Cross System Personalization by Factor Analysis

Bhaskar Mehta[⋆], **Thomas Hofmann**[†], **Peter Fankhauser**[⋆]
[⋆] Fraunhofer IPSI,Dolivostrasse 15, Darmstadt 64293, Germany
[†] Darmstadt University of Technology, Darmstadt 64289, Germany

Abstract

Today, personalization in information systems occurs separately within each system that one interacts with. However, there are several potential improvements w.r.t. such isolated approaches. Thus, 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 their profile, since it is deeply buried in personalization engines. *Cross-system personalization*, i.e. personalization that shares personal information across different systems in a user-centric way, overcomes these problems. User profiles, which are originally scattered across multiple systems, are combined to obtain maximum leverage. This paper discusses an approach in support of cross-system personalization, where a large number of users cross from one system to another, carrying their user profiles with them. These sets of corresponding profiles can be used to learn a mapping between the user profiles of the two systems. In this work, we present and evaluate the use of factor analysis for the purpose of computing recommendations for a new user crossing over from one system to another.

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. Each system independently builds up user profiles 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 (CSP) (Mehta, Niederee, & Stewart 2005) 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 relied on each user having a *unified profile* which different systems can understand. The unified profile contains facets modeling aspects of a multidimensional user which is stored inside a "*Context Passport*" that the user carries along in his/her journey across information space. When a user goes to a new system, the user's *Context Passport* is presented to a system, which can then understand the context in which the user wants to use the system. 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 results of the personalization process are then transfered back to the user's Context Passport via a protocol understood by both parties. 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*.

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. Both systems 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. For simplification, we assume that the complete user profiles for user i are represented as vectors $\mathbf{x}_i^A \in \mathcal{X} \subseteq \mathbb{R}^m$ and $\mathbf{x}_i^B \in \mathcal{Y} \subseteq \mathbb{R}^p$ for systems A and B, respectively. Given the profile $\mathbf{x}_i^A$ of a user in system A, the objective is then to find the corresponding profile $\mathbf{x}_i^B$ of the same user in system B. Formally we are looking to find a mapping

$$F_{AB} : \mathbb{R}^m \rightarrow \mathbb{R}^p, \qquad \text{s.t.} \quad F_{AB}(\mathbf{x}_i^A) \approx \mathbf{x}_i^B \qquad (1)$$

for all user i. The situation may be further complicated by the fact that for some users $\mathbf{x}_i^A$ may be only known partially, and that some aspects or values of $\mathbf{x}_i^B$ may not be directly observable at system B. In the latter case, the goal is to exploit the information available at system A in order to improve the inference about unknown values of $\mathbf{x}_i^B$.

Notice that if users exist for which profiles in both system are known, i.e. a training set $\{(\mathbf{x}_i^A, \mathbf{x}_i^B) : i = 1, \ldots, l\}$, then this amounts to a standard supervised learning problem. In contrast to standard regression problems that typically involve only a single real-valued response variable, however, the function F_{AB} that needs to be learned here 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. p can be 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 between response variables can seriously deteriorate the prediction accuracy 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), and that the profiles of the users in each of these systems is likely to be sparse (for collaborative filtering systems, this means that a small percentage of items are rated by each user). 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 $\mathbf{x}_i^A$ without corresponding $\mathbf{x}_i^B$ and vice versa. Similar issues also arise in Cross Language Information Retrieval (cf. (Oard 2003)).

In summary, we have three conceptual requirements from a machine learning method:

a) Perform vector-valued regression *en bloc* and not independently,

b) Exploit correlations between different output dimensions (or response variables), and

c) Utilize data without known correspondences.

In addition, the nature of the envisioned application requires:

a) Scalability of the method to large user populations and many systems/sites, and

b) Capability to deal with missing and incomplete data.

Related Work

There are a number of recent machine learning methods that can be utilized in principle for vector-valued regression problems like the one in Eq. (1), but some of them only partially fulfill the above requirements.

Kernel dependency estimation (KDE) is a technique that performs kernel PCA 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 solving computationally demanding pre-image problems for prediction (Bakir, Weston, & Schölkopf 2004). Another option is Gaussian process regression with coupled outputs (Keerthi & Chu 2006). 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.

There has been some recent work (Mehta & Hofmann 2006) which proposes the use of semi-supervised manifold alignment (Ham, Lee, & Saul 2005) for the purposes of cross system personalization. The latter is based on a manifold learning method, namely taking the Laplacian Eigenmap approach of (Belkin & Niyogi 2003) as the starting point. Laplacian Eigenmaps and other manifold learning techniques like *Locally Linear Embedding* (LLE) (Ham, Lee, & Saul 2003) are non-linear generalizations of linear dimension reduction and subspace techniques that have already been used successfully in various way for collaborative filtering. The problem with the approach described in (Mehta & Hofmann 2006) though is the lack of a probabilistic interpretation and the inability to deal with incomplete data and missing values in a principled way.

