Focused Matrix Factorization For Audience Selection in Display Advertising

Bhargav Kanagal [1] Amr Ahmed [1] Sandeep Pandey [3] Vanja Josifovski [1] Lluis Garcia-Pueyo [1] Jeff Yuan [2]

[1] {*bhargav,amra,vanjaj,lgpueyo*}*@google.com,* [2] { *yuanjef*}*@yahoo-inc.com,* [3] *spandey@twitter.com*

[1] Google Inc, [2] Yahoo! Research, [3] Twitter Inc

[1,3] *Work done while at Yahoo! Research*

***Abstract*— Audience selection is a key problem in display advertising systems in which we need to select a list of users who are interested (i.e., most likely to buy) in an advertising campaign. The users' past feedback on this campaign can be leveraged to construct such a list using collaborative filtering techniques such as matrix factorization. However, the user-campaign interaction is typically extremely sparse, hence the conventional matrix factorization does not perform well. Moreover, simply combining the users feedback from all campaigns does not address this since it dilutes the focus on target campaign in consideration. To resolve these issues, we propose a novel *focused matrix factorization* model (FMF) which learns users' preferences towards the specific campaign products, while also exploiting the information about related products. We exploit the product taxonomy to discover related campaigns, and design models to discriminate between the users' interest towards *campaign* products and *non-campaign* products. We develop a parallel multi-core implementation of the FMF model and evaluate its performance over a real-world advertising dataset spanning more than a million products. Our experiments demonstrate the benefits of using our models over existing approaches.**

I. Introduction

From its meager beginnings in the 1990's, online display advertising has grown into a $10 billion dollar industry in 2011[1]. Today, an increasing number of advertisers launch conversion-oriented ad campaigns where the goal is to display ads to those users who are most likely to show interest and respond positively to the campaigns (e.g., in terms of product purchases). Hence, the problem for the advertising network is to select such a list of users. This problem is typically referred to as "audience selection" or "audience retrieval" [5], [18].

Traditionally, advertisers have selected audiences using general demographic attributes such as gender, location and age. However, lately, the trend has been to avoid making such vague generalizations, and instead customizing the campaign audience using the set of users that have responded positively to the campaign in the past (as identified by the advertiser). This type of information is reminiscent of the user response matrix in the recommender systems [3], [14], [15], [17] settings. Here, the input data is matrix in which the rows of this matrix represent users, the columns are products, and matrix cells indicate whether a user has purchased a given product. The only difference being that now we need to recommend users for a given campaign product(s), instead of the other way around. For instance, for a *target* ad campaign from an electronics seller that seeks to advertise *a set of electronic items* from that seller, we want to recommend users who, given their past product purchases, are likely to buy electronic items like cameras, cell phones, etc.

The state of the art techniques in recommender systems such as matrix factorization [14], [17] and other approaches could be used to solve the audience retrieval task. However, as we show in our experimental analysis, they perform poorly – this is because the data set (user-product response matrix) in our application consists of real product purchases and is very sparse (on average, one in a million advertisement leads to a product purchase). The problem is even more pronounced for campaigns that are new or small in size in terms of the number of products spanned. One approach to deal with the sparsity challenge is by borrowing information from other ongoing as well as historical advertising campaigns. The key intuition is that by exploiting other campaigns, we can enrich the user-product response matrix, thus leading to better matrix factorization and better audience retrieval in turn. Note that in the extreme case, this would mean considering the complete user-product response matrix using all the present and past campaigns to perform audience retrieval for the given target campaign. However, as shown later in the paper, this does not perform very well since it treats each campaign equally and loses focus on the target campaign in consideration (since the optimization minimizes error over all the campaigns, rather than just this campaign). In other words, it is important to keep focus while borrowing information from other campaigns, thus motivating the *focused matrix factorization* approach proposed in the paper.

There are several challenges that need to be addressed in building such a framework. First, the information should be borrowed from those campaigns which are similar to the target campaign while ignoring those which are not. Second, the campaign *similarity and dis-similarity* must themselves be learned using the sparse response matrices. Third, the information transfer needs to be adaptive, i.e., when the target campaign is new and does not have enough response values of its own, it should borrow more information from others (i.e., cold start). But as the campaign gets older and accrues more data, it should gradually decrease the information transfer and maintain its specificity. Fourth, the approach must scale to millions of users and products as is the case with real world advertising campaigns.

[1]According to emarketer.com, including video advertising.

Our Approach

In this paper we propose a novel focused matrix factorization technique, FMF model, which learns users' preferences towards the specific campaign products while also exploiting the information about related campaigns/products. More specifically, our FMF model advances the existing audience retrieval systems in the following way. We discriminate between the users' preferences in the target campaign, which we call *focus preferences*, and the *non-focus preferences*. By this, akin to the transfer learning approaches [19], we allow for focusing on the target campaign, while also drawing from the data available in other campaigns. In particular, we learn two different sets of user preferences/factors as well as their importance towards predicting the user response to the given campaign. Simultaneously, we estimate the similarity/dis-similarity between the target campaign and other campaigns using the user-product response matrix to facilitate the information transfer. Also, we show how a taxonomy over the products can be integrated with our approach to aid in this task.

We develop an efficient parallel multi-core implementation of our FMF model and evaluate its performance over a real-world shopping dataset spanning more than a million users and products. Such implementation requires different processing threads to share the model parameters with both parameter reads and updates going to shared values. This leads to bottlenecks with some of the model parameters that are updated more frequently. To resolve this issue we provide an update caching scheme that batches the changes of individual threads to avoid conflicts. Our experiments demonstrate the significant benefits of using our FMF models over existing approaches.

