# Apprendre à construire des représentations à partir d'information incomplète : Application au démarrage à froid en filtrage collaboratif

Gabriella Contardo[1], Thierry Artières[2], and Ludovic Denoyer[1]

[1]Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
[2]Ecole Centrale Marseille - Laboratoire d'Informatique Fondamentale (Aix-Marseille Univ.), France

April 30, 2015

## Abstract

Une bonne représentation des données est une composante essentielle du succès d'un algorithme d'aprentissage automatique. Lorsque la représentation est apprise, son apprentissage est classiquement réalisé sur les données complètes, c'est à dire totalement observées. Or, dans certaines situations, on ne découvre les données que progressivement et on aimerait pouvoir prendre des décisions avant de connaitre la donnée complète, ce qui signifie d'être capable de construire une représentation de la donnée *à la volée*. C'est le cas par exemple de tâches de personnalisation où l'information sur un utilisateur n'est connue qu'au fur et à mesure de son interaction avec le système. Cet article s'intéresse à ce cadre et décrit une stratégie d'apprentissage de représentation qui permet (i) de sélectionner au fur et à mesure l'information qui serait la plus pertinente pour construire une représentation de la donnée, (ii) de mettre à jour la représentation de la donnée au fur et à mesure de la collecte d'informations sur celle-ci. Nous appliquons notre approche à la conception d'interviews statiques pour le problème de démarrage à froid en filtrage collaboratif mais notre point de vue est générique. Notre appproche peut en particulier permettre de concevoir des méthodes qui dépassent le cadre du démarrage à froid et s'adaptent au cadre plus classique du filtrage collaboratif au fur et à mesure que les données d'un utilisateur sont disponibles.

## 1 Introduction

Representation learning has recently gain a surge of interest in machine learning, illustrating the need for models capable of processing raw (potentially large) data and pulling out useful information. This has been highlighted ([Bengio et al., 2013]) as a crucial task in order to go further in Artificial Intelligence. While classic approaches to do so assume that data are fully observed (or observable), in many applications one only has access to partial information, for example when dealing with streams, or when the information can be acquired through an active acquisition process where the system has to 'ask' for new information. In this context, there is a need to build representations "on the fly", based only on these available partial information. Moreover, one would ideally want to be able to choose which information to acquire to improve the final decision. This is the case for instance in personalized applications where information is generated through users activities and has to be integrated in the model as soon as it has been produced. Another critical application of such a problem is the *cold-start* setting in recommender systems, on which we concentrate in this paper. Recommender systems have become an active field of research and are now used in an increasing variety of applications, such as e-commerce, social networks or participative platforms. They aim to suggest the most relevant items (e.g products) to each user, in order to facilitate their experience. To recommend such relevant items, recommender systems can rely on different types of data, such as users' explicit and/or implicit feedbacks (e.g rating a movie on a scale of stars, buying an item or listening to a song), or informative features about users (age, post code) or items (type of movie, actors). One of the most common approach to recommendation is **Collaborative Filtering** (CF) which consists in making recommendation only based on the ratings provided by users over a set of items (i.e without using any additional features).

Within CF context, a popular and efficient family of methods are *Latent Factor Models*, which rely on matrix factorization-based techniques [1]. These approaches treat the recommender problem as a representation learning one, by computing representations for users and items in a common

[1]Other families of approaches are detailed in Section 4.

latent space. More formally, let us consider a set $\mathcal{U}$ of $U$ known users and a set $\mathcal{I}$ of $I$ items. Let $r_{u,i}$ denote the rating of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. A rating is usually a discrete value between 1 and 5, that can be binarized (-1/1) with a proper threshold (often 3). The $U \times I$ matrix $\mathcal{R} = \{r_{u,i}\}$ is the rating matrix which is incomplete since all ratings are not known. We will denote $\mathcal{O}$ the set of observed pairs $(u, i)$ such that a rating on item $i$ has been made by user $u$. Let us denote $N$ the dimension of the latent representation space of users and items, $p_u \in \mathbb{R}^N$ being the (learned) representation of user $u$ and $q_i \in \mathbb{R}^N$ denoting the (learned) representation of item $i$. Given these representations, classical approaches are able to compute missing ratings made by a user $u$ over an item $i$ as the dot product between $p_u$ and $q_i$. In other words, the more similar the user and the item representations are, the higher the predicted rating will be. Let us denote $\tilde{r}_{u,i}$ this predicted rating, we have:

$$\tilde{r}_{u,i} = q_i^T p_u \tag{1}$$

The representation $p_u$ and $q_i$ are usually learned on the sparse input rating matrix $\mathcal{R}$ by minimizing an objective loss function over $\mathcal{L}(\mathbf{p}, \mathbf{q})$ which measures the difference between observed ratings $r_{u,i}$ and predicted ones. $\mathbf{p}$ is the set of users representations, $\mathbf{q}$ being the representation of items. The loss is usually defined as a $L_2$ objective:

$$\begin{aligned}\mathcal{L}(\mathbf{p}, \mathbf{q}) = &\sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T p_u)^2 \\ &+ \lambda(\sum_i ||q_i||^2 + \sum_u ||p_u||^2)\end{aligned} \tag{2}$$

The coefficient $\lambda$ is a manually defined regularization coefficient. This loss corresponds to a matrix decomposition in latent factors and different optimization algorithms have been proposed as alternated least squares or stochastic gradient descent ([Koren et al., 2009]). Note that this models is a **transductive model** since it allows one to compute representations over a set of *a priori* known users and items.