Factor Analysis

In this paper, we extend the collaborative filtering system proposed in (Canny 2002), which is based on the linear factor analysis model (Everitt 1984) to the CSP task. We start by motivating and introducing the factor analysis model.

Motivating Example

A very intuitive example explaining the idea behind Factor Analysis is available at Richard Darlington's homepage[1]. "Suppose each of 500 people, who are all familiar with different kinds of automobiles, rates each of 20 automobile models on the question, *How much would you like to own that kind of automobile?* One could usefully ask about the number of dimensions on which the ratings differ. A one-factor theory would posit that people simply give the highest ratings to the most expensive models. A two-factor theory would posit that some people are most attracted to sporty models while others are most attracted to luxurious models. Three-factor and four-factor theories might add safety and reliability."

Factor analysis is used to uncover the latent structure underlying a set of variables and as such is a "non-dependent"

[1] http://comp9.psych.cornell.edu/Darlington/factor.htm

procedure that does not require to explicitly specify dependent variables. It can be used to analyze the patterns of relationships between observed variables, eventually discovering the underlying (fewer and fundamental) independent variables that may not be directly observed. The inferred variables are called factors. A typical application of factor analysis suggests answers to following questions:

1. What are the latent factors underlying the data?
2. In which way do these factors explain correlations between observed variables?
3. How much of the observed variability is accounted for by latent factors, how much should be considered noise?

Factor analysis is also a generative model for *high dimensional data*, which is actually based on a low dimensional set of factors. Factor analysis is used to uncover the latent structure of a set of (observed) variables within such data, and to reduce the attribute space from a larger number of variables to a smaller number of factors.

Factor Analysis Model

Factor analysis is a latent variable model in which dependencies and correlations between multiple observable (dependent) variables $\mathbf{x}$ are explained by virtue of a typically much smaller number of latent variables or *factors* $\mathbf{z}$. In linear factor analysis the functional relationship between the observed random vector $\mathbf{x}$ and the unobserved $\mathbf{z}$ is assumed to be linear with some additive zero mean Gaussian noise added to each dimension of $\mathbf{x}$ independently. The fundamental equation that relates observables and latent factors can thus be described as

$$\mathbf{x} = \mu + \Lambda\mathbf{z} + \eta, \quad \eta \sim \mathcal{N}(0, \Psi), \tag{2}$$

where $\mu \in \mathbb{R}^m$ is a constant offset vector (mean), $\Lambda \in \mathbb{R}^{m\times k}$ is the matrix of factor loadings, and $\Psi = \text{diag}(\psi_1, \ldots, \psi_m)$ is a diagonal matrix modeling the variance of the additive Gaussian noise η. To complete the model, one usually assumes that *a priori* $\mathbf{z} \sim \mathcal{N}(0, I)$, i.e. the k latent factors follow an isotropic normal distribution with unit variance. The key assumption in factor analysis as in many latent class models is that conditioned on the latent classes, the observables are rendered independent; hence the crucial requirement on Ψ to be diagonal. It can be shown by integrating out the latent variables $\mathbf{z}$ that the distribution induced by factor analysis on the observables is a multivariate normal of the form

$$\mathbf{x} \sim \mathcal{N}(\mu, \Lambda\Lambda' + \Psi)\,. \tag{3}$$

This shows that factor analysis can be thought of as a multivariate normal model in which certain constraints are imposed on the co-variance matrix.

Factor Analysis for Collaborative Filtering

To cast the collaborative filtering scenario in the factor analysis framework, we assume the following: Let the user database for n users and m items be represented by a $m \times n$ matrix $\mathbf{X}$ with columns $\mathbf{x}_i$ corresponding to the profile for user i. We assume that each $\mathbf{x}_i$ is a realization of the random vector in Eq. (2) drawn i.i.d. from a factor analysis model with k factors and (unknown) parameters μ, Λ, and Ψ. We then can use the observed user ratings in $\mathbf{X}$ to learn the parameters of the model, e.g. using maximum likelihood estimation. (Canny 2002) has shown how factor analysis can also deal with missing data without the need for imputing values. This is a key advantage, since value imputation can be a pitfall in general (cf. (Ghahramani & Jordan 1994)). It is also relatively straightforward to compute posterior probabilities for missing $\mathbf{x}$-variables given observed ones, as well as computing the posterior over the latent factors. Notice in particular that the conditional mean of an unobserved rating is guaranteed to be a linear function of the observed ratings, which is a simple consequence of Eq. (3).