Contributions

In summary, the contributions of this paper are as following:

- we introduce the Focused Matrix Factorization model where we discriminate between the user interests in the target campaign and other campaigns, while using all the available information.
- we show how FMF can be applied to the audience selection problem in display advertising.
- we have provided a parallelized implementation for multi-core machines that caches the updates of the model parameters to avoid bottlenecks and speeds up the evaluation by orders of magnitude.
- we provide an experimental evaluation over a real-world dataset showing the merits of the proposed approach.

We note here that although we present FMF model in the context of audience selection, the approach is general and is applicable to all the recommendation scenarios. For example, FMF can be used to train models for movie recommendation for specific users or specific genres and so on.

Outline: The rest of the paper is organized as follows. First we formally describe the audience selection problem in Section II. Next, we describe our focused collaborative filtering model in Section III. We illustrate the algorithms to train the model in Section IV. We conclude with experiments in Section VI.

II. Problem Formulation

We start by providing a brief overview of audience selection in display advertising. Subsequently, we formulate it as a collaborative filtering problem.

A. Audience Selection

Display advertising constitutes graphical ads and media that are displayed alongside the main content of the web page. As with traditional advertising, the advertisers' intent is to reach users that might be interested in their products. In particular, the *performance* oriented advertisers are interested to target users that are most likely to *convert*, i.e, make a purchase corresponding to the advertisement. To this end the advertiser launches a *campaign* with the advertising network promoting its products. The advertising network now needs to select the set of users as desired by the advertiser. This task of retrieving relevant users by the advertising network is referred to as *audience selection*. For instance, an electronics company may launch a campaign with the network, targeting users who buy portable media players. A commonly used approach by the advertising networks is to select users who have previously converted on media players and related electronics goods.

B. Matrix Factorization for Audience Selection

As described before, each campaign consists of a set of items (products) that are being advertised as part of the campaign. For each item the advertising network maintains the set of customers that have bought it. In other words, we have a sparse (partially populated) response matrix for campaign c, say $\boldsymbol{X}^c$, where $\boldsymbol{X}^c(i,j) = 1$ if user i has purchased item j. Given a item p, the audience selection problem requires us to select the set of users that are most likely going to purchase p. One such approach is using *matrix factorization* [3], [16], [17], [21]. By factorizing this response matrix, we project each user and item into a lower dimensional space (such lower dimensional projections are called *latent factors*) which essentially clusters similar users and items together; subsequently we measure the *affinity* between a user and an unpurchased item as the dot product between the corresponding factors and finally select the top-k users with the highest affinity for each item. We explain how to learn the user and item factors given the historical purchases data.

The input to the audience selection system is the user-item response matrix X^c (size $m \times n$, m users and n items in the campaign c). The goal of the system is to predict, for each product p, a ranked list of users in the matrix. We assume that each user u can be represented by a latent factor $\mathbf{v_u^U}$ which is a vector of size $1 \times K$ (K is commonly referred to as the number of factors in the model, typically much smaller than m and n). Similarly, each item i can be represented by a latent factor $\mathbf{v_i^I}$ (also, a vector of size $1 \times K$). User u's affinity/interest in item i is assumed to follow this model:

$$\hat{x}_{ui} = \langle \mathbf{v_u^U}, \mathbf{v_i^I} \rangle$$

Here, x_{ui} is the affinity of user u to item i, $\langle \mathbf{v_u}, \mathbf{v_i} \rangle$ represents the dot product of the corresponding user and item factors. The

