Cross System Personalization and Collaborative Filtering by Learning Manifold Alignments

Bhaskar Mehta[1] and Thomas Hofmann[2]

[1]Fraunhofer IPSI, Damstadt 64293, Germany
[2]Darmstadt University of Technology, Darmstadt 64289, Germany
mehta@ipsi.fhg.de, hofmann@int.tu-darmstadt.de

Abstract. Today, personalization in digital libraries and other information systems occurs separately within each system that one interacts with. However, there are several potential improvements w.r.t. such isolated approaches. Investments of users in personalizing a system, either through explicit provision of information, or through long and regular use are not transferable to other systems. Moreover, users have little or no control over the information that defines their profile, since user profiles are deeply buried in personalization engines. Cross-system personalization, i.e. personalization that shares personalization information across different systems in a usercentric way, overcomes the aforementioned problems. Information about users, which is originally scattered across multiple systems, is combined to obtain maximum leverage. The key idea is that when a large number of users cross over from one system to another, carrying their user profiles with them, a mapping between the user profiles of the two systems can be discovered. In this paper, we discuss the use of manifold learning for the purpose of computing recommendations for a new user crossing over from one system to another.

1 Introduction

The World Wide Web provides access to a wealth of information and services to a huge and heterogeneous user population on a global scale. One important and successful design mechanism in dealing with this diversity of users is to *personalize* Web sites and services, i.e. to customize system contents, characteristics, or appearance with respect to a specific user. The ultimate goal is to optimize access to relevant information or products by tailoring search results, displays, etc. to a user's presumed interests and preferences. More specifically, this optimization may aim at, for example, increasing the efficiency of system usage or improving the quality and relevance of results. Given the huge and rapidly growing amount of data available online as well as an ever growing user population that uses the World Wide Web, the relevance of personalized access is likely to further increase in the future.

While most users will interact with different systems and sites on the Web, personalization most often occurs separately within each system. Each system independently builds up user profiles, for instance, by locally storing information about a user's likes and dislikes, interests, and further characteristics, and may then use this information to personalize the system's content and service offering. Such isolated approaches have two major drawbacks: Firstly, investments of users in personalizing a system either through explicit provision of information or through long and regular use are not transferable to other systems. Secondly, users have little or no control over the information that defines their profile, since user data are deeply buried in personalization engines running on the server side.

Cross system personalization [14] allows for sharing information across different information systems in a user-centric way and can overcome the aforementioned problems. Information about users, which is originally scattered across multiple systems, is combined to obtain maximum leverage and reuse of information. Previous approaches to cross system personalization [15] rely on each user having a *unified profile* which different systems can understand. The unified profile will contain facets modeling aspects of a multidimensional user. The basis of *understanding* in this approach is of a semantic nature, i.e. the semantics of the facets and dimensions of the unified profile are known, so that the latter can be aligned with the profiles maintained internally at a specific site. The main challenge in this approach is to establish some common and globally accepted vocabulary and to create a standard every system will comply with. Without such a convention, the exact mapping between the unified user profile and the system's internal user profile would not be known.

Machine learning techniques provide a promising alternative to enable cross system personalization without the need to rely on accepted semantic standards or ontologies. The key idea is that one can try to learn dependencies between profiles maintained within one system and profiles maintained within a second system based on data provided by users who use both systems and who are willing to share their profiles across systems – which we assume is in the interest of the user. Here, instead of requiring a common semantic framework, it is only required that a sufficient number of users cross between systems and that there is enough regularity among users that one can learn within a user population, a fact that is commonly exploited in social or *collaborative filtering* [20].

2 Automatic Cross System Personalization

For simplicity, we consider a two system scenario in which there are only two sites or systems denoted by A and B that perform some sort of personalization and maintain separate profiles of their users; generalization to an arbitrary number of systems is relatively straightforward and is discussed later. We assume that there is a certain number of c common users that are known to both systems. For simplification, we assume that the user profiles for a user u_i are stored as vectors $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\mathbf{y}_i \in \mathcal{Y} \subseteq \mathbb{R}^m$ for systems A and B, respectively. Given the profile $\mathbf{x}_i$ of a user in system A, the objective is to find the profile $\mathbf{y}_i$ of the same user in system B, so formally we are looking to find a mapping

$$F_{AB} : \mathbb{R}^n \to \mathbb{R}^m, \qquad \text{s.t.} \quad F_{AB}(\mathbf{x}_i) \approx \mathbf{y}_i \tag{1}$$

for users u_i. Notice that if users exist for which profiles in both system are known, i.e. a training set $\{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \ldots, l\}$, then this amounts to a standard supervised learning problem. However, regression problems typically only involve a single (real-valued) response variable, whereas here the function F_{AB} that needs to be learned is *vector-valued*. In fact, if profiles store say rating information about products or items at a site, then the dimensionality of the output can be significant (e.g. in the tens of thousands). Moreover, notice that we expect the outputs to be highly correlated in such a case, a crucial fact that is exploited by recommender systems. For computational reasons it is inefficient and often impractical to learn independent regression functions for each profile component. Moreover, ignoring inter-dependencies can seriously deteriorate the prediction accucracy that is possible when taking such correlations into account. Lastly, one also has to expect that a large fraction of users are only known to one system (either A or B). This brings up the question of how to exploit data without known correspondence in a principled manner, a problem generally referred to as *semi-supervised learning*. Notice that the situation is symmetric and that unlabeled data may be available for both systems, i.e. sets of vectors $\mathbf{x}_i$ without corresponding $\mathbf{y}_i$ and vice versa. In summary, we have three conceptual requirements from a machine learning method:

– Perform vector-valued regression *en bloc* and not independently
– Exploit correlations between different output dimensions (or response variables)
– Utilize data without known correspondences

In addition, the nature of the envisioned application requires:

– Scalability of the method to large user populations and many systems/sites
– Capability to deal with missing and incomplete data

There are some recent learning methods that can be utilized for vector-valued regression problem, but some of them do not fulfill the above requirements. Kernel dependency estimation [21] (KDE) is a technique that performs kernel PCA [19] on the output side and then learns independent regression functions from inputs to the PCA-space. However, KDE can only deal with unlabeled data on the output side and requires to solve computationally demanding pre-image problems for prediction [1]. Another option is Gaussian process regression with coupled outputs [12]. Here it is again difficult to take unlabeled data into account while preserving the computational efficiency of the procedure. The same is true for more traditional approaches like Multi-Layer-Perceptrons with multiple outputs. Instead of using regression methods, we thus propose the use of *manifold learning* in this context. Manifold learning methods generalize linear dimension reduction techniques that have already been used successfully in various ways for collaborative filtering. Moreover, they are usually motivated in an unsupervised setting that can typically be extended to semi-supervised learning in a rather straightforward manner. More specifically, we propose to use the *Laplacian Eigenmaps*[3] and *Locally Linear Embedding* (LLE)[18] approaches as our core method. LLE constructs a low-dimensional data representation for a given set of data points by embedding the points in a way that preserves the local (affine) geometry. Compared to other manifold learning and non-linear dimension reduction algorithms, such as Sammon's MDS [16] or Isomap [6], the LLE approach is computationally attractive and highly scalable, since it only relies on distances within local neighborhoods. Moreover, as presented in [9], constrained LLE (CLLE) can be utilized to learn mappings between two vector spaces by semi-supervised alignment of manifolds. The former work also provides empirical evidence that CLLE can outperfom standard regression methods. The key idea is to embed user profiles from different systems in low-dimensional manifolds such that profiles known to be in correspondence (i.e. profiles of the same user) are mapped to the same point. This means the manifolds will be aligned at correspondence points. A more general version of CLLE has been derived in [10], which takes the Laplacian Eigenmap approach [3] as the starting point. In the next section, we will provide more detail on these methods.

3 Non Linear Dimensionality Reduction and Manifold Alignment

3.1 Laplacian Eigenmaps

Suppose we are given l data points in $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^n \colon i = 1, \ldots, l\}$. When the data lie approximately on a low-dimensional manifold embedded in the n-dimesional Euclidean space, manifold learning methods such as Laplacian Eigenmaps [3], Hessian Eigenmaps [7], Isomap [6] or locally linear embeddings [18] can be used to recover the manifold from a sample $\mathcal{S}$. We pursue the Laplacian Eigenmap approach, which has been used sucessfully in semi-supervised learning [10] and for which rigorous convergence results exists in the large sample limit [11].

The starting point in Laplacian Eigenmaps is the construction of a weighted graph whose nodes are the sample points and whose edges connect the nearest neighbors of each node. Neighborhoods may consist of the k-nearest neighbors of a sample point or the set of all points that are within an ϵ-ball. We write $i \sim j$ as a shorthand for sample points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are neighbors. The weights W_{ij} between neighbors are usually assumed to be non-negative and symmetric, $W_{ij} = W_{ji} \geq 0$ and are summarized in an affinity matrix $\mathbf{W}$. There are several alternatives on how to define these weights when starting from a vector-valued representation over $\mathbb{R}^n$, one popular choice being the Gaussian kernel,

$$W_{ij} \equiv \exp\left[-\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2\right] , \tag{2}$$

where $\beta > 0$ is a suitably chosen bandwidth parameter. Another choice is to compute weights based on a local affine approximation over neighbors, as discussed in the following subsection on LLE.

The heart of the Laplacian Eigenmap approach is the generalized graph Laplacian $\mathbf{L}$ defined as,

$$\mathbf{L} = (L_{ij})_{i,j=1}^{n}, \quad L_{ij} = \begin{cases} \sum_{j\sim i} W_{ij}, & \text{if } i = j \\ -W_{ij}, & \text{if } i \sim j \\ 0, & \text{otherwise}\,. \end{cases} \tag{3}$$

An Laplacian Eigenmap is a function $f : \mathcal{S} \to \mathbb{R}$ for which $\mathbf{L}f = \lambda f$ and $\|f\|^2 = 1$, where we think of f as a vector of function values for convenience. Moreover, in order to remove the trivial solution with $\lambda = 0$ one can add the constraints $(1,\ldots,1)f = \sum_{i=1}^{l} f_i = 0$. It can be shown that the eigenmap corresponding to the smallest eigenvalue $\lambda > 0$ minimizes the criterion

$$f^T\mathbf{L}f = \sum_{i,j} W_{ij}(f_i - f_j)^2\,. \tag{4}$$

The eigenmaps corresponding to the d smallest eigenvalues span a d-dimensional coordinate system on the low-dimensional data manifold.