Factor Analysis for Cross System Personalization

In a two system scenario, we have two sites A and B, containing user profiles for their users represented as vectors. A user i has a profile $\mathbf{x}_i^A \in \mathbb{R}^m$ at site A, and a profile $\mathbf{x}_i^B \in \mathbb{R}^p$ at site B. We assume that c users are common to both site and that the data matrices can be partitioned as

$$\mathbf{X}^A = \begin{bmatrix}\mathbf{X}_c^A & \mathbf{X}_s^A\end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix}\mathbf{X}_c^B & \mathbf{X}_s^B\end{bmatrix}, \tag{4}$$

where $\mathbf{X}_c^A$ and $\mathbf{X}_c^B$ represent the sub-matrices of $\mathbf{X}^A$ and $\mathbf{X}^B$ corresponding to the common users and $\mathbf{X}_s^A$ and $\mathbf{X}_s^B$ the sub-matrices for users that are unique to A and B.

One way of looking at the CSP problem in the context of factor analysis is to relate the profiles in both (or multiple) systems by assuming that the user profiles are likely to be consistent in terms of the basic factors, i.e. that they can be explained by latent factors common to both systems. This is similar to the *manifold alignment* idea of (Ham, Lee, & Saul 2003) and effectively couples the factor analyses between the different systems.

A simple manner of enforcing this constraint is to construct a new combined random vector $\mathbf{x} = [\mathbf{x}^A\ \mathbf{x}^B]$ and to perform a joint factor analysis over the combined profile space of system A and B. This means we effectively generate a data matrix

$$\mathbf{X} = \begin{bmatrix}\mathbf{X}_c^A & \mathbf{X}_s^A & ? \\ \mathbf{X}_c^B & ? & \mathbf{X}_s^B\end{bmatrix}, \tag{5}$$

where '?' denotes matrices of appropriate size with unobserved values. Again we assume that the columns of $\mathbf{X}$ are independent realizations of $\mathbf{x}$ in a factor analysis model. Note that the other submatrices of $\mathbf{X}$ may also contain (many) missing entries.

It is interesting to make a further simplication by restricting the data matrix to users that are known to both systems

$$\mathbf{X}_c = \begin{bmatrix}\mathbf{X}_c^A \\ \mathbf{X}_c^B\end{bmatrix}, \tag{6}$$

and to ignore the data concerning users only known to one system. Obviously, this will accelerate the model fitting compared to working with the full matrix $\mathbf{X}$. Also, this setting is more realistic, since in a real world scenario, only the

restricted portion of crossing users might be made available by individual systems. This situation corresponds to a *supervised learning* setting where labeled output data is available for all training samples. However, it is likely to be less accurate than the *semi-supervised learning* setting where $\mathbf{X}$ is used, since the unlabeled samples will potentially improve the estimation of the parameters.

A third scenario is one, where each site gets profiles of some of its users from another system, but wants to make use of all of its locally available user profiles. In this case, factor analysis based on the data matrix

$$\mathbf{X}^A = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A \\ \mathbf{X}_c^B & ? \end{bmatrix} \text{ and } \mathbf{X}^B = \begin{bmatrix} \mathbf{X}_c^A & ? \\ \mathbf{X}_c^B & \mathbf{X}_s^B \end{bmatrix}, \quad (7)$$

will be performed at system A and B, respectively. We have explored this model in our experiments for the case where users with an existing profile at one site bring along their profile from the second system.

Factor Analysis using Expectation Maximization

A standard approach for performing maximum likelihood estimation in a factor analysis model is the expectation maximization (EM) algorithm. For completeness, we state here the expectation maximization recurrence relations (for simplicity restricted to the $\mu = 0$ case). In the EM approach maximum likelihood estimation is performed by maximizing the expected complete data log-likelihood with respect to the parameters of the model, i.e. one needs to perform the maximization

$$(\hat{\Lambda}, \hat{\Psi}) = \underset{\Lambda,\Psi}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{E}_{\mathbf{z}} \left[\log p(\mathbf{x}_i, \mathbf{z}; \Lambda, \Psi)\right], \quad (8)$$

where the expectation for $\mathbf{z}$ is computed with respect to the posterior distribution of $\mathbf{z}$ given a particular profile $\mathbf{x}_i$. Note that the latter will also depend on the parameters Λ and Ψ, so that both steps, the computation of the posteriors (E-step) and the re-estimation of the parameters (M-step) needs to be alternated until (guaranteed) convergence. The posterior distribution of the $\mathbf{z}$ is a multivariate normal for which the mean vector and co-variance matrix can be calculated as

$$\mathbf{E}\left[\mathbf{z}|\mathbf{x}\right] = \langle \beta, \mathbf{x} \rangle, \quad \text{where} \quad \beta = \Lambda'(\Psi + \Lambda\Lambda')^{-1} \text{ and} \quad (9)$$

$$\mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}\right] = \mathbf{I} - \beta\Lambda + \beta\mathbf{x}\mathbf{x}'\beta'. \quad (10)$$

Using these equalities, maximizing the expected complete data log-likelihood results in the equations

$$\Lambda = \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{E}(\mathbf{z}|\mathbf{x}_i)'\right) \left(\sum_{i=1}^{n} \mathbf{E}(\mathbf{z}\mathbf{z}'|\mathbf{x}_\mathbf{i})\right)^{-1} \quad (11)$$

$$\Psi = \frac{1}{n} \text{diag} \left[\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i' - \Lambda \mathbf{E}(\mathbf{z}|\mathbf{x}_i)\mathbf{x}_i'\right]. \quad (12)$$

A detailed derivation can be found in (Ghahramani & Hinton 1996).