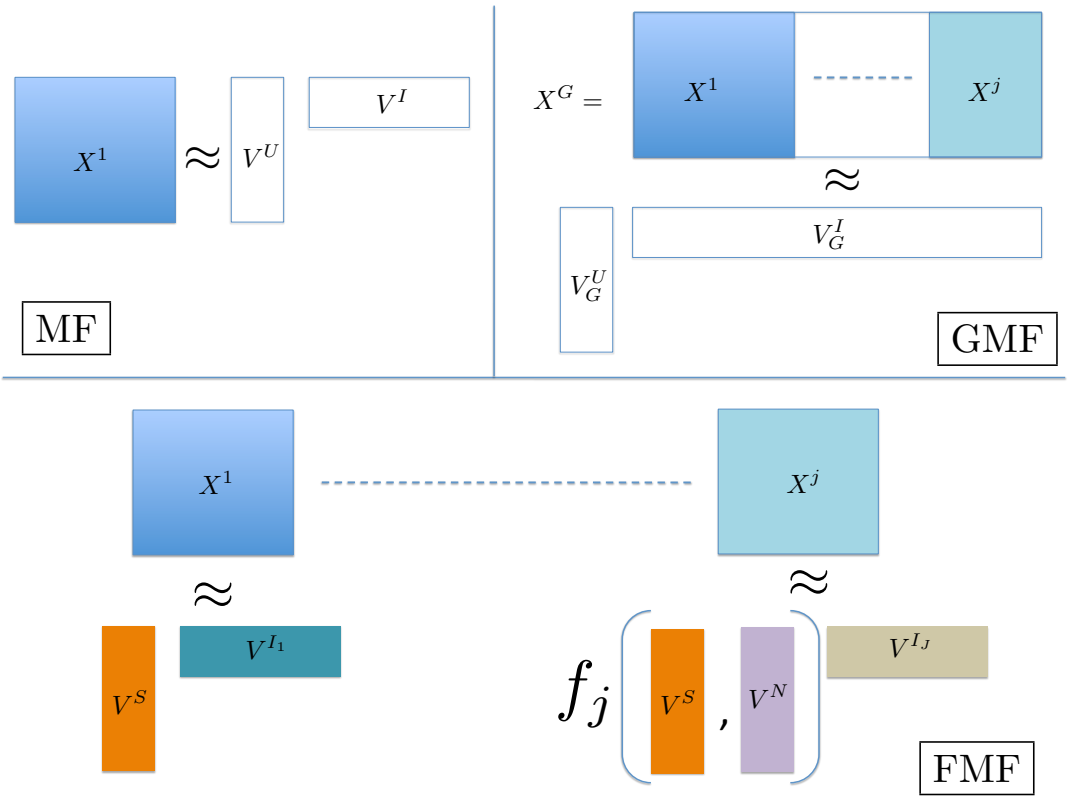


Fig. 1: Illustrating the different approaches considered in this paper. X^1 is the user-item matrix for the target campaign. $X^2 \cdots X^J$ are the user-item matrices for the non-target campaigns. In the Matrix Factorization (MF) approach, we just factorize the target campaign to get the user and item factors. In this case we do not utilize any non-target campaigns data. In the GMF approach, we poll together data from all campaigns, to get the global user-item matrix X^G. We then factorize this matrix to get the user and item factors, V_G^U and V_I^G, respectively. In the last approach, Focused Matrix Factorization (FMF), each user has two two factors, a target campaign factor V^S and non-target campaign factors V^N. The target campaign user-item matrix, X^1, is factorized using user factors V^S and item factors V^{I_1}. Each non-target campaign user-item matrix X^j is also factorized using user factors $f_j(V^S, V^N)$, and item factors V^{I_j}. The user factors $f_j(V^S, V^N)$ for non-target campaign j combines the V^S which is shared with the target-campaign and V^N which is not shared with the target campaign. The combination function $f_j()$ depends on the campaign j such that the more similar it is to the target campaign, the more pronounced the role of V^S int this combination and as such information is transfered into the target campaign. In the text, we will discuss three different models, each of which utilizes different form of the function $f_j()$.

learning problem here is to determine the best values for $\mathbf{v_u^U}$ and $\mathbf{v_i^I}$ (for all users u and all items i) based on the given rating matrix Y, these parameters are usually denoted by Θ.

A widely used approach to learning Θ is Bayesian personalized ranking (BPR) proposed by Rendle et al. [20]. In BPR, the goal is to discriminate between items bought by the user and items that were not bought. In other words, we need to learn a ranking function R_i for each item i that ranks i's interesting users higher than the non-interesting users. In other words, if user u_1 has purchased item i and user u_2 has not purchased the item, we must have $R_i(u_1) > R_i(u_2)$. For this, we need to have: $x_{u_1 i} > x_{u_2 i}$. Based on the above arguments, our likelihood function $p(R_i|\Theta)$ is given by:

$$p(R_i|\Theta) = \prod_{i \in I} \prod_{u_1 \in L_i} \prod_{u_2 \notin L_i} \sigma(x_{u_1 i} - x_{u_2 i})$$

Here, L_i is the list of users who have purchased item i.

Following Rendle et al. [20], we have approximated the non-smooth, non-differentiable expression $x_{u_1 i} > x_{u_2 i}$ using the logistic sigmoid function $\sigma(x_{u_1 i} - x_{u_2 i})$, where $\sigma(z) = \frac{1}{1+e^{-z}}$. We use a Gaussian prior $N(0, \sigma)$ over all the factors in Θ and compute the MAP ((maximum aposteriori) estimate of Θ. The posterior over Θ (which needs to be maximized) is given by:

$$\begin{aligned} p(\Theta|R_i) &= p(\Theta)p(R_i|\Theta) \\ &= p(\Theta) \prod_{i \in I} \prod_{u_1 \in L_i} \prod_{u_2 \notin L_i} \sigma(x_{u_1 i} - x_{u_2 i}) \end{aligned}$$

We need to maximize the above posterior function, (or its log-posterior), shown below.

$$\log p(\Theta|R_i) = \sum_i \sum_{u_1 \in L_i} \sum_{u_2 \notin L_i} \ln \sigma(x_{u_1 i} - x_{u_2 i}) - \lambda ||\Theta||^2$$

The first summation term corresponds to the log-likelihood, i.e., $\log p(R_i|\Theta)$ whereas the second term corresponds to the log of the Gaussian-prior, i.e., $\log p(\Theta)$. Here, λ is a constant, proportional to $\frac{1}{\sigma^2}$. $||\Theta||^2$ is given by the following expression:

$$||\Theta||^2 = \sum_u ||v_u^{\mathrm{U}}||^2 + \sum_i ||v_i^{\mathrm{I}}||^2$$

The second term is commonly called as the *regularization* term, and is used to prevent overfitting by keeping the learned factors v_u^{U} and v_i^{I} sparse.