In the case of semi-supervised learning one may utilize $f^T\mathbf{L}f$ as a regularizer and combine it with supervised information about target values t_i that may be available at some subset $\mathcal{S}' \subseteq \mathcal{S}$ of the nodes of the graph to define the regularized solution (cf. [2])

$$f^* = \arg\min_f \sum_{\mathbf{x}_i \in \mathcal{S}'} (f_i - t_i)^2 + \lambda f^T\mathbf{L}f\,. \tag{5}$$

3.2 Aligned Manifold Learning

Consider now the case where two sets of points are given $\mathcal{S}_x \equiv \{\mathbf{x}_i \in \mathbb{R}^n : i = 1,\ldots,l_x\}$ and $\mathcal{S}_y \equiv \{\mathbf{y}_j \in \mathbb{R}^m : i = 1,\ldots,l_y\}$ where we assume without loss of generality that the first $l \le \min\{l_x, l_y\}$ points are in correspondence. In the case of cross system personalization, $\mathbf{x}_i$ will denote a user profile in system A, $\mathbf{y}_j$ will denote a user profile in system B and $\mathbf{x}_i \leftrightarrow \mathbf{y}_i$ for users u_i, $i = 1,\ldots,l$, who are known in both systems. We will separately construct graphs $\mathcal{G}_x$ on $\mathcal{S}_x$ and $\mathcal{G}_y$ on $\mathcal{S}_y$ in order to find low-dimensional embeddings of the points in $\mathcal{S}_x$ and $\mathcal{S}_y$, respectively. In addition, we will follow the approach in [10] and utilize the correspondence information to enforce that embeddings of user profiles for the same user are close to one another. To that extend we compute a simultaneous embedding f of $\mathcal{S}_x$ and g of $\mathcal{S}_y$ by minimizing the objective

$$C(f,g) = \sum_{i=1}^{l}(f_i - g_i)^2 + \lambda\left(f^T\mathbf{L}_x f + g^T\mathbf{L}_y g\,.\right) \tag{6}$$

More specifically, in order to deal with simultaneous re-scaling of f and g, one minimizes the Rayleigh quotient

$$\tilde{C}(f,g) = \frac{C(f,g)}{f^Tf + g^Tg}\,. \tag{7}$$

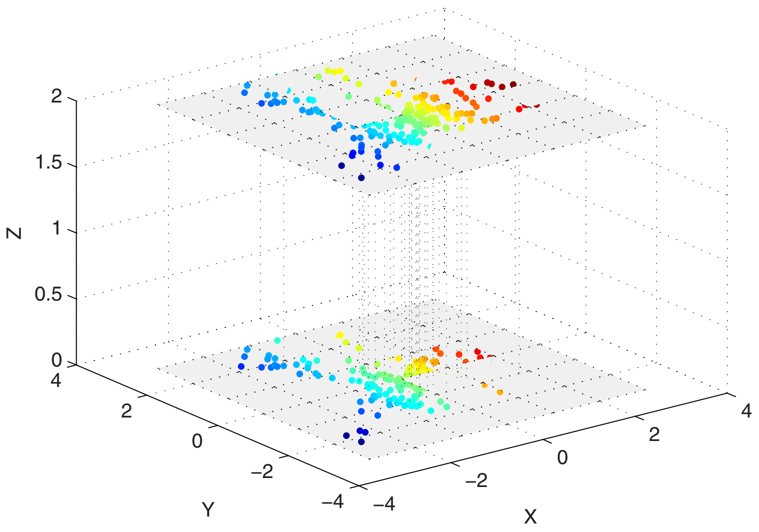

Fig. 1. Aligned 2D manifolds for two subsets of MovieLens dataset. The vertical lines show the points which are in correspondance.

By defining the combined graph $\mathcal{G} \equiv \mathcal{G}_x \cup \mathcal{G}_y$ with Laplacian $\mathbf{L}$ and combined functions $h = (f^T, g^T)^T$ the above objective can be rewritten as

$$\tilde{C}(h) = \frac{h^T \mathbf{H} h}{h^T h}\,, \quad \text{where } \mathbf{H} \equiv \lambda \mathbf{L} + \begin{pmatrix} \mathbf{U}^{nn} & \mathbf{U}^{nm} \\ \mathbf{U}^{mn} & \mathbf{U}^{mm} \end{pmatrix} \tag{8}$$

and $\mathbf{U}^{nm} \in \mathbb{R}^{n \times m}$ is diagonal with $U^{nm}_{ii} = 1$ for $1 \leq i \leq l$ and 0 otherwise. Again, a solution is obtained as before by finding the eigenvectors of the matrix $\mathbf{L}$.

One can also enforce the embeddings of points in correspondence to be the same on both manifolds [10]. In this case, one identifies the first l points in $\mathcal{S}_x$ and $\mathcal{S}_y$, resulting in a combined graph $\mathcal{G}$ with $l_x + l_y - l$ nodes with a combined weight matrix. Notice that weights between pairs of nodes with indices $1 \leq i, j \leq l$ are simply given by the sum of the weights from $\mathcal{G}_x$ and $\mathcal{G}_y$. Introducing functions h one then minimizes

$$\tilde{C}(h) = \frac{h^T \mathbf{L} h}{h^T h}\,, \quad \text{s.t.} \sum_i h_i = 0\,. \tag{9}$$

3.3 Locally Linear Embedding

One way to define the weights W_{ij} for neighboring nodes in the graph is to compute them based on a local affine approximation. This idea has originally presented in the context of the Locally Linear Embedding (LLE) method [18]. Its use as a preprocessing step in conjunction with Laplacian Eigenmaps has been proposed in [10]. For a sample of l data points $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^n \colon i = 1, \ldots, l\}$, LLE proceeds as follows:

- For each data point $\mathbf{x}_i$, compute the k nearest neighbors in $\mathcal{S}$ which are closest to $\mathbf{x}_i$ in Euclidean distance.
- Compute for each $\mathbf{x}_i$ the optimal approximation weights for an affine local regression over the neighbors. This is equivalent to approximating the nonlinear manifold at $\mathbf{x}_i$ by the linear hyperplane that passes through the neighboring points. This step of the algorithm amounts to solving a quadratic optimization problem:
$$W^*_{ij} = \arg\min_W |\mathbf{x}_i - \sum_{j \sim i} W_{ij}\mathbf{x}_j|^2 \,, \text{s.t.} \sum_j W_{ij} = 1 \,, \tag{10}$$
where $j \sim i$ indicates that $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$ (notice that the relation is in general not symmetric).
- Finally, a low-dimensional representation $\hat{\mathbf{x}}_i$ is computed by solving the minimization problem
$$\hat{\mathbf{X}}^* = \arg\min_{\hat{\mathbf{X}}} \sum_i \|\hat{\mathbf{x}}_i - \sum_{j \sim i} W_{ij}\hat{\mathbf{x}}_j\|^2 \tag{11}$$
This can be shown to be equivalent to an eigenvector decomposition problem involving the matrix
$$M = (I - W^*)^T (I - W^*) \tag{12}$$
where I is the $l \times l$ identity matrix. The bottom $d + 1$ eigenvectors of M (excluding the smallest, which is 1) form a co-ordinate system for the low dimensional data manifold.

While the LLE algorithm can be used in its own right for manifold learning, we have employed it here to compute the affinity matrix for the Laplacian Eigenmap method. Note that the matrix $\mathbf{L}^* = I - W^*$ corresponds to the graph Laplacian $\mathbf{L}$ (defined in eq. 3) for a graph with $\sum_j W_{ij} = 1$ for all graph nodes. Also note that the graph Laplacian thus formed is not symmetric and the weights can be negative. Multiplying $\mathbf{L}^*$ with its transpose gives a symmetric matrix M. [3] explains that under some conditions, the matrix M is approximately the same as $\mathbf{L}^2$, which has the same eigenvectors as $\mathbf{L}$. It has been shown in [10] that the matrix M can be substituted for the graph Laplacian $\mathbf{L}$ in the aligned manifold method.

3.4 Reconstructing Points from Alignments

The remaining problem we would like to discuss is how to map a point on the low-dimensional manifold back into the original data space. This is particularly relevant in the context of manifold alignment, where one ultimately may want to realize a mapping from $\mathbb{R}^n \to \mathbb{R}^m$. After mapping a point $\mathbf{x} \in \mathbb{R}^n$ to a k-dimensional representation $\hat{\mathbf{x}}$, we would thus like to compute an approximation $\mathbf{y} \in \mathbb{R}^m$ by finding a pre-image to $\hat{\mathbf{y}}$ and identifying $\hat{\mathbf{y}} = F(\hat{\mathbf{x}})$. The next section explains the method used to compute the pre-images.

4 The Manifold Alignment Collaborative Filtering Algorithm

Manifold alignment using non-linear dimensionality reduction has the promise of a fast and effective supervised learning technique for the correspondence problem. It has been reported that dimensionality reduction techniques are effective for k-NN algorithms used typically for collaborative filtering[17]. *Non Linear dimensionality Reduction* (NLDR) techniques in turn have performed better than linear dimensionality reduction techniques like Factor Analysis and PCA[9]. Therefore we expect manifold alignment for the purposes of cross system personalization to be an effective approach. The algorithm essentially works as a k-NN algorithm as well. After projecting the user profile vectors from two (or more) systems on a low dimension manifold, we are able to find nearest neighbors based on distance measures like Euclidean distance. The additional constraint of aligning profiles belonging to the same user aligns the two submanifolds and helps in finding more effective neighborhoods. Our algorithm has 4 steps, the first 3 of which form the manifold projection phase (see *Algorithm 1*), and the last step does the pre image computation and is outlined in *Algorithm 2.* Our algorithm assumes two datasets $\mathcal{X}$ and $\mathcal{Y}$ of sizes $n_{\mathcal{X}}$ and $n_{\mathcal{Y}}$ with c common users and is as follows:

1. **Neighborhood identification:** For each point $\mathbf{x}_i \in \mathcal{X}$, we find the k-nearest neighbors. NLDR techniques usually use Euclidean distance to identify the nearest points. In our setting, data is sparse, therefore Euclidean distance on pure data is not neccesarily effective unless missing data is imputed. Options here include mean imputation (with item mean), measuring distance only on commonly-voted items, or using a distance based on a similarity measure like Pearson's correlation coefficient. This procedure also has to be repeated for $\mathbf{y}_i \in \mathcal{Y}$. Note that choosing exactly k-nearest neighbors for every node may result in a graph Laplacian thats not symmetric. Using LLE, one selects *exactly* k neighbors, while for the Laplacian one does not impose this constraint. As a result, the neighborhood of some points in the Laplacian Eigenmaps method can be much larger than k. This usually shows the importance of a node and is similar to the notion of *authority* nodes in the HITS algorithm[13].
2. **Calculate Affinity Matrix:** After the k nearest neighbors have been identified for every point, an affinity weight with every neighbor has to be computed. Options here include an affine decomposition (like in LLE), an exponential weight (aka the heat kernel used in Laplacian Eigenmaps) based either on euclidean distance or on a similarity measure like Pearson's correlation.

$$W_{ij} = \exp\left[-\beta\|\mathbf{x}_i - \mathbf{x}_j\|^2\right] \; or \tag{13}$$

$$W_{ij} = \exp\left[-\beta\|1 - correlation(\mathbf{x}_i, \mathbf{x}_j)\|\right] \; or \tag{14}$$

$$W_{ij} = 1 \; \text{ or} \tag{15}$$

$$W_{ij} = \arg\min_{W} |\mathbf{x}_i - \sum_{j\sim i} W^*_{ij}\mathbf{x}_j|^2 \,, \text{s.t.} \sum_{j} W^*_{ij} = 1 \tag{16}$$