We can rewrite the above recurrences compactly in matrix notation, using the entire user matrix $\mathbf{X}$, and the latent space matrix $\mathbf{Z}$, in a readable form as follows:

$$\boxed{\begin{aligned} \beta &= \Lambda'(\Psi + \Lambda\Lambda')^{-1} \\ \mathbf{Z} &= \beta\mathbf{X} \\ \Lambda^{[t]} &= \mathbf{X}\mathbf{Z}'(\mathbf{Z}\mathbf{Z}' + \Psi(\Psi + \Lambda\Lambda')^{-1})^{-1} \\ \Psi^{[t]} &= \frac{1}{n}\text{diag}(\mathbf{X}\mathbf{X}' - \Lambda^{[t]}\mathbf{Z}\mathbf{X}') \end{aligned}} \quad (13)$$

Sparse Factor Analysis

Canny's approach (Canny 2002) also uses an Expectation Maximization recurrence to solve the factor analysis model, while paying attention to the case of incomplete data.[2] Since

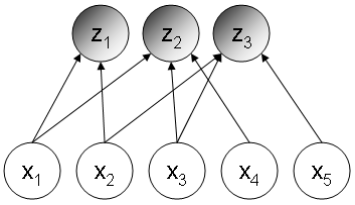


Figure 1: Factor analysis using incomplete data. Only observed variables $(x_1, x_2, x_3, ...)$ are used to predict the latent variables(z_1, z_2, z_3).

each user has rated only a subset of items, only the available rating data is used to compute the value of the latent variables, effectively removing missing variables from the inference process. Defining an $m \times m$ trimming diagonal matrix $\mathbf{T_i}$ for the i^{th} user which has $\mathbf{T_i}(j,j) = 1$ whenever the user i has voted for item j, the factor analysis E-step equations are modified as follows:

$$\mathbf{E}\left[\mathbf{z}|\mathbf{x}_i\right] = \langle \beta_i, \mathbf{x}_i \rangle, \; \beta_i = \Lambda_i'(\Psi + \Lambda\Lambda_i')^{-1}, \Lambda_i = \Lambda\mathbf{T}_i \quad (14)$$

$$\mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right] = \mathbf{I} - \beta_i\Lambda_i + \beta_i\mathbf{x}_i\mathbf{x}_i'\beta_i'. \quad (15)$$

Similarly, the M-step equations can be generalized to the missing data case to yield:

$$\boxed{\begin{aligned} \Lambda^{[t]} &= \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{E}[\mathbf{z}|\mathbf{x}_i]\right) \left(\sum_{i=1}^{n} \mathbf{T}_i \mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right]\right)^{-1} \\ \Psi^{[t]} &= \left(\sum_{i=1}^{n} \mathbf{T}_i\right)^{-1} \text{diag}\left(\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i' - \Lambda^{[t]}\mathbf{E}[\mathbf{z}|\mathbf{x}_i]\mathbf{x}_i'\mathbf{T}_i\right) \end{aligned}} \quad (16)$$

Further, the latent space projection of a given user profile $\mathbf{x}_i$ can be computed by

$$\mathbf{z}_i = \beta_i \mathbf{x}_i \quad (17)$$

A detailed derivation can be found in (Traupman & Wilensky 2004).

[2] An additional advantage of his approach is that the model building can be distributed among participating users in a manner that preserves privacy. In Canny's approach, each user can contribute to the model building by computing locally terms that contribute to the overall computation.

The model requires an initial value for Λ and ψ. A random matrix with a Gaussian distribution is used for Λ. This random matrices are then used by the linear regression model to generate an estimate. A linear regression model assumes no noise ($\psi = 0$), and can be obtained from the Eq. (13) by setting $\psi = 0$. The linear regression uses the following update scheme:

$$\mathbf{z} = (\Lambda'\Lambda)^{-1}\Lambda'\mathbf{x} \tag{18}$$
$$\Lambda^{[t]} = \mathbf{x}\mathbf{z}'(\mathbf{z}\mathbf{z}')^{-1}$$

Here the matrix $\mathbf{x}$ is a $m \times n$ user rating matrix, were missing values are replaced by some mean values. An overall average has been used by Canny. A few iterations of this recurrence gives a reasonable starting value of Λ which can be used by the factor analysis model. Please note that Canny's approach assumes that the observed variables have a mean of zero. While this is not true for user ratings, simple transformations like subtracting per-user or per-item means from every known value can create data with approximately zero mean. In our experiments, we have subtracted per-user means from known ratings.