1) Stochastic Gradient Descent (SGD): SGD is typically used to optimize objective functions that can be written as sums of (differentiable) functions, e.g., in the objective function above, we have one function per training data point (i, u_1, u_2). The standard gradient descent method is an iterative algorithm: Suppose that we want to maximize a given objective function. In each iteration, we compute the gradient of the function and update the arguments in the direction of the gradient. However, computing the overall gradient requires computing the derivative for each function in the summation and is quite expensive when we have a large training dataset. In SGD, we approximate the derivative by computing it only at a single (randomly chosen) term in the summation and update the arguments in this direction. Despite this approximation, SGD has been shown to work very well in practice [9], [7], [8], often outperforming other methods including the standard gradient descent. In the above example, a given training data point (i, u_1, u_2) defines a term in the summation. The derivatives with respect to the v_i^{I}, $v_{u_!}^{\mathrm{U}}$ and $v_{u_2}^{\mathrm{U}}$ variables are shown below. Denote c_{i,u_1,u_2} to be equal to $(1 - \sigma(x_{u_1 i} - x_{u_2 i}))$.

$$\begin{aligned}
\frac{\partial L(U,V)}{\partial \mathbf{v_i^I}} &= c_{i,u_1,u_2}(\mathbf{v_{u_1}^U} - \mathbf{v_{u_2}^U}) - \lambda \mathbf{v_i^I} \\
\frac{\partial L(U,V)}{\partial \mathbf{v_{u_1}}} &= c_{i,u_1,u_2}\mathbf{v_i^I} - \lambda \mathbf{v_{u_1}^U} \\
\frac{\partial L(U,V)}{\partial \mathbf{v_{u_2}}} &= -c_{i,u_1,u_2}\mathbf{v_i^I} - \lambda \mathbf{v_{u_2}^U}
\end{aligned}$$

Now, we use the above derivatives to update the appropriate factors. Note that since we have a maximization problem, we need to move in the same direction as the derivative. The corresponding update equations are shown below:

$$\begin{aligned}
\mathbf{v_i^I} &= \mathbf{v_i^I}(1 - \epsilon\lambda) - \epsilon c_{i,u_1,u_2}(\mathbf{v_{u_1}^U} - \mathbf{v_{u_2}^U}) \\
\mathbf{v_{u_1}^U} &= \mathbf{v_{u_1}^U}(1 - \epsilon\lambda) - \epsilon c_{i,u_1,u_2}\mathbf{v_i^I} \\
\mathbf{v_{u_2}^U} &= \mathbf{v_{u_2}^U}(1 - \epsilon\lambda) + \epsilon c_{i,u_1,u_2}\mathbf{v_i^I}
\end{aligned}$$

Here, ϵ is the learning rate which is set to a small value. The regularization term λ is usually chosen via *cross-validation*. An exhaustive search is performed over the choices of λ and the best model is picked accordingly. The overall algorithm proceeds as follows. A training data point (u, i, j) is sampled uniformly at random. The gradients are computed at this particular data point and the variables are modified according to the update rules shown below. An *epoch* is roughly defined as a complete pass over the data set (i.e., over all the non-zero entries in the rating matrix). By deriving the factors for users and items from solving the above optimization problem, we can perform audience selection for campaign c. In particular, for each product j in campaign c, we compute the dot product $(\langle v_u^U, v_i^I \rangle)$ for every user and recommend the ones with the highest values. We refer to this approach in the rest of the paper by Matrix Factorization (`MF`).

While the `MF` approach works well when the campaign has a rich response matrix, it is known to struggle in the presence of sparsity. This is particularly apparent in our case because (a) most tail campaigns are small in terms of the number of items and have a "narrow" response matrix, (b) and for new campaigns where we do not have enough historical purchase data (i.e., cold-start problem). One possible approach to mitigating these issues is to borrow information from other campaigns. We describe this in the next section.

III. Focused Matrix Factorization Model

By bringing in other campaigns, we can enrich the user-product interaction matrix and find better insights into user preferences. For instance, if a user often buys items from campaigns on computer/gaming accessories, then her buying pattern can be leveraged for performing audience selection for related campaigns on cell phones, music players etc.

`GMF`: Global Matrix Factorization

A simple way to borrow information is by combining the response matrix of all campaigns. In other words, we construct *global* matrix $\boldsymbol{X}^G$ which includes user-product data from all campaigns. The global response matrix is dense and can be factorized under the BPR framework (as described before) to derive the item and user factors, V_U^G and V_I^G, respictively. Then, given any target campaign, these factors can be used to perform audience selection. We refer to this approach by *Global Matrix Factorization* (`GMF`). Note that in this approach, we learn one set of user factors in this approach, irrespective of the target campaign, unlike the `MF` approach, where we learn user factors for each target campaign separately.

While `GMF` resolves the sparsity issue, it has other drawbacks. Since we are factorizing the global matrix, the derived user factors represent the global user preferences, e.g., user buys electronics often. Such global preferences give us insights into the general buying pattern of the user, but they do not capture the campaign-specific user preferences accurately. This is especially true for the small campaigns which are likely to be overwhelmed by large campaigns in the global matrix. (Note that the whole idea of borrowing information is to help the small campaigns.) Hence, we propose our *focused collaborative filtering* models which allow us to appropriately borrow relevant information from other campaigns while still retaining focus on the target campaign.

A. Focused Matrix Factorization Models (`FMF`)

We present three focused collaborative filtering models `FMF1`, `FMF2` and `FMF3` with varying degrees of sharing between campaigns. Subsequently, we present details about how to incorporate a taxonomy over the products into our `FMF` framework.