Algorithm 1. ComputeManifold-NLDR $(\mathcal{X}, \mathcal{Y}, c, k, d)$

Input: Matrices $\mathcal{X}, \mathcal{Y}$ with the first c columns aligned. k is the number of neighbors, d is the dimensionality of the manifold.

```
1: Impute missing values with mean item votes respectively for $\mathcal{X}$ and $\mathcal{Y}$ to get $\mathcal{X}_{norm}, \mathcal{Y}_{norm}$.
2: Calculate adjacency matrices $A_{\mathcal{X}}, A_{\mathcal{Y}}$ for graphs representing $\mathcal{X}_{norm}, \mathcal{Y}_{norm}$ by choosing $k$-nearest neighbors for every $\mathbf{x}_i \in \mathcal{X}$, $\mathbf{y}_i \in \mathcal{Y}$.
   $A_{\mathcal{X}}(i,j) = \begin{cases} 1, & \text{if } i \sim j \\ 0, & \text{otherwise}. \end{cases}$
3: Compute reconstruction weights $W_{\mathcal{X}}, W_{\mathcal{Y}}$.
4: Compute the graph Laplacians $\mathbf{L}_{\mathcal{X}}, \mathbf{L}_{\mathcal{Y}}$ from the Weight Matrices as defined in equation 3. For constrained LLE, use $\mathbf{L}^*_{\mathcal{X}} = (I - W_{\mathcal{X}})^T (I - W_{\mathcal{X}})$, etc.
5: Compute $\mathbf{L}_{\mathcal{XY}} = \begin{bmatrix} \mathbf{L}^{cc}_{\mathcal{X}} + \mathbf{L}^{cc}_{\mathcal{Y}} & \mathbf{L}^{cs}_{\mathcal{X}} & \mathbf{L}^{cs}_{\mathcal{Y}} \\ \mathbf{L}^{sc}_{\mathcal{X}} & \mathbf{L}^{ss}_{\mathcal{X}} & 0 \\ \mathbf{L}^{sc}_{\mathcal{Y}} & 0 & \mathbf{L}^{ss}_{\mathcal{Y}} \end{bmatrix}$
   $c$ represents the points in alignment, while $s$ represents the *single* points.
6: Find the low dimensional manifold $\mathbf{H}$ for the matrix $\mathbf{L}_{\mathcal{XY}}$. $\mathbf{H}$ has a dimensionality of $(n_{\mathcal{X}} + n_{\mathcal{Y}} - c) \times d$.
```

Output: Low dimensional manifold $\mathbf{H}$

In our experiments, we use the similarity measures defined in eq. 14. Finally, the Laplacians $\mathbf{L}_{\mathcal{X}}, \mathbf{L}_{\mathcal{Y}}$ of the graphs characterized by affinity matrices for $\mathcal{X}$ and $\mathcal{Y}$ are computed.

3. **Compute points on manifold:** This is usually done by solving an eigenvalue problem, and finding the eigenvectors of the Laplacian $\mathbf{L}$ (or $\mathbf{L}^*$ in case of LLE). For points in alignment, a modified eigenvalue problem has to be solved: A joint graph of the two datasets is formed and the eigenvectors of this Laplacian matrix $\mathbf{L}_{\mathcal{XY}}$ are computed (see eq. 8). The only parameter here is the dimensionality of the manifold (the number of eigenvectors that are chosen).
4. **Compute preimages for points not in correspondence:** In this step, neighborhoods for points not in correspondence are formed in a manner similar to the first step. The normal method to follow here is to do find the nearest neighbors (based on Euclidean distance) and compute a weight distribution over this neighborhood. We do this in the following manner: For a point $\mathbf{x}_i \in \mathcal{X}$ with $i > c$ and manifold coordinates $\hat{\mathbf{x}}_i$ we first identify a set of k nearest neighbors $\hat{\mathbf{y}}_r$ on the manifold among the points that are images of points in $\mathcal{Y}$, resulting in some set of image/pre-image pairs $\{(\mathbf{y}_r, \hat{\mathbf{y}}_r)\}$. We then compute the optimal affine combination weights w_r that optimally reconstruct $\hat{\mathbf{x}}_i \approx \sum_r w_r \hat{\mathbf{y}}_r$. Then the pre-image prediction is given by $F(\mathbf{x}_i) = \sum_r w_r \mathbf{y}_r$. Similarly, we can compute an inverse map by exchanging the role of the $\mathbf{x}_i$ and $\mathbf{y}_j$. Notice that one can also generalize this for arbitrary new samples $\mathbf{x} \in \mathbb{R}^n$ by generalizing the manifold mapping $\mathbf{x} \mapsto \hat{\mathbf{x}}$ to new points, which can be done along the lines presented in [4].