In summary, the proposed method works as follows in practice: At first, the linear regression model in Eq. (18) is used to calculate an initial estimate of Λ, using a low dimensional latent space. After approximately 10 iterations of linear regression, the factor analysis model (cf. Eq. (16)) is initialized with this estimate of Λ. The EM recurrence for factor analysis (EM-FA) converges reliably in 15-25 iterations in all our experiments. After this, a new user with a partial rating vector $\mathbf{x}_j$ can be used to calculate $\mathbf{z}_j$ using Eq. (17). Given $\mathbf{z}_j$, $\hat{\mathbf{x}}_j = \Lambda\mathbf{z}_j$ provides the prediction for the complete rating vector.

Extension to n-system Scenario

One of the important factors on which the success of a method for CSP depends, is to be able to effectively use data spread across multiple systems. The 2-system scenario is clearly simplistic: a real world scenario has multiple systems with users having profiles in one or more of these systems. Besides the obvious case of a new user entering a system, where CSP has benefits, it is also desirable that the recommendation quality for an existing user can be improved by leveraging his/her profiles in other systems. While the n-system scenario can be dealt with by treating the entire setup like one huge collaborative system, with items and users ratings distributed across multiple sites, subtle issues make the difference crucial. These issues concern correspondence (user profiles on only a subset of systems), and item overlap. Therefore, a generalized approach should take these issues into consideration.

We start by assuming, as earlier, that each site can be modeled be a linear factor analysis model. Further, corresponding profiles have the same representation in the latent space. Thus, if random variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ represent the user profiles at the n sites respectively, then we form a new random variable $\mathbf{x}$ by concatenating the n vectors, i.e.:

$$\mathbf{x} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad ... \quad \mathbf{x}_n] \tag{19}$$

Let the matrix $\mathbf{X}$ encode all the user profiles: as before, each user now represents a column in this matrix. Note that this simple setup can allow for correspondence for every user, since the combined user profile is a concatenation of user profiles at individual sites. In case a site does not have a profile for the given user, a vector (of appropriate length) with missing values '?' is chosen. The resulting model can then be used in a fashion similar to the 2-system scenario: All known ratings for all systems are combined appropriately (as dictated by Eq. (19)) to yield a vector of length m (where m is the sum of number of items in all n -systems). This vector is then first projected to get the lower dimension embedding $\mathbf{z}$ in the latest space, and then $\Lambda\mathbf{z}$ gives the desired complete vector with all predicted values filled in. This approach also does not need explicit item to item mapping in case the same items are available in more than one system. Instead, the learning approach can figure this out by looking at only principal components in the data.

CSP for Existing Users

As mentioned before, one of the objectives of CSP is to leverage data about the user available at other sites. While we posit that new users can unmistakably gain from CSP, users with existing profiles should also be able to gain from their profiles at other systems in a similar way. The advantage offered by CSP in this context is a systematic and principled integration of data and evidence across all systems. By taking user's ratings at other sites into account in addition to locally gathered data, one can expect to get more accurate ratings. In our experimental results, we have tested this hypothesis by performing *All-but-1*, *All-But-5*, and *Only-n* tests at site A, while providing information about the ratings at site B as side-information.

Data and Evaluation

We choose the EachMovie[3] data with ratings from 72,916 users for 1,682 movies. Ratings are given on a numeric six point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). The entire dataset consists of around 2.8 million votes, however around 2.1 million of these votes are by the first 20,000 users. We chose this dense subset of 20,000 users and 1682 movies and scaled ratings to integers between 1 and 6. We split this data set into two parts to form two datasets by splitting the item set of the entire data. This way we get two datasets with the same number of users, but with ratings over different items. To mimic a real life setting, we allow a random 5% of items to overlap between the data sets. The overlap is not explicitly maintained nor is the correspondence information made available to the learning algorithm. Moreover, we choose 10,000 test users, since this model is useful only if it works for a large number of users, with only a few correspondences known. In our test runs, we build an FA model using the matrix $\mathbf{X}$ (see eq. (4)) varying c from 500 users to 10,000 users. For the users not in correspondence, we randomly rearrange their order. In our setting, it is vital that we can build an effective predictive model with as few users crossing over from one system to another which works effectively for a