The transductive nature of Matrix Factorization approaches makes them well adapted when the sets of users and of items are fixed. Yet in practical applications, new items and new users regularly appear in the system. This requires often retraining the whole system which is time consuming and also makes the system behavior unstable. Furthermore, one main limitation of transductive Matrix Factorization approaches is that they strongly rely on a certain amount of data to build relevant representations, e.g. one must have enough ratings from a new user to construct an accurate representation. Indeed, facing new users, MF methods (and more generally CF-based approaches) have to wait for this user to interact with the system and to provide ratings before being able to make recommendations for this user. These methods are thus not well-suited to propose recommendation at the beginning of the process.

We propose to focus on the user cold-start problem[2] by static interview method, which consists in building a set of items on which ratings are asked to any new user. Note that, as discussed in Section 4, our approach is not adaptive but is justified by observations made over real groups of users [Golbandi et al., 2011] that prefer to provide answers to a list of questions instead of facing a sequence of questions. Once the interview has been carried out, recommendations are made based on this list of (incomplete) ratings. We consider a representation-learning approach which is an original approach in this context and which simultaneously learns which items to use in the interview, but also how to use these ratings for building relevant user representations. Our method is based on an inductive model whose principle is to code ratings on items as translations in the latent representation space, allowing to easily integrate different opinions at a low computational cost. The contributions of this paper are thus the following:

1. We propose a generic representation-learning formalism for user cold-start recommendation. This formalism integrates the representation building function as part of the objective loss, and restriction over the number of items to consider in the interview process.

2. We present a particular representation-learning model called **Inductive Additive Model** (IAM), which is based on simple assumptions about the nature of users' representations to build and that we are able to optimize using classical gradient-descent algorithms.

3. We perform experiments on four datasets in the classical CF context as well as in the user cold-start context. Quantitative results show the effectiveness of our approach in both contexts while qualitative results show the relevancy of learned representations. We also conduct experiments that illustrate the ability of our approach to handle the transition from the cold-start setting to the warm classical recommendation problem.

The paper is organized as follow: in Section 2, we propose the generic formulation of the representation learning problem for user cold-start, and the particular instance of model we propose. The Section 3 presents the experiments and Section 4 discusses the related work in the collaborative filtering domain. Section 5 proposes perspectives to this contribution.

[2]The integration of new items which is less critical in practical applications is not the focus of this paper but is discussed in the conclusion.

## 2 Proposed Approach

We now rewrite the objective function detailed in Equation 2 in a more general form that will allow us to integrate the user cold-start problem as a representation-learning problem. As seen above, we still consider that each item will have its own learned representation denoted $q_i \in \mathbb{R}^N$ and focus on building a user representation. When facing any new user, our model will first collect a set of ratings by asking a set of queries during a static interview process. This process is composed by a set of items that are selected during the training phase. For each item in the interview, the new user can provide a rating, but can also choose not to provide this rating when he has no opinion. This is typically the case for example with recommendation of movies, where users are only able to provide ratings on movies they have seen. The model will thus have to both select relevant items to include in the interview, but also to learn how (incomplete) collected ratings will be used to build a user representation.

Let us denote $\mathcal{Q} \subset \mathcal{I}$ the subset of items that will be used in the interview. The representation of a new incoming user $u$ will thus depend on the ratings of $u$ over $\mathcal{Q}$ that we will note $\mathcal{Q}(u)$. This representation will be given by a function $f_\Psi(\mathcal{Q}(u))$ whose parameters, to be optimized, are denoted $\Psi$. These $\Psi$ parameters are global, i.e shared by all users. The objective function of the cold-start problem (finding the parameters $\Psi$, the items' representations and the interview questions conjointly) can then be written as:

$$\begin{aligned}\mathcal{L}^{cold}(\mathbf{q}, \Psi, \mathcal{Q}) = &\sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T f_\Psi(\mathcal{Q}(u)))^2 \\ &+ \lambda_1 (\sum_i ||q_i||^2 + \sum_u ||f_\Psi(\mathcal{Q}(u))||^2) \\ &+ \lambda_2 \#\mathcal{Q}\end{aligned} \tag{3}$$

The difference between this loss and the classical CF loss is twofold: (i) first, the learned representations $p_u$ are not free parameters, but computed by using a parametric function $f_\Psi(\mathcal{Q}(u))$, whose parameters $\Psi$ are learned; (ii) the loss includes an additional term $\lambda_2 \#\mathcal{Q}$ which measures the balance between the quality of the prediction, and the size of the interview, $\#\mathcal{Q}$ denoting the number of items of the interview; $\lambda_1$ and $\lambda_2$ are manually chosen hyper-parameters - by changing their values, the user can obtain more robust models, and models with more or less interview questions. Note that solving this problem aims at simultaneously learning the items representations, the set of items in the interview, and the parameters of the representation building function.