`FMF1`

Let T denote the items in the target campaign for which we want to perform audience selection. Let N denote the items in the campaigns that are not part of the target (i.e., non-target campaigns, $T \cup N = I$). This includes the campaigns which are currently running as well as old campaigns which were run in the past. The key idea behind `FMF1` is that instead of learning one set of user preferences (i.e., factors) like `GMF`, we allow each user to have two sets of preferences: one set

for the target campaign (*focus preferences*) and the other for the non-target campaigns (*non-focus preferences*). We define this formally next.

Let v_u^1 and v_u^2 denote the factors for user u for the target and non-target campaigns respectively. To allow information transfer between the target and non-target campaigns, we constrain these factors in the following manner: $v_u^1 = v_u^S + v_u^T$ and $v_u^2 = v_u^S + v_u^N$. Here v_u^S captures the information that can be *shared* between the target and non-target campaigns, v_u^T captures the residual specific to the target items and v_u^N for the non-target items. Then, the affinity x_{ui} between user u and item i in the FMF1 model is written as:

$$x_{ui} = \begin{cases} \langle v_u^S + v_u^T,\ v_i^I \rangle & \text{if } i \in T \\ \langle v_u^S + v_u^N,\ v_i^I \rangle & \text{if } i \in N \end{cases}$$

Our FMF1 model is stringent in terms of model variables but still general enough to capture both MF and GMF models that were proposed before. For instance, if the target and non-target items are completely independent and do not share any information, then v_u^S can be set to 0 and only v_u^T will be used to compute user affinity for the target items (i.e., the MF model). On the other hand, if the target and non-target items are completely alike, then the residual vectors v_u^T and v_u^N can be set ot 0 and the shared vector, v_u^S, is used for the affinity computation (i.e., the GMF model).

Without loss of generality, we can simplify the FMF1 model by setting v_u^T to 0. Effectively, the shared factors in the new model, say $v_u'^S$, can be thought of as $v_u^S + v_u^T$ and the residual vector v'^N_u as $v_u^N - v_u^T$. Doing this not only reduces the number of factors that need to be estimated, but also makes the model *mathematically identifiable* (by breaking symmetry between the targeted and the non-targeted campaigns) and hence interpretable. Thus, our final FMF1 model is:

$$\boxed{x_{ui} = \begin{cases} \langle v_u^S,\ v_i^I \rangle & \text{if } i \in T \\ \langle v_u^S + v_u^N,\ v_i^I \rangle & \text{if } i \in N \end{cases}}$$

Thus in FMF1, we have $f_j(V^S, V^N) = v_u^S + v_u^N$.

FMF2

While FMF1 captures the sharing between the target campaign and the non-target campaigns (through v_u^S), it borrows identical amounts of information from all the non-target campaigns. This will "confuse" the model if the non-target campaigns are different and represent a diverse set of items. For instance, a target campaign on "electronic" items must borrow more from a non-target campaign on "computers" than the one on "furniture".

Essentially, we need to take into account the similarity between the target and non-target campaigns while sharing information. To allow this, we introduce an α_j variable for every non-target campaign j from which we want to borrow(let N_j be the set of items in the non-target campaign j) to control the degree of sharing. Note that we also need to learn the α values from the data. A large positive value for α_j would indicate a large correlation between the target and the j^{th} non-target campaign, a small value indicates the absence of interaction, and a negative value suggests anti-correlation. The model FMF2 is shown below:

$$\boxed{x_{ui} = \begin{cases} \langle v_u^S,\ v_i^I \rangle & \text{if } i \in T \\ \langle \alpha_j v_u^S + v_u^N,\ v_i^I \rangle & \text{if } i \in N_j \end{cases}}$$

Thus in FMF2, we have $f_j(V^S, V^N) = \alpha_j v_u^S + v_u^N$.
We pictorially illustrate this model using a plate-based graphical model [12] representation in Figure 2. In the user plate, we have random variables v_u^N and v_u^S that denote the user factors. In the item plate, we have random variables α_j and v_i^I that denotes the sharing factor and the item factor respectively. The shaded variable x_{ui} corresponds to the observed entities in the user-item matrix. The other random variables w_j^I correspond to the taxonomy, which we discuss in Section III-B.

FMF3

The FMF2 model allows borrowing information differently from different campaigns by keeping α_j variables. We can further generalize this model to not only have different α_j but also different $v_u^{N_j}$ for each non-target campaign j. In other words, each non-target campaign has its own residual vector to capture its specificity. However, this results in too many user factors (K factors per campaign, multiplied by the number of non-target campaigns) and can be difficult to estimate in practice. To avoid this, we set each $v_u^{N_j}$ to be $(1 - \alpha_j) \cdot v_u^N$ and hence, keep the number of user factors to be $2 \cdot K$ as before.

Essentially, for each campaign, target or non-target, the effective user factor is a linear combination of v_u^S and v_u^N with α_j being the mixture coefficient. Since α_j for the target campaign is 1 (i.e., the campaign is completely correlated with itself), we get the effective user factor for the target campaign to be v_u^S. Thus, the FMF3 model looks like:

$$x_{ui} = \begin{cases} \langle v_u^S,\ v_i^I \rangle & \text{if } i \in T \\ \langle \alpha_j v_u^S + (1 - \alpha_j) v_u^N,\ v_i^I \rangle & \text{if } i \in N_j \end{cases}$$

Thus in FMF3, we have $f_j(V^S, V^N) = \alpha_j v_u^S + (1 - \alpha_j) v_u^N$. Moreover, we restrict α_j to be in the range $[0, 1]$. We pictorially illustrate the FMF model in relation to the MF and GMF approaches in Figure 1.

