of the manifold $\mathcal{M}$, $\mathcal{U}(X)$ is the user set corresponding to the rows of X, and P_Ω is the projection operator to Ω,

$$P_\Omega(X) = \begin{cases} x_{ij}, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases}$$

By referring to the theory in differential geometry, the optimization problems (1) and (9) are shown to be equivalent [3] which gives our model a geometric interpretation.

3. Estimating the Weight Matrix

If the underlying manifold $\mathcal{M}$ (the true rating matrix A^* given by an oracle in the discrete setting) were known, the n problems in (2) would be independent with each other and could be solved individually by (4). However, $\mathcal{M}$ is not known, and therefore we have to get a good estimate for the operator $\Delta_\mathcal{M}$ based on f_j's, or $\boldsymbol{W}$ based on $\boldsymbol{f}_j$'s.

Table 1: Statistics of the data sets in our experiments. N is the number of ratings in the training set for each user.

Data set		# users	# items	# ratings
MovieLens-100k	$N = 10$	943	1,682	100K
	$N = 20$	744	1,682	95K
MovieLens-1m	$N = 10$	6,040	3,706	1M
	$N = 20$	5,289	3,701	982K
	$N = 50$	3,938	3,677	924K
MovieLens-10m	$N = 10$	69,878	10,677	10M
	$N = 20$	57,534	10,675	9.7M
	$N = 50$	38,604	10,672	8.9M

The weight matrix $\boldsymbol{W}$ plays an important role in our algorithm as well as other graph-based approaches [6]. We employ the typical user-user or item-item graph with cosine similarity used in existing memory-based approaches for recommendation. However, we have made substantial changes to make the procedure efficient for large sparse rating matrices. Our algorithm for building the weight matrix is described in detail in Algorithm 2. Again, we consider the user-user graph without loss of generality.

First, as usual, we can only afford to compute and store a sparse nearest-neighbor weight matrix. To get the K nearest neighbors for a target user u, traditional algorithms in memory-based methods require computing the distances between u and every other user, and selecting the K closest ones, where most of the computation is wasted if $K << m$. In our algorithm, we first identify the nearest neighbors approximately, without computing the actual distances or similarities, and then compute the similarities between u and its nearest neighbors only. We use a binary rating matrix R_B that records "rated or not-rated" information (Algorithm 2, line 1-2), and determine the K nearest neighbors using an *kd*-tree based approximate nearest neighbor algorithm (line 3, line 5). That is to say, two users who have rated similar sets of movies are more likely to be considered to be in each other's neighborhood, regardless of their numeric ratings for those movies. Neither of the ways to build the *kd*-tree and to find nearest neighbors based on the tree are as precise as a naïve search; however, empirical results in the next section have shown that our approximate strategy does not compromise the quality.

Second, we extended the VLFeat package[1] to enable building a *kd*-tree from a sparse data matrix (in our case, R_B) and querying the tree with sparse vectors. *kd*-tree uses a space partitioning scheme for efficient neighbor search. For high-dimensional data, we employ the greedy way that chooses the most varying dimension for space partitioning at each step of building the tree, and the procedure terminates when each leaf partition has one data point. Thus, the

[1] `http://www.vlfeat.org/`

Table 2: Benchmarking results of ranking quality NDCG@10 and run-time.

	NDCG@10				Time (seconds)			
	SVD	LCR	AltSVM	LDM	SVD	LCR	AltSVM	LDM
MovieLens-1m, $N = 10$	0.6836	0.7447	0.6680	0.7295	844.4	254.2	3.6	61.1
MovieLens-1m, $N = 20$	0.6758	0.7428	0.6879	0.7404	843.3	437.3	6.8	52.3
MovieLens-1m, $N = 50$	0.6178	0.7470	0.7730	0.7527	730.5	1168.8	53.0	37.0
MovieLens-10m, $N = 10$	0.6291	0.6866	0.6536	0.7077	24913.4	4544.4	61.2	1496.3
MovieLens-10m, $N = 20$	0.6201	0.6899	0.7208	0.7213	14778.5	6823.5	275.4	1653.1
MovieLens-10m, $N = 50$	0.5731	0.6830	0.7507	0.7286	10899.1	14668.5	648.4	1295.0