### 2.1 Inductive Additive Model (IAM)

The generic formulation presented above cannot easily be optimized with any representation function. Particularly, the use of a transductive model in this context is not trivial and, when using MF-based approaches in that case, we only obtained very complex solutions with a high computation complexity. We thus need to use a more appropriate representation-learning function $f_\Psi$ that is described below. The Inductive Additive Model (IAM) is based on two simple ideas concerning the representation of users we want to build: (i) First, one has to be able to provide good recommendation to any user that does not provide ratings during the interview process $\mathcal{Q}$. (ii) Second we want the user representation to be easily enriched as new ratings are available. This feature makes our approach suitable for the particular cold-start setting but also for the standard CF setting as well.

Based on the first idea, IAM considers that any user without answers will be mapped to a representation denoted $\Psi_0 \in \mathbb{R}^N$. Moreover, the second idea naturally led us to build an additive model where a user representation is defined as a sum of the particular items' representations. This means that providing a rating will yield a **translation** of the user representation in the latent space. This translation will depend on the item $i$ but also on the rating value. This translation will be learned for each possible rating value and item, and denoted $\Psi_i^r$, where $r$ is the value of the rating. More precisely, in case of binary ratings *like* and *dislike*, the *like* over a particular item will correspond to a particular translation $\Psi_i^{+1}$, and a *dislike* to the translation $\Psi_i^{-1}$. The fact that the two rating values correspond to two different unrelated translations is interesting since, for some items, the *dislike* rating can provide no additional information represented by a null translation, while the *like* rating can be very informative, modifying the user representation - see Section 3 for a qualitative study over $\Psi$. The resulting model $f_\Psi$ can thus be written as:

$$f_\Psi(u, \mathcal{Q}) = \Psi_0 + \sum_{(u,i)\in\mathcal{O}/i\in\mathcal{Q}} \Psi_i^{r_{u,i}} \tag{4}$$

where the set $\{(u,i) \in \mathcal{O}/i \in \mathcal{Q}\}$ is the set of items selected in the interview on which user $u$ has provided a rating.

#### 2.1.1 Continuous Learning Problem

Now, let us describe how the objective function described in Equation 3 with IAM model described in Equation 4 can be optimized. The optimization problem consisting in minimizing $\mathcal{L}^{cold}(\mathbf{q}, \Psi, \mathcal{Q})$ over $\mathbf{q}$, $\Psi$ and $\mathcal{Q}$ is a combinatorial problem since $\mathcal{Q}$ is a subset of the items. This combinatorial nature prevents us from using classical optimization methods such as gradient-descent methods and involves an intractable

number of possible combinations of items. We propose to use a $L_1$ relaxation in order to transform this problem in a continuous one. Let us denote $\alpha \in \mathbb{R}^I$ a weight vector, one weight per item, such that if $\alpha_i = 0$ then item $i$ will not be in the interview. The cold-start loss can be rewritten with $\alpha$'s as:

$$\mathcal{L}^{cold}(\mathbf{q}, \Psi, \alpha) = \sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T f_\Psi(u, \alpha))^2 + \lambda|\alpha| \quad (5)$$

Note that the $L_2$ regularization term over the computed representation of users and items is removed here for sake of clarity. The representation of a user thus depends on the ratings made by this user for items $i$ that have a non-null weight $\alpha_i$, restricting our model to compute its prediction on a subset of items which compose the interview. If we rewrite the proposed model as:

$$f_\Psi(u, \alpha) = \Psi_0 + \sum_{(u,i)\in\mathcal{O}} \alpha_i \Psi_i^{r_{u,i}} \quad (6)$$

then we obtain the following loss function:

$$\mathcal{L}^{cold}(\mathbf{q}, \Psi, \alpha) = \sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T(\Psi_0 + \sum_{(u,i)\in\mathcal{O}} \alpha_i \Psi_i^{r_{u,i}}))^2 + \lambda|\alpha| \quad (7)$$

which is now continuous. Note that, in that case, the translation resulting from a rating over an item corresponds to $\alpha_i \Psi_i^{r_{u,i}}$ rather than to $\Psi_i^{r_{u,i}}$.

#### 2.1.2 Cold-Start IAM (CS-IAM) Learning Algorithm

This objective loss (Equation 7) can be optimized by using stochastic gradient-descent methods, such as the one detailed in Algorithm 1. At each iteration, a user is selected randomly (Line 3). His ratings are used to compute the gradient for each parameters $(\mathbf{q}, \Psi, \alpha)$ and to update them accordingly (Lines 6-9). Since the loss contains a $L_1$ term that is not derivable on all points, we propose to use the same idea than proposed in [Carpenter, 2008], which consists in first making a gradient step without considering the $L_1$ term, and then applying the $L_1$ penalty to the weight to the extent that it does not change its sign. In other words, a weight $\alpha_i$ is clipped when it crosses zero. This corresponds to the lines 9-18 in Algorithm 1.