We would like to point out here that while using additional factors to model user preferences doubles the number of parameters that we need to train, we use appropriate regularization to keep them correlated – thereby the effective number of parameters is much less. As we mention in Section VI-A.4, we use cross-validation to select the best parameters for the FMF models.

B. Leveraging Item Taxonomy

In this section, we show how a taxonomy over products/items can be integrated in the FMF framework. This alleviates the sparsity issue and improves the performance of our models (as shown in the experiments).

As described in Section I, sparsity is a prevalent issue with ad campaigns. Tail campaigns contain very few items

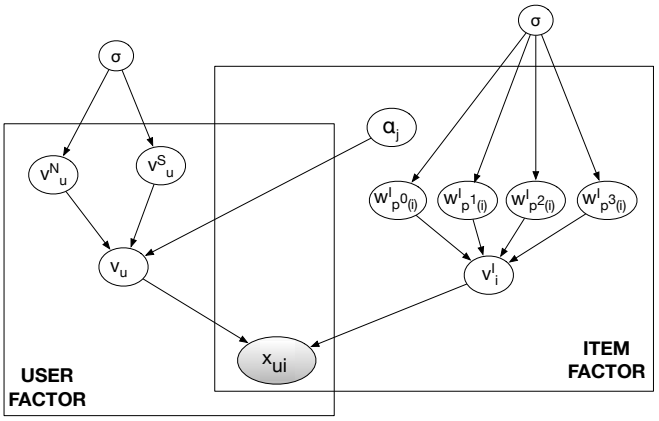


Fig. 2: Graphical model description of FMF2 model. Random variables in the user plate include user factors v_u^N and v_u^S. Random variables in the item plate include item factor v_i^I and the taxonomy-node factors w^I (Section III-B). Note that α_j is in the item plate since the campaign index j is unique for a given item i.

and are therefore sparse as a result, while large campaigns struggle with sparsity when they are new. Our FMF models from Section III exploit information from other campaigns to resolve this problem, but we can further boost the performance of our models by leveraging an additional source of information: taxonomy over the products, which are commonly available [2], [13]. The taxonomy provides lineage for a product in terms of the parent categories that it belongs to, e.g., an Apple iPod is a *portable media player*, which itself is an *electronics* item.

We incorporate taxonomy using a *hierarchical* additive model [10], [11] over the item factors. In particular, we introduce item factors w_j^I for all nodes in the taxonomy (including the actual items, which are considered as leaves of the taxonomy tree) and define the item factors using the following equation:

$$v_i^I = \sum_{\ell=0}^{L} w^I_{p^l(i)}$$

where $p^l(i)$ denotes the l^{th} ancestor of item i. Now, every item factor has a contribution from each of its ancestors. Note that this allows us to derive factors for even those items which may not have any purchases (and are hence not trained) by using the estimate of their parent in the taxonomy. And as we get more purchases for this item, the models allows the item factor to get away from the parent if necessary.

IV. Model Learning and Inference

In this section, we describe algorithms for learning the FMF models. Specifically, we illustrate the approach for the FMF2 model. As mentioned in Section III, we use the Bayesian personalized ranking (BPR) objective function of Rendle et al. [20]. We use the stochastic gradient descent optimization algorithm [7] for training our models.

The output of our model needs to be a ranking function R_i for each item i in the target campaign T that ranks the users according to who is most likely going to purchase the item. From the training data, if we know that user u_1 has bought item i and that another user u_2 has not bought item i, then we must have $R_i(u_1) > R_i(u_2)$. The BPR objective function essentially enforces this criterion for all items and every pair of users within a given item. Before discussing the objective function, we present some of the notations used. The parameters that we need to learn are the user factors v_u^S, v_u^N, and α_j for all campaigns and the item factor matrix v_i^I. We denote the union of these parameters using Ψ. Also, denote $B(u_1)$ to be the set of items that are bought by user u_1. Also, let T denote the set of items in the target campaign and let N_j denote the set of items in the non-target campaign j. The log-likelihood function $F(v^S, v^N, v^I, \overline{\alpha})$ for this case (w/ BPR) is given by:

$$F(\psi) = \sum_{i \in I} \sum_{\substack{u_1: \\ i \in B(u_1)}} \sum_{\substack{u_2: \\ i \notin B(u_2)}} \log \sigma(x_{u_1 i} - x_{u_2 i}) - \lambda_\Psi ||\Psi||^2$$

We wish to treat the campaign items differently from the non-target campaign items, i.e., we penalize the errors on the target campaign much more than the non-target campaigns by using weights A and B for respective terms in the summation.The log-likelihood expression is now given by:

$$\begin{aligned} A \sum_{i \in T} \sum_{\substack{u_1: \\ i \in B(u_1)}} \sum_{\substack{u_2: \\ i \notin B(u_2)}} & \big[\log \sigma(\langle v_{u_1}^S - v_{u_2}^S, v_i^I \rangle) \\ & -\lambda ||v^S||^2 - \lambda ||v^I||^2 \big] + \\ B \sum_{i \notin T} \sum_{\substack{u_1: \\ i \in B(u_1)}} \sum_{\substack{u_2: \\ i \notin B(u_2)}} & \big[\log \sigma(\langle \alpha_j (v_{u_1}^S - v_{u_2}^S) + v_{u_1}^N - v_{u_2}^N, v_i^I \rangle) \\ & -\lambda_S ||v^S||^2 - \lambda_I ||v^I||^2 - \lambda_N ||v^N||^2 - \lambda_\alpha ||\alpha||^2 \big] \end{aligned}$$

where λ's denote the regularization constants. To use SGD, we sample a term from the above summation, which we denote using (i, u_1, u_2). Depending on whether the item is from the target campaign (i.e., $i \in T$) or from some non-target campaign j (i.e., $i \in N_j$), we obtain two sets of gradients which are shown in Figure 3.