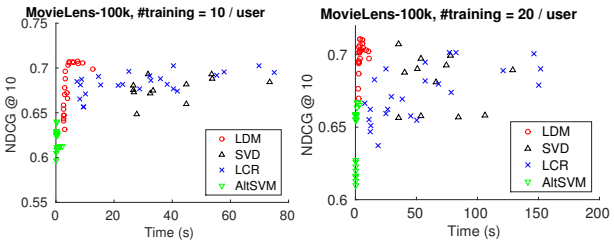

Figure 1: Ranking quality NDCG@10 vs. run-time under various hyperparameters.

complexity of building the tree is not exponential, contrary to common understanding; and the practical performance of *kd*-tree can be very efficient.

4. Experiments

We use three MovieLens data sets in our experiments, and randomly select N ratings for each user as the training set, and the other ratings were used for testing. Information regarding the selected data is summarized in Table 1.

We evaluate our proposed method LDM in terms of both run-time and ranking quality against SVD[2] and two state-of-the-art methods, namely local collaborative ranking (LCR) [2] and Alternating support vector machine (AltSVM) [4]. In Algorithm 1, which typically accounts for most ($\sim$ 95%) of the run-time in our method, we set $\mu = 1$ and run one inner iteration and one outer iteration only; and in Algorithm 2, we set $k = 64$ and $D = 256$. We evaluate the ranking quality by *normalized discounted cumulative gain* (NDCG) @K (see details in [1]).

Fig. 1 plots NDCG@10 against the run-time for several small data sets where performances of the four methods are compared under varying hyperparameters. Ideally, a good performance of a collaborative ranking method means producing higher NDCG@10 scores in less time. LDM achieves the highest NDCG@10 in a reasonable amount of time compared to the other methods. AltSVM is efficient but produces unsatisfactory ranking quality, which is also sensitive to its hyperparameters. For LCR, the ranking quality is acceptable but it takes considerably longer time, especially when the size of training set increases. On MovieLens-100k ($N = 20$), SVD and LDM achieve similar NDCG@10 scores but LDM costs much shorter run-time.

[2] `http://prea.gatech.edu`

Table 2 reports the run-time and NDCG@10 scores on the larger data sets where each method uses its best-performing hyperparameters. LDM does not achieve the highest NDCG@10 scores in every case, but produces robust ranking quality with decent run-time (except MovieLens-10m, $N = 50$). For LCR and SVD, the time cost increases dramatically on larger data sets. AltSVM achieves superior ranking quality when the number of training ratings N is large, but its performance is sensitive to the number of iterations, which in turn depends on the data set and the given tolerance parameter. We conclude that LDM is an overall competitive method that is efficient and robust to hyperparameters and the underlying data sets. Also, LDM has particular advantages when N is small, which we consider is a more difficult problem than the cases with richer training information.

Acknowledgements

The authors would like to thank NSF grants DMS-1737770, DMS-1417674; ONR grant N00014-16-1-2119; DOE grant DE-SC00183838. NSFC grants 11371220 and 11671005.

References

[1] D. Kuang, Z. Shi, S. Osher, and A. L. Bertozzi, "A harmonic extension approach for collaborative ranking," 2016, https://arxiv.org/abs/1602.05127.

[2] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *WWW 2014*.

[3] S. Osher, Z. Shi, and W. Zhu, "Low dimensional manifold model for image processing," accepted by *SIAM Journal on Imaging Sciences*, 2017.

[4] D. Park, J. Neeman, J. Zhang, S. Sanghavi, and I. Dhillon, "Preference completion: Large-scale collaborative ranking from pairwise comparisons," in *ICML 2015*.

[5] Z. Shi and J. Sun, "Convergence of the point integral method for Poisson equation on point cloud," accepted by *Research in the Mathematical Sciences*, 2017.

[6] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML 2003*.