**Algorithm 1** Learning algorithm for CS-IAM

**Require:** $\mathcal{O}$ : set of observed ratings.
**Require:** $\epsilon$ : gradient step.
**Require:** $\lambda_{l_1}$ for $l_1$-regularization.
1: Initialize $\mathbf{q}, \Psi, \alpha$ randomly.
2: **repeat**
3: Select a random user $u$, where $r_u$ are the ratings of user $u$ in $\mathcal{O}$.
4: Compute predicted ratings $\hat{r}_u = q^T f_\Psi(\alpha, r_u)$ { See Eq. 6 }
5: {Update parameters accordingly with gradient descent :}
6: $\mathbf{q} \leftarrow \mathbf{q} - \epsilon\nabla\mathcal{L}^{cold}(\mathbf{q}, \Psi, \alpha)$
7: $\Psi \leftarrow \Psi - \epsilon\nabla\mathcal{L}^{cold}(\mathbf{q}, \Psi, \alpha)$
8: { L1-regularization on $\alpha$ using clipping :}
9: $\alpha \leftarrow \alpha - \epsilon\nabla\mathcal{L}^{cold}(\mathbf{q}, \Psi)$
10: **for all** $\alpha_i$ **do**
11: **if** $\alpha_i < 0$ **then**
12: $\alpha_i = min(0, \alpha_i + \lambda_{l_1})$
13: **else**
14: **if** $\alpha_i > 0$ **then**
15: $\alpha_i = max(0, \alpha_i - \lambda_{l_1})$
16: **end if**
17: **end if**
18: **end for**
19: **until** stopping criterion
20: **return** $\mathbf{q}, \Psi, \alpha$

## 2.2 IAM and classical warm collaborative filtering

The IAM, which is particularly well-fitted for user cold-start recommendation, can also be used in the classical collaborative filtering problem, without constraining the set of items. In that case, the objective function can be written as:

$$\mathcal{L}^{warm}(\mathbf{q}, \Psi) = \sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T(\Psi_0 + \sum_{(u,i)\in\mathcal{O}} \Psi_i^{r_{u,i}}))^2 \quad (8)$$

which can be easily optimized through gradient descent. This model is a simple alternative to matrix factorization-based approaches, which is also evaluated in the experimental section. This model have some nice properties in comparison to transductive techniques, mainly it can easily update users' representations when faced with new incoming ratings.

## 2.3 IAM from cold-start to warm collaborative filtering

We presented in the previous sections how the IAM can be used to tackle each problem of cold-start and warm CF separately. From this, the model can be easily adapted to bridge cold and warm context without additional retraining. This is a crucial and interesting aspect, as it prevents from having two distinct models (as it is often done). The IAM model can be used in a unified way, with the ability of handling the transition between one setting (cold-start) to

the other (warm). This can be done by simply changing the learning paradigm to a two-steps process as follow:

1. Learn the $\mathbf{q}$ and $\Psi$ parameters by optimizing the $\mathcal{L}^{warm}(\mathbf{q}, \Psi)$ from Equation 8.
2. Consider the function $\mathcal{L}^{cold}_{\mathbf{q},\Psi}(\alpha)$ being defined as $\mathcal{L}^{cold}(\mathbf{q}, \Psi, \alpha)$ in Equation 7, but with parameters $\mathbf{q}, \Psi$ being fixed. Learn the alpha parameters for the cold-start interview process by optimizing on $\alpha$ the newly defined $\mathcal{L}^{cold}_{\mathbf{q},\Psi}(\alpha)$.

In the cold-start case, the number of items selected is constrained by the $\alpha$ parameters, which naturally prevents any extreme variation of the representation's norm. In the *cold to warm* case, the number of items from which the representation is computed is expected to vary through time, and its norm might be unbounded, which could have undesirable effects. We therefore propose to pass the representation computed by Equation 4 through an hyperbolic tangent function to limit its norm[3], which is applied in both steps of the learning process describes above. Then, learning the $\alpha$ parameters is done by optimizing the following loss function:

$$\mathcal{L}^{cold}_{\mathbf{q},\Psi}(\alpha) = \sum_{(u,i)\in\mathcal{O}} (r_{u,i} - q_i^T tanh((\Psi_0 + \sum_{(u,i)\in\mathcal{O}} \alpha_i \Psi_i^{r_{u,i}})))^2 + \lambda|\alpha| \tag{9}$$

After the interview, each new incoming rating modifies the user representation as explained in Equation (4), resulting in a system that is naturally able to take into account new information.

# 3 Experiments

We evaluate our models on four benchmark datasets - Table 1a - of various size in terms of number of users, of items or regarding the sparsity of ratings. The datasets are classical datasets used in the literature ([Zhou et al., 2011],[Golbandi et al., 2010]). **ML1M** corresponds to the *MovieLens 1 million* dataset and **Yahoo** corresponds to the *Yahoo! Music* benchmark. **Flixter** and **Jester** are classical datasets. As our main goal is mainly to evaluate the quality of our approach in the context of *new users* arriving in the system, we define the following protocol in order to simulate a realistic interview process on incoming users, and to evaluate different models. We proceed as follow: (i) We randomly divide each dataset along users, to have a pool of *training* users denoted $\mathcal{U}^{train}$, composed of 50% of the users of the complete dataset, on which we learn our model. The remaining users are split in two sets ( representing each 25% of initial users) for validation and testing. The interview process will be applied on each of these two subsets. (ii) The $\mathcal{U}^{test}$ and $\mathcal{U}^{valid}$ sets are then randomly split in two subsets of ratings to simulate the possible known answers : 50% of the ratings of a set are used as the possible answers to the interview questions (*Answer Set*). The 50% of ratings left will be used for evaluating our models (*Evaluation Set*). Ratings have been binarized for each datasets, a rating of -1 (resp. 1) being considered a dislike (resp. like).