Even though the objective function is highly non-convex, stochastic gradient descent has been shown to work well [22], [21], [20] with BPR on real datasets. To handle the local minima, we use multiple initializations of our factor matrices and select the best performing model via cross-validation.

Training model FMF3

The gradient rules for training FMF3 model is similar to the ones shown in Figure 3 and we do not present it owing to lack of space. However, for this model to be meaningful, we enforce that all values of $\alpha_j \in [0, 1]$ by using projected gradient [6] over the α_j. Essentially, whenever the gradient rules force the value of α_j to be less than 0 (more than 1), we force it to be exactly 0 (1). This enables us to intuitively interpret the α_j parameter as determining the fraction of the user's local and global interests.

$$
\begin{aligned}
c_{i,u_1,u_2} &= 1-\sigma(\langle v^S_{u_1}-v^S_{u_2}, v^I_i\rangle)\\
\frac{\partial F}{\partial v^I_i} &= A(c_{i,u_1,u_2}(v^S_{u_1}-v^S_{u_2})-\lambda v^I_i)\\
\frac{\partial F}{\partial v^S_{u_1}} &= A(c_{i,u_1,u_2}v^I_i-\lambda v^S_{u_1})\\
\frac{\partial F}{\partial v^S_{u_2}} &= A(-c_{i,u_1,u_2}v^I_i-\lambda v^S_{u_2})
\end{aligned}
$$

(i) item i is in the target campaign (i.e., $i \in T$).

$$
\begin{aligned}
c'_{i,u_1,u_2} &= 1-\sigma(\langle \alpha_j(v^S_{u_1}-v^S_{u_2})+v^N_{u_1}-v^N_{u_2}, v^I_i\rangle)\\
\frac{\partial F}{\partial v^I_i} &= B(c'_{i,u_1,u_2}(\alpha_j(v^S_{u_1}-v^S_{u_2})+v^N_{u_1}-v^N_{u_2})-\lambda v^I_i)\\
\frac{\partial F}{\partial v^S_{u_1}} &= B(\alpha_j c'_{i,u_1,u_2}v^I_i-\lambda v^S_{u_1})\\
\frac{\partial F}{\partial v^S_{u_2}} &= B(-\alpha_j c'_{i,u_1,u_2}v^I_i-\lambda v^S_{u_2})\\
\frac{\partial F}{\partial v^N_{u_1}} &= B(c'_{i,u_1,u_2}v^I_i-\lambda_N v^N_{u_1})\\
\frac{\partial F}{\partial v^N_{u_2}} &= B(-c'_{i,u_1,u_2}v^I_i-\lambda_N v^N_{u_2})\\
\frac{\partial F}{\partial \alpha_j} &= Bc'_{i,u_1,u_2}\langle v^S_{u_1}-v^S_{u_2}, v^I_i\rangle
\end{aligned}
$$

(i) item i is in the non-target campaign j (i.e., $i \in N_j$).

Fig. 3: Update rules for the FMF2 model.

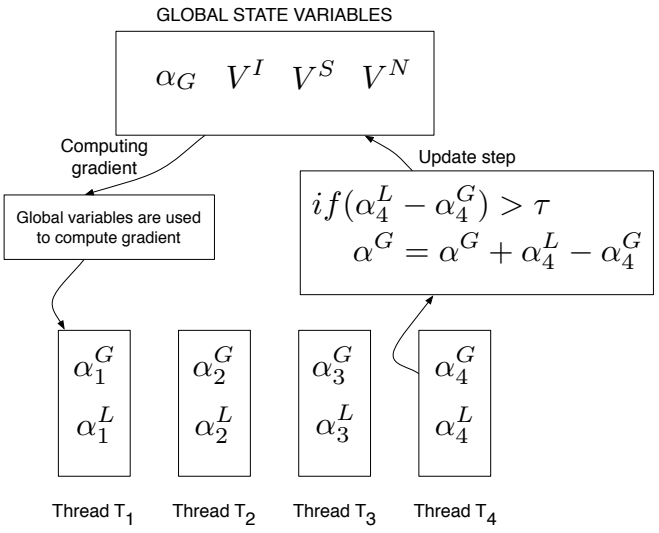


Fig. 4: Pictorial description of the multi-threaded implementation. Note that the global variables are used to compute the gradients. The global variable is updated whenever the difference exceeds the threshold.

V. Implementation

We developed a multi-core implementation of our focused collaborative models in C++. We used the BOOST [1] library package for storing the factor matrices.