[3] http://research.compaq.com/SRC/eachmovie

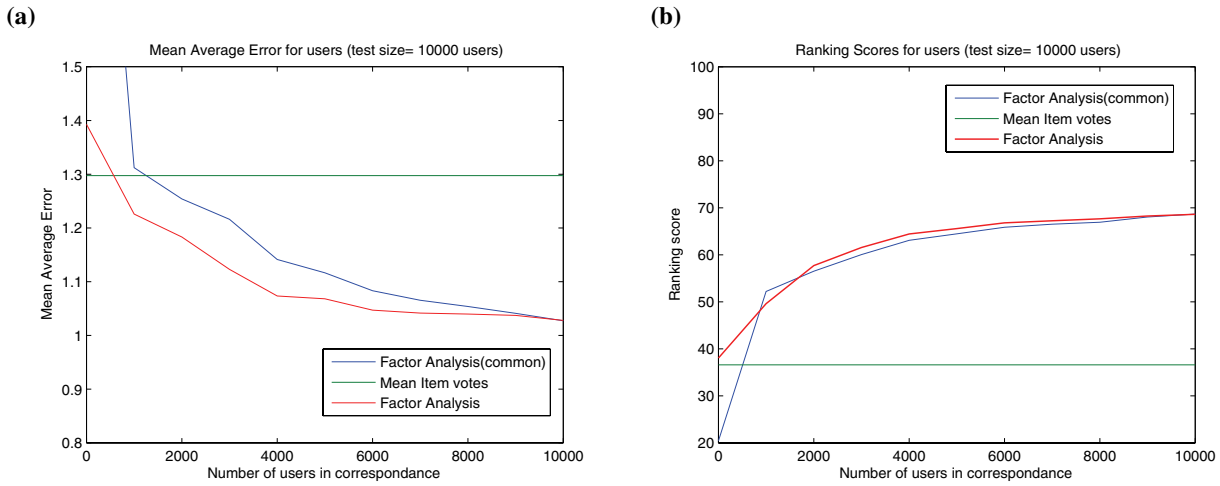
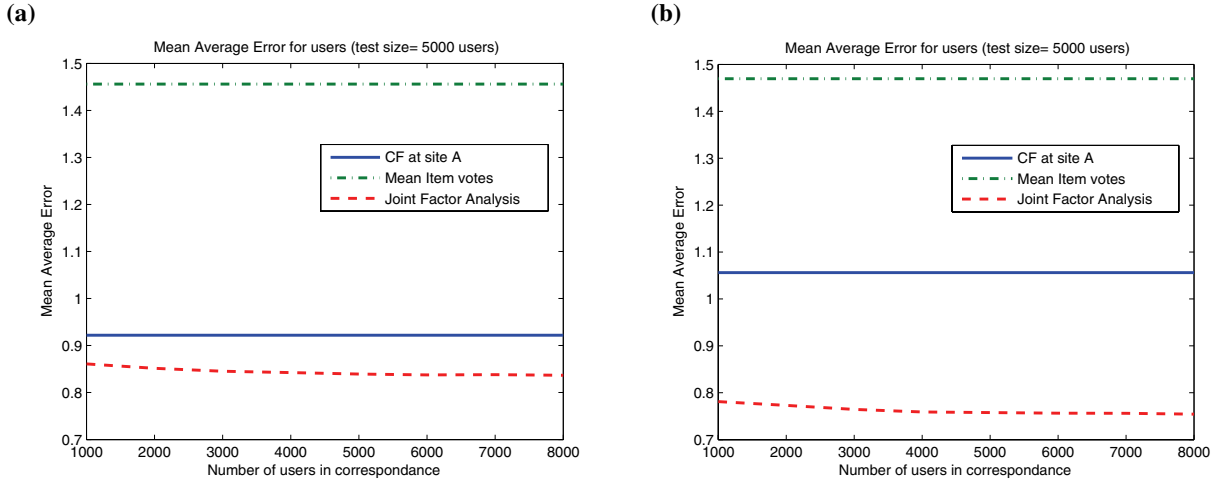


Figure 2: MAE and Ranking scores for 10,000 test users (with 10 fold validation). "$common$" refers to the use of only common users (Eq. 5) for training the model.



(a) MAE for *All-But-1* protocol **(b) MAE for *All-But-5* protocol**

Figure 3: MAE and Ranking scores for 5000 test users (with 10 fold validation)

large number of new users. We randomly choose the test set and the item set for every run of the algorithm. In addition, we also performed the model building step using only the users common to both systems using $\mathbf{X}_c$ (see Eq. (5)).

Time and Space complexity

The time complexity of a single iteration of sparse FA is $O(nmk^2)$, which is linear in the size of the user database, and quadratic in k, which is the number of dimensions in the latent space. The prediction time is $O(mk^2)$ which is linear in the number of items. The space complexity is low $\sim O(mk)$, as is characteristic of model based collaborative filtering.

Implementation

The factor analysis algorithm has been implemented on a standard Pentium IV based PC running Matlab R14. Canny's original Matlab code has been made available publicly[4], and this served as a starting point for our purposes. The core code is very quick and scales very well to a large data set like EachMovie. Running times for the model building phase with 10,000 users is around 40 seconds, and less than 10 seconds when only common users are used. Prediction times are much faster: recommendations of all 10,000 test users are generated in 6.5 seconds. We have used $k = 14 - 20$, and have found that there is negligible difference for values higher than 14: hence we report results for $k = 14$.