The quality of the different models is evaluated by two different measures. The *root mean squared error* (RMSE) measures the average ratings' prediction precision measured as the difference between predicted and actual ratings $(\hat{r}_{u,i} - r_{u,i})^2$. As we work with binary ratings, we also use the accuracy as a performance evaluation. In this context, it means that we focus on the overall prediction, i.e on the fact that the system has rightly predicted *like* or *dislike*, rather than on its precision regarding the "true" rating. The accuracy is calculated as the average "local" accuracy along users. These measures are computed over the set of missing ratings i.e the *Evaluation Set*.

We explore the quality of our approach on both the classical CF context using the IAM Model (Equation 8) and on the cold-start problem using the CS-IAM model defined in Equation 7. We compare our models with two baseline collaborative filtering methods: Matrix Factorization (MF) that we presented earlier, and the Item-KNN with Pearson correlation measure ([Koren, 2010]) which does not compute representations for users nor items but is a state-of-the-art CF method. Note that the inductive models (IAM and CS-IAM) are trained using only the set of training users $\mathcal{U}^{train}$. The ratings in the *Answer Sets* of $\mathcal{U}^{test}$ and $\mathcal{U}^{valid}$ are only taken as inputs during the testing phase, but not used during training. Transductive models are trained using both the training users $\mathcal{U}^{train}$, but also the *Answer sets* of ratings defined over the testing users. It is a crucial difference as our model has significantly less information during training.

Each model has its own hyper-parameters to be tuned: the learning-rate of the gradient descent procedure, the size $N$ of the latent space, the different regularization coefficients... The evaluation is thus made as follows: models are evaluated for several hyper-parameters values using a grid-search procedure, the performance being averaged over 3 different randomly initialized runs. The models with the best average performance on validation set are selected and the respective results on the test set are presented in the next figures and tables. All models have been evaluated over the same datasets splits.

[3]Experiments were conducted with the tangent function in the two other cases, resulting in similar results

Table 1: Table (a) shows the datasets description. Table (b) shows the accuracy performance of different models in the classical Collaborative Filtering context (i.e without cold-start). NA (Not Available) means that, due to the complexity of ItemKNN, results were not computed over the Flixter dataset.

(a) Description of the datasets

| DataSet | Users | Items | Ratings |
|---|---|---|---|
| **ML1M** | 5,954 | 2,955 | 991,656 |
| **Flixter** | 35,657 | 10,247 | 7,499,706 |
| **Jester** | 48,481 | 100 | 3,519,324 |
| **Yahoo** | 15,397 | 1000 | 311,672 |

(b) Accuracy in the warm CF context.

| DataSet | MF | IAM | ItemKNN |
|---|---|---|---|
| Jester | 0.723 | **0.737** | 0.725 |
| ML1M | 0.689 | **0.727** | 0.675 |
| Yahoo | 0.675 | 0.719 | **0.726** |
| Flixter | **0.766** | 0.758 | NA |

## 3.1 Collaborative Filtering

First, we evaluate the ability of our model to learn relevant representations in a classical CF context. In that case, the IAM model directly predicts ratings based on the ratings provided by a user. Results for the four different datasets are presented in Table 1b. We can observe that, despite having much less information during the learning phase, IAM obtains competitive results, attesting the ability of the additive model to generalize to new users. More precisely, IAM is better than MF on three out of four datasets. For example, on the MovieLens-1M dataset, IAM obtains 72.7% in terms of accuracy while MF's accuracy is only 68.9%. Similar scores are observed for Jester and Yahoo. Although ItemKNN model gives slightly better results for one dataset, one should note that this method do not rely on nor provide any representations for users or items and belongs to a different family of approach. Moreover, ItemKNN - which is based on a KNN-based method - has a high complexity, and is thus very slow to use, and unable to deal with large scale datasets like Flixter on which many days are needed in order to compute performance. Beyond its nice performance IAM is able to predict over a new user in a very short-time, on the contrary to MF and ItemKNN.

## 3.2 Cold-start Setting

We now study the ability of our approach to predict ratings in a realistic cold-start situation. As MF and ItemKNN do not provide a way to select a set of items for the interview, we use two benchmark static selection methods used in the literature ([Rashid et al., 2002]). The **POP** method selects the most popular items - i.e the items with the highest number of ratings in the training set - and the **HELF** (*Harmonic mean of Entropy and Logarithm of rating Frequency*) method which selects items based on both their popularity but also using an entropy criterion, which focus on the *informativeness* of items (e.g a controversial movie can be more informative than a movie liked by everyone) ([Rashid et al., 2008]). Our model is learned solely on the $\mathcal{U}^{train}$ set. Baselines are computed on a dataset composed of the original $\mathcal{U}^{train}$ ratings with the additional ratings of the *AnswerSets* of $\mathcal{U}^{valid}$ and $\mathcal{U}^{test}$ that lie into the set of items selected by the POP or the HELF approach. As before, transductive approaches use more information during training that our inductive model.