Algorithm 2. ComputePreimage $(\mathbf{H}, c, n_{\mathcal{X}}, n_{\mathcal{Y}}, k, \mathcal{X}_{norm}, \mathcal{Y}_{norm}, n_{\mathcal{X}}, n_{\mathcal{Y}})$

Input: Matrix $\mathbf{H}$ of length $(n_{\mathcal{X}} + n_{\mathcal{Y}} - c)$ representing the aligned manifold with c points overlapping between manifolds of $\mathcal{X}$ and $\mathcal{Y}$. k is the number of neighbors. $\mathbf{H}_M$ has the first c points representing the overlapping users. The next $n_{\mathcal{X}} - c$ points represents single points of $\mathcal{X}$ and the last $n_{\mathcal{Y}} - c$ points represent the single points of $\mathcal{Y}$. $\mathbf{H}(i)$ denotes the i^{th} d-dimensional point on the manifold.

```
1: Extract submanifold $\mathbf{H}_{\mathcal{Y}}$ by combining the first $c$ and the last $n_{\mathcal{Y}} - c$ points of $\mathbf{H}$.
2: for $i = (c+1)$ to $(n_{\mathcal{X}})$ do
3:    $\hat{\mathbf{x}}_\mathbf{i} \leftarrow \mathbf{H}(i)$
4:    Compute the $k$ nearest neighbors $\hat{\mathbf{y}}_r$ of $\hat{\mathbf{x}}_\mathbf{i}$ on the sub-manifold $\mathbf{H}_{\mathcal{Y}}$. Let $\mathbf{y}_r$
      represent the preimage of $\hat{\mathbf{y}}_r$ in $\mathcal{Y}_{norm}$.
5:    Compute affine weights $\mathbf{W}^* = (\mathrm{w}_r)_{r=1}^k$ for the neighborhood.
6:    Compute Preimage prediction $F(\hat{\mathbf{x}}_\mathbf{i}) = \sum_r \mathrm{w}_r \mathbf{y}_r$.
7:    $\hat{\mathcal{X}}_\mathbf{s}(i - c) = F(\hat{\mathbf{x}}_\mathbf{i})$
8: end for
9: Repeat above procedure by exchanging $\mathcal{X}$ and $\mathcal{Y}$ to compute preimages for single
   points of $\mathcal{Y}$.
```

Output: Preimages $\hat{\mathcal{X}}_\mathbf{s}, \hat{\mathcal{Y}}_\mathbf{s}$

5 Evaluation

The manifold algorithm seeks to predict the ratings of a user who has not rated even a single item on the current system so far. In this scenario, truly nothing is known about the active user. The best rating prediction that a system can provide is the popularity vote based on the mean votes of every item. The items with the highest mean votes are then recommended to the user. Our algorithm seeks to do better than this non-personalized recommendation. On the other extreme, given all the data for both systems, the best possible rate prediction could be calculated if all the data was known to one system. In this scenario, a SVD or Pearson's correlation based algorithm could compute the predictions (which serve as the gold standard for us). In order to be useful, our algorithm should perform better than predictions from the popularity vote and perform as close as possible to the gold standard.

5.1 Dataset and Evaluation Scheme

We chose the Movielens[1] data with 100,000 votes for the purposes of our evaluation. This data set consists of rates in 1682 items by 944 different users. This data is quite sparse ($\sim$ 6%) as is typically for user ratings. We split the data into two subsets X and Y by spiliting the movie ratings for all users (e.g. two matrices 840×944 and 842×944). In principle the overlap between datasets can be varied anging from no overlap to all items overlapping. While in the earlier case, the movie ratings are effectivly spilt into half, the complete data is

[1] http://www.grouplens.org/

available to both systems in the second case. However, in real world scenarios, item overlaps are very small. Therefore we chose a random 5% from the item-set as an overlap. The other free parameter is the number of users set to be in correspondence, which we vary from 0 to 800. The last 144 users form the test set for our evaluations. We randomly choose the test set and the item set for every run of the algorithm. Individual NLDR methods(LLE and Laplacian) have other parameters which need to be varied in order to judge their effect. There parameters are (a) the dimensionality of the manifold, (b) the size of the neighborhood for the adjacency matrix, and (c) the size of the neighborhood for the user profile reconstruction. Additionally, the Laplacian Eigenmap method has a free parameter β which can take any real value. In our experiments, we have varied the parameter and present the results for the optimized values. Further increases in neighborhood sizes offers some advantage, but at a much increased computational cost. We have chosen these values: $k = 36$, $d = 6$, and size of neighborhood on the manifold $k_1 = 55$. In addition, we choose different values of β, namely $0, 0.4$ and 4.

Evaluation Metrics

1. Mean Average Error $= \frac{1}{m}|p_v - a_v|$, where p_v is the predicted vote and a_v is the actual vote. The average is taken only over known values (assume the active user has provided m votes).
2. Ranking score of top-20 items. $R_{score} = 100 * (\sum R / \sum R_{max})$. This metric gives a value between 0 and 100 and was proposed in [5]. Higher values indicate a better ranking with top items as the most highly rated ones. We measure this metric only over known ratings. One big advantage of this metric is that it gives the same score for permutations of items with the same score. Thus if a user has rated 6 items with the maximum score 5, then the R_{score} is the same for any permutation of the ranking. This removes the problem of breaking ties.

6 Discussion

The results of the evaluation are encouraging. A simple NLDR to a manifold even with any explicit alignment of user profiles performs better than popular voting. Expectedly, the predicted votes become more accurate as more users cross over and their profiles are aligned. While the predictions are not as good as the gold standard even in the case of complete overlap according to the MAE, the algorithm provides a 4-5% improvement over the baseline after $\sim 35\%$ user profiles have been aligned. For collaborative filtering, this is not an insignificant improvement: the gold standard is only 12.6% better than the baseline. Experimental results also show that the top-N recommendation using manifold alignment is a significantly higher quality than the baseline. In case of complete overlap, Laplacian Eigenmap based manifold aligment can provide a $top - 20$ ranked list which is more relavant than the gold standard. The results presented here are obtained after a $10-$fold validation; in some cases, the algorithm was