Metric used

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-N items. $R_{score} = \frac{100 * \sum R}{\sum R_{max}}$. Ranking scores were introduced in (Breese, Heckerman, & Kadie 1998) and have been widely used in literature to evaluate the ranking of a top-n recommendation. We choose the only the top 20 items instead of the entire set of rated items. The choice of 20 items is also based on the fact that we have picked users with at least 20 votes, and would like to evaluate this metric over observed votes only. This metric gives a values between 0 and 100. Higher values indicate a ranking with top items as the most highly rated ones. 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.

Results

In order for CSP to be useful, we require CSP to provide a perceptable advantage over status quo. Clearly, a new user who has profiles with other systems should be able to use his/her previous ratings to get a meaningful recommendation. This advantage should increase when the number of systems in the CSP setting is higher (i.e. many users have profiles in n systems and provide this information when moving to a new system), or when the user already has a profile at the current system. In summary, our evaluation aimed at testing 3 hypotheses:

1. CSP offers an advantage over *popular item voting*[5] for a large number of first time users,
2. CSP offers an advantage for existing users in a standard collaborative filtering setting, and
3. CSP offers an advantage in a n-system scenario.

Figure 2. provides experimental evidence for the first hypothesis. While popular voting has been reported to provide a good recommendation quality, here it performs very poorly. The simple reason for this is our large number of test users; clearly the same rating and ranking can not appeal to a large variety of users. In contrast, the CSP approach offers significantly better results, performing 12% better than the baseline when 4,000 users have crossed over. When the complete set of training users are set in correspondence, this advantage jumps to 22%. Note that these numbers are not insignificant for a collaborative filtering setting, where the best performing k-Nearest Neighbor algorithms perform only 12-16% better than popularity voting.

Not surprisingly, the *semi-supervised* approach to CSP performs better than the *supervised* approach. In the supervised approach, only the users with known correspondences are used for training the model. While this approach is clearly more efficient, the early advantage gained by semi-supervised learning here is to the tune of 6% when 4,000 users have crossed. Subsequently, both approaches perform similarly, but this is because the input data to both starts to look much similar and is actually the same when all the training users have been set in correspondence. Clearly, more available data helps to make the model learn faster.

The advantage offered by CSP is even more evident in the Ranking score metric, which depicts the utility of the ranked recommendation list returned to the user. This metric was originally proposed in (Breese, Heckerman, & Kadie 1998) and bases itself on the observation that the utility of an item in the ranked list decays exponentially, with a half life of, say 5 items. We have measure this metric only over the observed votes. As figure 2 shows, the popularity voting offers a very low ranking score in comparison with the CSP approach. The advantage is more than 70% when around 4000 users have crossed over, and conitnues to grow a further 15% till all the training set has been used.

To test the second hypothesis, the following setup is used: the entire set is split into 2 parts as earlier. At site A, 10,000 users are used to train a model of just site A. This model is then used to generate recommendations for 5,000 test users, and then evaluated using the *All-But-1*, *All-but-5*, and *Only-n* protocols (see (Breese, Heckerman, & Kadie

[4] http://guir.berkeley.edu/projects/mender/

[5] Popular voting involves using mean rating of every item and recommending the mostly highly rated items to the active user

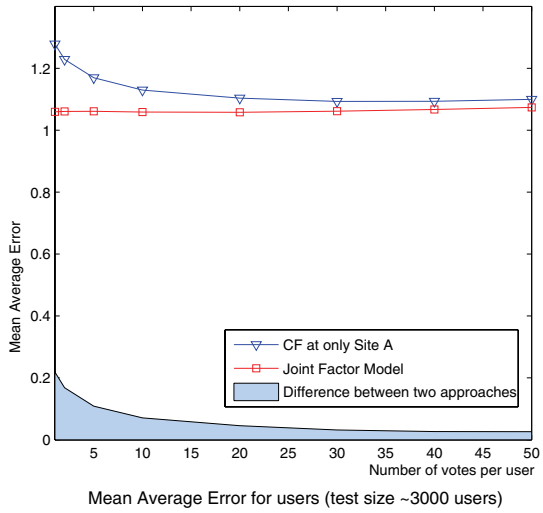


Figure 4: MAE and Ranking scores for $\sim 3,000$ test users (with 10 fold validation) in an *Only-n* scenario. n is varied from 1 to 50.