The number of items selected by the CS-IAM model directly depends on the value of the $L_1$ regularization coefficient and several values have been evaluated. In CS-IAM, the number of selected items correspond to the number of non-null $\alpha_i$ parameters. The number of items selected by POP and HELF is manually chosen.

Figures 1 and 2 respectively show RMSE and accuracy results for all models on the Yahoo dataset as a function of the interview size. It first illustrates that ItemKNN approach does not provide good results for RMSE-evaluation, as it is not a *regression*-based method, but is better than MF in terms of accuracy. It also shows that HELF criterion does not seem to be specifically better on this dataset than the POP criterion. For both evaluations, CS-IAM gives better results, for all sizes of interview. It can also be noted that CS-IAM also gives good results when no item is selected due to the $\Psi_0$ parameters that correspond to the learned default representation. The model with 0 items also expresses the base performance obtained on users unable to provide ratings during the interview.

Detailed accuracy results for the four datasets are summarized in Table 2, for different reasonable sizes of interview. Similar observations can be made on the results, where CS-IAM managed to have the best or competitive accuracy for all datasets and all number of questions allowed, while using less information in train.

At last, when comparing the performance of CS-IAM with a version of IAM where items have been selected by the POP criterion -IAM-Pop, Figure 3 - one can see that the CS-IAM outperforms the other approaches. It interestingly shows that (i) IAM managed to give better results than MF with the same information selection strategy (POP) (ii) CS-IAM with all

its parameters learned, managed to select more useful items for the interview process, illustrating that the performance of this model is due to both, its expressive power, but also on its ability to simultaneously learn representations, and select relevant items.

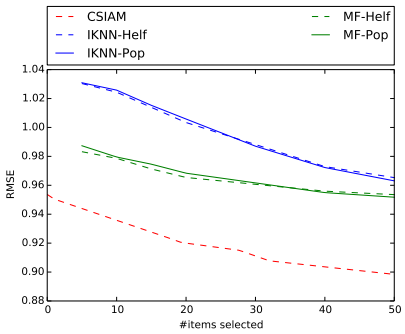


Figure 1: RMSE performance on **Yahoo** dataset for all models, regarding the size of the interview (number of questions/items asked)

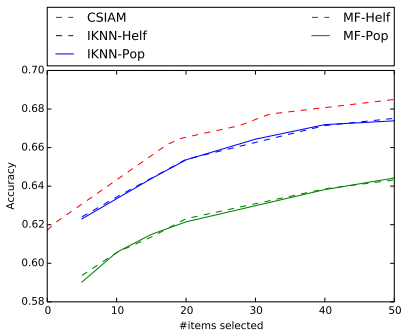


Figure 2: Accuracy performance on **Yahoo** dataset for all models, regarding the size of the interview (number of questions/items asked)

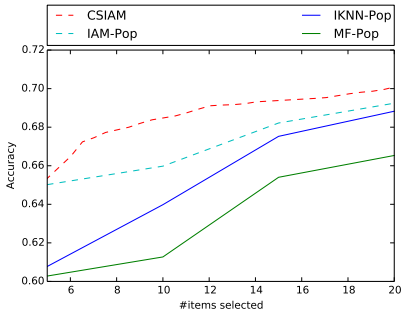


Figure 3: Accuracy performance on **Jester** dataset comparing the **Pop** selection critera and the CS-IAM selection.

We have shown that our approach gives significantly good quantitative results. We now focus our interest on a *qualitative* analysis of the results performed over the MovieLens dataset. First, we compare the items selected by the three selection methods (CS-IAM, POP and HELF). These items are presented in Table 3. First, when using the POP criterion, one can see that many redundant movies are selected - i.e the three last episodes of Star Wars on which the ratings are highly correlated: a user likes or dislikes Star Wars, not only some episodes. The same effect seems to appear also with CS-IAM which selects *Back to the future I* and *Back to the future III*. But, in fact, the situation is different since the ratings on these two movies have less correlations. Half of the users that like *Back to the future I* dislike *Back to the future III*.

Figure 4 shows the translations $\alpha_i \Psi_i$ after having performed a PCA in order to obtain 2D representations. What we can see is that depending on the movie, the fact of having a positive rating or a negative rating does not have the same consequences in term of representation: For example, liking or disliking *Saving Private Ryan* is different than liking or disliking *Star Wars*; the translation concerning these two movies are almost perpendicular and thus result in a very different modification of the representation of the user. *Schindler's List* has less consequences concerning the user representation i.e the norm of $\alpha_i \Psi_i^r$ is lower than the others.