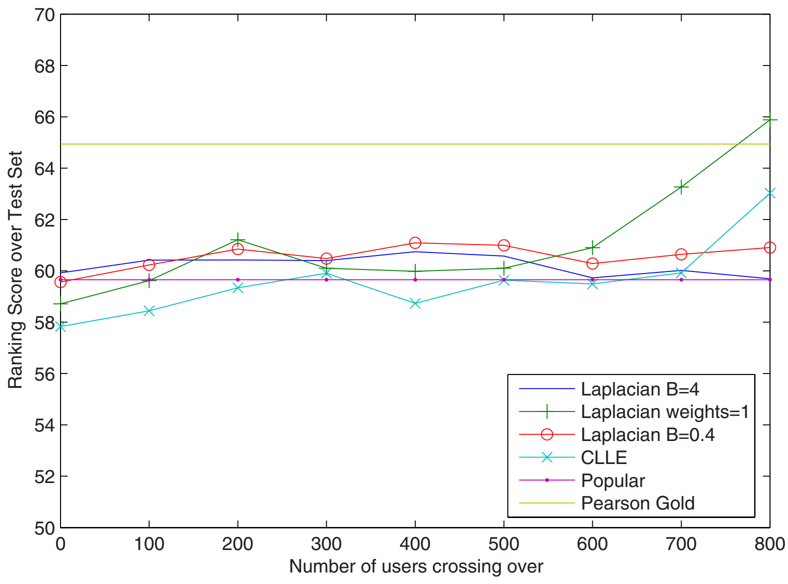

Fig. 2. Precision for the different NLDR methods within the manifold alignment framework. Numbers plotted are after 10-fold validation and averaging.

able to outperform the gold standard for MAE as well. One possible reason for the lower performance is the small size of data which is very sparse. With more training data, we expect to find more neighbors for every user which have votes for many items. Due to the sparsity of data, the majority of the normalized user database consists of mean. Therefore, the reconstructed values are heavily weighted towards the mean votes, especially for items that are note frequently rated. Previous research [8] has shown that learning from incomplete data offers significant advantage over strategies like mean imputation. Given that our approach works better than popularity votes even with a heavy bias towards mean values, algorithmic enhancements which offer a probablistic interpretation to manifold alignment are likely to be more accurate and form our future work.

6.1 Implementation and Performance

The manifold algorithm outlined in this paper has been implemented using Matlab R14 on a Pentium 4 based Desktop PC. Standard Matlab routines have been used and sparse matrices are used wherever possible. For the smaller MovieLens data with 100,000 rates, the algorithm uses around 100 MB of RAM. It performs reasonably w.r.t. to time as well. Each run of the *Algorithm 1* followed by *Algorithm 2* runs in approximately 5 seconds using Laplacian Eigenmaps. The LLE algorithm runs slower (70 seconds) since a quadratic program has to be solved for every point. The memory requirements of the LLE algorithm are also higher.

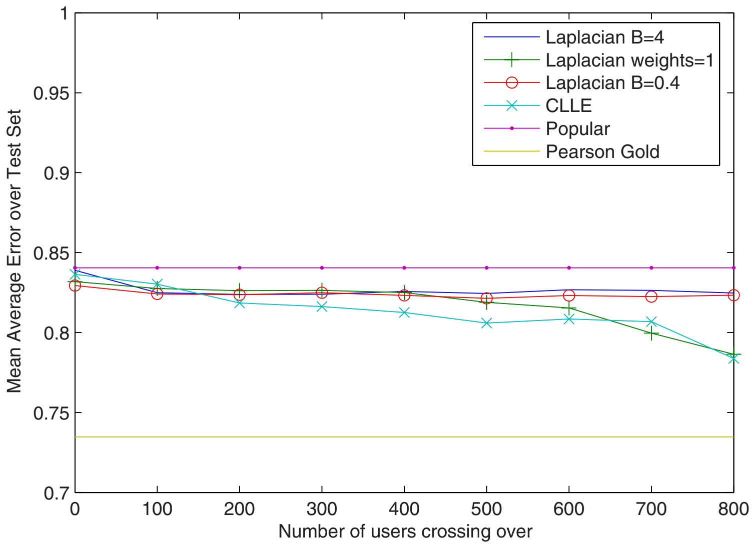

Fig. 3. Precision for the different NLDR methods within the manifold alignment framework. Numbers plotted are after 10-fold validation and averaging.

6.2 Computation Complexity

The Laplacian Eigenmap method clearly offers computational advantages over the LLE method. The LLE method has 3 basic steps: a) find nearest neighbors, b) compute reconstruction weights, and c) find eigenvalues and eigenvectors. For two datasets of sizes $m_{\mathcal{X}} \times n_{\mathcal{X}}$ and $m_{\mathcal{X}} \times n_{\mathcal{X}}$ with c common points, the size of the common graph is $n_{\mathcal{X}} + n_{\mathcal{Y}} - c$ nodes. The complexity of the LLE method for a matrix with n points each of dimensionality m is thus $O(mn) + O(dnk^3) + O(dn^2) \equiv O(dn(n+k^3))$. The Laplacian Eigenmap method essentially skips the second step, and hence has a complexity of $O(dn^2)$. Therefore the overall complexity of *Algorithm 1* (without the reconstruction of user profiles) is $O(dn(n+k^3))$ where $n = n_{\mathcal{X}} + n_{\mathcal{Y}} - c$. For our experiments, k typically had a value between $24 - 48$, while n was around 1000. In this range, k^3 was 1-2 orders of magnitude higher than n, thus explaining the difference between the running times of LLE and LapE based NLDR. Note however that this entire alignment computation can be performed off line. For a new user, out of sample extensions for LLE and Laplacian Eigenmaps[4] can be used. These typically have a computational complexity of $O(m) + O(dk^3)$. Importantly, the neighborhood formation step can be reused in the second part of the algorithm where user profiles have to be reconstructed.