1998) for details on these protocols). In addition, a second model is trained with 8,000 users(total), out of which some users have profiles at Site B. The number of such profiles in correspondence is increased from 1000 to 8000. Now, this second model is used to generate recommendation for 5,000 test users *for site* A. Notice that the test users have their complete profiles from site B available, in addition to some ratings for site A. Figure 3 provides evidence for this hypothesis, where the MAE for *All-but-1* and *All-but-5* show an advantage for CSP even for existing users. Clearly, this advantage is higher when a lower number of votes for a user are available at site A. Figure 4 shows the results of an experiment where the number of votes available at site A (per user) was varied from 1 to 50. As test users, we chose only those users who had cast at least 50 votes in our test set of 5000 users. Over various runs, this number of test users was noted to be around 3000. It is clear that CSP has a big early advantage over single-system collaborative filtering to the tune of 20%. This advantage decreases to 5% when 20 votes per user are available. At the 50-votes per user point, the advantage is less than 3%.

To test the third hypothesis, we constructed a setup with 3 systems in the following way: for 4,000 users, corresponding profiles from all 3 systems are available, for 2,000 users, profiles for system 1 and 2 are available, and for the next 2,000 users, profiles from system 1 and 3 are available. Now for 5,000 test users, we tested the 3 following scenarios for new users at system 1:

1. No profile is available
2. Profile from system 2 is available
3. Profile from system 2 and 3 is available.

	Popular vote	**Only from A**	**From A,B**
MAE	2.6052	1.1725	1.1142
RS	14.5585	61.8124	66.4900

Table 1: MAE and Ranking Score for 3-system scenario. Each system had an non-overlapping item set of size 500.

With this setup, we constructed a joint factor model and evaluated the predictions of the model in the cases stated above. Table 1. shows the MAE and Ranking Scores for the n-system scenario. The results clearly show that in an n-system environment, CSP offers a definite advantage. When a new user brings along a profile from another system, there is a dramatic improvement in performance. The numbers in this experiment for MAE are higher than the 2-system cases due to the decreased size of the item sets for each system. Finally, a new user who brings profiles from two systems has a bigger advantage than a user make available his/her profile from only from one system.

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, developing an approach which makes very few assumptions about systems and users is of paramount importance. While our approach 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*, and can be communicated from the user to the system and back using protocols like CSP (Mehta, Niederee, & Stewart 2005). 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.

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.

The framework we adopt here is based on Privacy Enhanced Collaboratve filtering (cf. (Canny 2002)), where user profile data is not known to anyone except the system and the user. In our case, we could even relax the requirement of any one system requiring the complete profile, and maintaining the unified user profile only at the user's end (cf. (Mehta, Niederee, & Stewart 2005)) in the form of a passport. We have previously called this unified profile as the Context Passport. The computation of the model parameters can be decentralized and users can provide only *their* profile's contribution to a central totaller. Importantly, this contribution can be encrpyted, and properties of public-key/private-key encryption allow the computation to proceed with these encrypted values without loss of precision. The interested reader can find more details of the encryption framework and its use in (Canny 2002).

Conclusion

We have outlined a novel approach for leveraging user data distributed across various electronic systems in order 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 overall item popularity. Our approach allows making better predictions by utilizing the user's profile in other systems. The main contribution of this paper is the presented factor analysis method, which offers a satisfactory improvement over status quo for a potentially important application scenario. It also offers an algorithmic improvement over previous work by taking into account the incompleteness of data. Future work includes developing a practical framework around the sparse Factor Analysis algorithm for CSP and exploring further algorithmic improvements and alternatives.

References

Bakir, G.; Weston, J.; and Schölkopf, B. 2004. Learning to find pre-images. In Thrun, S., L. S., and Schölkopf, B., eds., *Advances in Neural Information Processing Systems*, volume 16, 449–456. Cambridge, MA, USA: MIT Press.

Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.

Breese, J. S.; Heckerman, D.; and Kadie, C. M. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 43–52.

Canny, J. F. 2002. Collaborative filtering with privacy via factor analysis. In *SIGIR*, 238–245.

Everitt, B. S. 1984. *An Introduction to Latent Variable Models*. New York: Chapman and Hall.

Ghahramani, Z., and Hinton, G. E. 1996. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto.

Ghahramani, Z., and Jordan, M. I. 1994. Learning from incomplete data. Technical Report AIM-1509, MIT.

Ham, J.; Lee, D.; and Saul, L. 2003. 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*.

Ham, J.; Lee, D.; and Saul, L. 2005. Semisupervised alignment of manifolds. In Cowell, R. G., and Ghahramani, Z., eds., *AISTATS 2005*, 120–127. Society for Artificial Intelligence and Statistics.

Keerthi, S., and Chu, W. 2006. A matching pursuit approach to sparse gaussian process regression. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press.

Mehta, B., and Hofmann, T. 2006. Cross system personalization by learning manifold alignment. Submitted for publication.

Mehta, B.; Niederee, C.; and Stewart, A. 2005. Towards cross-system personalization. In *UAHCI*.

Oard, D. W. 2003. Information retrieval systems as integration platforms for language technologies. In *HLT-NAACL*.

Traupman, J., and Wilensky, R. 2004. Collaborative quality filtering: Establishing consensus or recovering ground truth? In *WebKDD: : KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*.