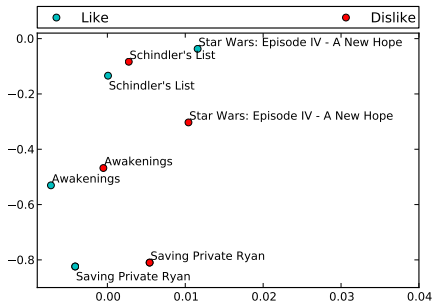


Figure 4: Visualization of some $\alpha_i \Psi_i$ after a PCA, on dataset MovieLens-1M

## 3.3 Mixing Cold-start and Warm Recommendation

We propose in this section to use our approach in a context where one want to move from *cold-start* to *warm* recommendation. After having answered the interview, the new user will start interacting with the system, eventually providing new ratings on its own. We follow the two-steps learning process presented in Section 2.3 to address this specific task. This approach is evaluated on the **Yahoo** dataset with the following experimental protocol being applied after learning:

Table 2: Accuracy performance of models on four datasets regarding the number of questions asked. NA (Not Available) means that, due to the complexity of ItemKNN, results were not computed over the Flixter dataset. Bold results corresponds to best accuracy.

| DataSet | NbItems | MF POP | MF HELF | IKNN POP | IKNN HELF | CS-IAM |
|---|---|---|---|---|---|---|
| Jester | 5 | 0.603 | 0.589 | 0.608 | 0.634 | **0.667** |
| | 10 | 0.613 | 0.609 | 0.640 | 0.608 | **0.686** |
| | 20 | 0.665 | 0.641 | 0.688 | 0.676 | **0.701** |
| MovieLens 1M | 5 | 0.629 | 0.617 | 0.649 | 0.647 | **0.690** |
| | 10 | 0.634 | 0.620 | 0.651 | 0.653 | **0.695** |
| | 20 | 0.648 | 0.621 | 0.663 | 0.638 | **0.696** |
| Yahoo | 5 | 0.590 | 0.594 | 0.623 | 0.624 | **0.638** |
| | 10 | 0.601 | 0.610 | 0.633 | 0.634 | **0.647** |
| | 20 | 0.621 | 0.623 | 0.654 | 0.654 | **0.665** |
| Flixter | 5 | 0.719 | 0.722 | NA | NA | **0.723** |
| | 10 | 0.720 | 0.726 | NA | NA | **0.727** |
| | 20 | 0.727 | **0.739** | NA | NA | 0.735 |

Table 3: MovieLens 1M - Selected items for the interview process by the three selection methods.

| **CS-IAM** | **Popularity** | **HELF** |
|---|---|---|
| American Beauty | American Beauty | Jurassic Park |
| Being John Malkovich | Star Wars: Episode I | Independence Day |
| Lion King | Star Wars: Episode V | Men in Black |
| Ghost | Star Wars: Episode IV | Total Recall |
| Superman | Star Wars: Episode VI | Mission: Impossible |
| Back to the Future | Jurassic Park | Speed |
| Fargo | Terminator 2 | Face/Off |
| Armageddon | Matrix | Who Framed Roger Rabbit? |
| Get Shorty | Back to the Future | Abyss |
| Splash | Saving Private Ryan | Austin Powers |
| 20 000 Leagues Under the Sea | Silence of the Lambs | Beetlejuice |
| Back to the Future Part III | Men in Black | Titanic |

1. Performance is firstly measured in the cold-start setting, using the items with non-null $\alpha$'s values for the interview process, as in Section 3.2.

2. We calculate the performance of this model when adding increasing amount of "new" ratings sampled uniformly from the set of real available ratings left in the *Answer Set*.

The results are shown in Figure 5, which illustrates the accuracy given the percentage of additional ratings taken in the ratings left after the interview process, for three different sizes of initial interviews. This shows that this strategy starts (0% of additional ratings used) with a good accuracy performance, consistent with the results obtained in Table 2 on the strict cold-start context. The accuracy then increases as new ratings are added and almost reaches the one obtain for the classical warm setting (see Table 1b).

This extension of our approach makes the link between the cold-start and the warm settings, which is an original and promising feature, and is novel compared to other classical approaches, which usually focus on either the cold-start or the classical warm context, and potentially need (extensive) additional computation to go from one to the other (typically for transductive models such as MF).

# 4 Related Work

The recommendation problem has been studied under various assumptions. We focus here on **Collaborative Filtering** (CF) methods, which relies only on interactions between users and items, such as ratings or purchase history. Other families of approaches exists, such as *Content-Based* methods, which use informative features

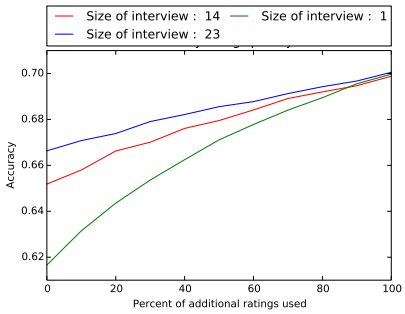


Figure 5: Accuracy regarding percentage of ratings added after the interview (from cold-start to warm setting).

on users and items ([Pazzani and Billsus, 2007]), and hybrid methods that mix ratings and informative features ([Basilico and Hofmann, 2004]).