The reconstruction of a user profile(*Algorithm 2*) involves (a) neighborhood formation (b) finding reconstruction weights, and (c) combining neighbor votes. The complexity reconstructing the profile for one user therefore is $O(dn) + O(dk^3) + O(mk)$. The significant term here depends on the values of the parameters: for a large neighborhood, the second term dominates. However if the number of items is very large (say a million), then the last term is the most significant one.

6.3 Usefulness in Practical Scenarios

With a variety of systems using personalization engines, there is a lot of data being collected about users as they go about their day to day pursuits. Combining this data from various sources in a secure and transparent way can significantly improve the level of personalization that electronic systems currently provide. In this scenario, creating an approach which makes very few assumptions about systems and users is of paramount importance. While our algorithm has been demonstrated in a collaborative filtering setting, there is no binding to use only rating data. The profiles of a content based system can be just as easily plugged in, as can be a profile from a hybrid system. Importantly, we also hypothesize that user profiles should be stored on the user's side in *Context Passport* which can leverage data about the user available with multiple systems. We envision that even data from operating systems and email clients can be plugged into the *Context Passport*. Our approach makes all of this possible in principle. However, the absence of relevant data, where user profiles of the *same users* at multiple sites are available, makes it difficult to evaluate the effectiveness of our algorithm in a real life setting. Our attempts are on to collect such a dataset in a scientific conference setting.

While our currently implementation performs all calculations in an online fashion, it is possible to implement the learning phase and the actual user profile reconstruction as separate phases. In a practical environment, the alignment learning would be performed off line and a new user will approach a new system with dimensionally reduced profile. This profile will then be projected on to the manifold using out-of-sample extensions, and reconstruction of the user profile can be done performed in real time. In case the item space is huge, the reconstruction phase can be performed online only on a sample set of item extracted from the entire item space. The entire profile can then be reconstructed offline.

6.4 Privacy

One important aspect of cross system personalization is privacy. People and companies alike are likely to have reservations against sharing their data with our systems. Users fear the loss of their anonymity, while companies fear a loss of their competitive edge. With our method, the important thing is to discover the underlying social similarity between people and not their exact buying/rating patterns. A less accurate, but more secure (w.r.t privacy) approach could start with a dimensionally reduced user database from say 1 million items to 1000 dimensions. Also the complete user database does not need to be known: a random selection of a sufficient number of users might be sufficient to learn the mapping from one system to another.

6.5 Scaling to a $n-$System Scenario

The manifold alignment algorithm needs only a minor modification in case some users are common to all n systems. This modification in in the step where a joint graph $\mathcal{G}$ is formed. The low dimensional embedding of this graph will have all the submanifolds aligned. More fined tuned modifications are required in case the set of overlapping users is different between different users. Manifold alignment in $n-$system scenario is successful only if a small fraction of users need to cross from one system to another. In order to test this scenario, larger datasets are needed.

7 Conclusions and Future Work

This paper outlines a novel approach to leverage user data distributed across various electronic systems to provide a better personalization experience. One major benefit of this approach is dealing with the *new user* problem: A new user of a collaborative filtering system can usually be provided only the non-personalized recommendation based on popular items. Our approach allows system to make a better prediction using the user's profile in other systems. The contribution of this paper is in describing an algorithm which offers a satisfactory improvement over *status quo* for a potentially important application scenario. Future work includes developing a practical framework around the manifold alignment algorithm. Also, there is a potential for improvement in performance both from an algorithmic and methodological point of view.

References

1. G. Bakir, J. Weston, and B. Schlkopf. Learning to find pre-images. *Advances in Neural Information Processing Systems*, 16:449–456, 2004.
2. M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In J. Shawe-Taylor and Y. Singer, editors, *COLT*, volume 3120 of *Lecture Notes in Computer Science*, pages 624–638. Springer, 2004.
3. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
4. Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS*, 2003.
5. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
6. V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, pages 705–712, 2002.
7. D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for highdimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100(10):5591–5596, 2003.
8. Z. Ghahramani and M. I. Jordan. Learning from incomplete data. Technical Report AIM-1509, MIT, 1994.
9. J. Ham, D. Lee, and L. Saul. Learning high dimensional correspondence from low dimensional manifolds. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 34–41, 2003.
10. J. Ham, D. Lee, and L. Saul. Semisupervised alignment of manifolds. In R. G. Cowell and Z. Ghahramani, editors, *AISTATS 2005*, pages 120–127. Society for Artificial Intelligence and Statistics, 2005.

11. M. Hein, J.-Y. Audibert, and U. von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *COLT*, pages 470–485, 2005.
12. S. Keerthi and W. Chu. A matching pursuit approach to sparse gaussian process regression. In Y. Weiss, B. Schlkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006.
13. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
14. B. Mehta, C. Niederee, and A. Stewart. Towards cross-system personalization. In *UAHCI*, 2005.
15. B. Mehta, C. Niederée, A. Stewart, M. Degemmis, P. Lops, and G. Semeraro. Ontologically-enriched unified user modeling for cross-system personalization. In *User Modeling*, pages 119–123, 2005.
16. J. W. Sammon. A non-linear mapping for data structure analysis. In *IEEE Transactions on Computing*, volume C18 (5), pages 401–409, May 1969.
17. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.
18. L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifold. *Journal of Machine Learning Research*, 4:119–155, 2003.
19. B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
20. U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
21. J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *NIPS*, pages 873–880, 2002.