CF techniques can be distinguished into two categories. *Memory-based* methods, such as Neighbor-based CF [Resnick et al., 1994], calculate *weights* between pairs of items ([Sarwar et al., 2001]) or users ([Herlocker et al., 1999]), based on similarities or correlations between them. *Model-based* methods, such as *Latent Factor Models*, have rather a *representation learning* approach, where representations vectors for each user and item are inferred from the matrix of ratings with matrix factorization techniques ([Koren et al., 2009]). However, collaborative filtering models have a major limitation when there is no history for a user or an item, the cold-start case. While specific similarities have been designed to address this problem for memory-based models ([Bobadilla et al., 2012], [Ahn, 2008]), a classical and intuitive approach is to use an interview process with a few questions asked to the new user, as it is done in this paper. Several papers have proposed different methods to choose which questions to select.

**Static approaches** (see [Rashid et al., 2002] for a comparative study), construct a static seed set of questions (fixed for all users) following a selection criterion like measures of popularity, entropy or coverage while [Golbandi et al., 2010] also proposed a greedy algorithm that aims to minimize the prediction error performed with the seed set.

**Adaptive approaches** have also been proposed, where the interview process considers the user's answers to choose the next question. For example, [Rashid et al., 2008] fits a decision tree to find a set of clusters of users, while [Golbandi et al., 2011] uses a ternary tree where each node is an item and branch corresponds to eventual answers (*like,dislike,unknown*). [Zhou et al., 2011] presents *functional matrix factorization*, a decision tree based method which also associate a *latent profile* to each nodes of the tree. The closest model to our approach is [Sun et al., 2013], who learn a ternary tree allowing multiple questions at each node, each node containing a (learned) regressor and translations functions on selected items. Our model can be seen as one node of their tree. However, their approach does not seem to allow a bridge between cold start and warm context as ours does.

It is also interesting to note that while usually more efficient, one drawback of such adaptive approaches is that users usually dislike having to rate item *sequentially* and prefer rating several items in one shot ([Golbandi et al., 2011], [Rashid et al., 2002]).

Another possible strategy is the use of *Active Learning* approaches (see [Rubens et al., 2011] for a general "foray" into Active Learning in recommender system). For example, [Jin and Si, 2004] proposed a Bayesian Selection approach which has been extended by [Harpale and Yang, 2008] to tackle the user cold start problem of selecting the ratings to ask for.

# 5 Conclusion and Perspectives

We proposed a novel representation-learning based model for partially observable data and information selection. More precisely, we focused on collaborative filtering, which is an intuitive and emblematic problem of sparse data arriving through time. The inductive model we presented (IAM) directly computes the representation of a user by cumulative translations in the latent space, each translation depending on a rating value on a particular item. We have also proposed a generic formulation of the user cold-start problem as a representation learning problem and shown that the IAM method can be instantiated in this framework allowing one to learn both which items to use in order to build a preliminary interview for incoming users, but also how to use these ratings for recommendation. The results obtained over four datasets show the ability of our approach to outperform baseline methods. Different research directions are opened by this work: (i) first, the model can certainly be extended to deal with both incoming users, but also new items. In that last case, the interview process would consist in asking reviews for any new item to a particular subset of relevant users. (ii) While we have studied the problem of building a static interview - i.e the opinions on a fixed set of items is asked to any new user - we are currently investigating how to produce personalized interviews by using sequential learning models i.e reinforcement learning techniques.

# References

[Ahn, 2008] Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-

starting problem. *Information Sciences*, 178(1):37–51.

[Basilico and Hofmann, 2004] Basilico, J. and Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9. ACM.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

[Bobadilla et al., 2012] Bobadilla, J., Ortega, F., Hernando, A., and Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238.

[Carpenter, 2008] Carpenter, B. (2008). Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. *Alias-i, Inc., Tech. Rep*, pages 1–20.

[Golbandi et al., 2010] Golbandi, N., Koren, Y., and Lempel, R. (2010). On bootstrapping recommender systems. In *Proceedings of the 19th ACM CIKM*, pages 1805–1808. ACM.

[Golbandi et al., 2011] Golbandi, N., Koren, Y., and Lempel, R. (2011). Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 595–604. ACM.

[Harpale and Yang, 2008] Harpale, A. S. and Yang, Y. (2008). Personalized active learning for collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98. ACM.

[Herlocker et al., 1999] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR*, pages 230–237. ACM.

[Jin and Si, 2004] Jin, R. and Si, L. (2004). A bayesian approach toward active learning for collaborative filtering. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 278–285. AUAI Press.

[Koren, 2010] Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Pazzani and Billsus, 2007] Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.

[Rashid et al., 2002] Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., and Riedl, J. (2002). Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th IUI*, pages 127–134. ACM.

[Rashid et al., 2008] Rashid, A. M., Karypis, G., and Riedl, J. (2008). Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

[Rubens et al., 2011] Rubens, N., Kaplan, D., and Sugiyama, M. (2011). Active learning in recommender systems. In Kantor, P., Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 735–767. Springer.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW*, pages 285–295. ACM.

[Sun et al., 2013] Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., and Zha, H. (2013). Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 445–454. ACM.

[Zhou et al., 2011] Zhou, K., Yang, S.-H., and Zha, H. (2011). Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324. ACM.