A. Parallelizing Training & Evaluation

Our multi-threaded approach is developed using locks. The global state maintained by the SGD algorithm consists of the 3 factor matrices $\{v^S, v^N, v^I\}$ and the α vector. We introduce a lock for each row in our factor matrices. In the SGD algorithm, in each iteration of training, we execute 3 steps. In the *first step*, we sample a 3-tuple (i, u_1, u_2). In the *second step*, we read the appropriate user and item factors and compute the gradients with respect to them. Before reading, we obtain a read-lock over the factor and release it after reading. In the *third step*, we update the factor matrices based on the gradients. Before writing we obtain a write lock on the item factor and subsequently release the lock once we update the factor.

Each user and item factor matrix has a large number of rows (over 500,000). Even with a fairly large number of threads, the contention over such rows is fairly small. However, the α vector is relatively small (equal to the number of campaigns, as in FMF2 and FMF3 models). Therefore, there is much more contention on this vector. As we show in Section VI, using locks over such small vector can result in significant increase in the processing time. To alleviate this problem, we propose to use a novel caching technique which we illustrate below.

Caching:

We illustrate the caching technique for a scalar α value (this can be easily generalized to vectors). In this technique, each thread T_i maintains two values in its cache: α^G_i which is the value with which the current thread started off, and α^L_i which is the locally cached value. We maintain that the $|\alpha - \alpha_G| <= \tau$ where τ is a given tolerance threshold. In each iteration, the threads read the value of α^G from the global value by obtaining a read lock. Whenever a thread needs to update the value of α, it updates α^L_i using the update rules. Whenever $|\alpha^G_i - \alpha^L_i|$ exceeds τ, we reconcile the locally cached copy with the global value in the following manner:

$$
\begin{aligned}
\alpha^G &= \alpha^G + (\alpha^L_i - \alpha^G_i)\\
\alpha^L_i &= \alpha^G_i = \alpha^G
\end{aligned}
$$

We pictorially illustrate the caching in Figure 4. We use a similar technique for caching the α vector for FMF2. We also parallelize evaluation similarly. Each thread takes a subset of items in the target campaign and ranks the users within each item independently. Note that we only need to read the factor matrices in this case.

VI. Experimental Evaluation

In this section, we present the results of our experimental evaluation. We compare our proposed FMF techniques with the alternative approaches MF and GMF. We start with a description of the data set and the evaluation metric.

A. Experimental Setup

1) Dataset: To evaluate our proposed models, we use the log of previous advertising campaigns obtained from a major advertising network. The dataset contains information about the items corresponding to various advertising campaigns and

an *anonymized* list of users who actually responded to the campaign by making a purchase of the campaign item. In addition, we have a taxonomy over the various items in the campaigns. For our experiments we sample a fraction of campaigns from this dataset. In this sample, we have about 50,000 users and around a million items in the taxonomy. The taxonomy itself is 3 levels deep, with around 1500 nodes at lowest level, 270 at the middle level and 23 top level categories. As described in Section II, a campaign is a set of related items. We have a total of 23 campaigns in our dataset. For each of the campaigns, we do the following: we select it as a target campaign and put the remaining set of campaigns as the additional non-target campaigns. Each campaign is characterized by the number of items that is targeted by it, number of purchases that are present, how homogeneous the items are and how much it is related to other campaigns. We do not report the campaign identifiers due to anonymity reasons, we denote them using C_1, C_2 and so on.

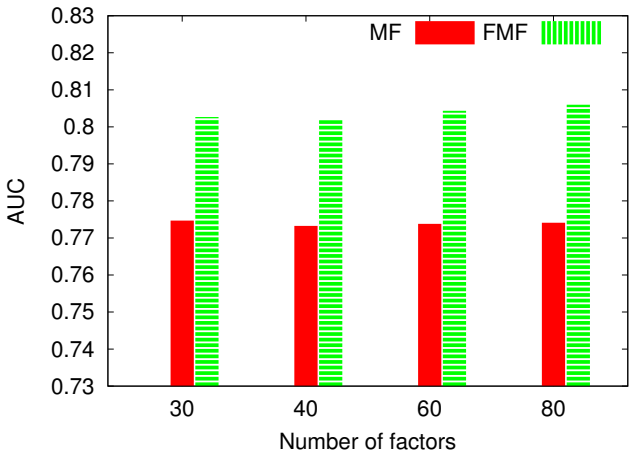


Fig. 5: Figure shows the improvement in performance over all campaigns using our proposed FMF model over the baseline matrix factorization approach.

To construct the user-item data matrix necessary for collaborative filtering, we select all the purchases made by the users in the above set of campaigns and sort them by their timestamps. For each item we select a random timestamp and select all the data prior to this timestamp into the training dataset. The rest of the data is used for evaluation. The last T transactions in the training dataset are used for cross-validation and first T transactions in the test dataset are used for prediction and reporting the error estimates. In all experiments, we use $T = 1$.

2) Comparison Systems: We compare our proposed models FMF1, FMF2 and FMF3 against the following methods.

1) MF: Here, we use the MF approach described in Section III as a baseline.
2) GMF: We also compare against the global matrix factorization (Section III).
3) GMF(t), MF(t): In addition, we use the above models along with the taxonomy extension (proposed in Section III-B).

3) Metrics: We use the area under the ROC curve (AUC) to compare our model with the above systems. AUC is a commonly used metric for testing the quality of rank orderings. It gives the probability with which a random positive example is ranked higher than a negative example in the ordering (i.e., a pure random ordering achieves an AUC of 0.5). Suppose the list of users to rank (for a given item) is U and our ground truth (i.e., the set of users that actually bought the item) is B. Also suppose $r(u)$ is the numerical rank of the user u according to our model (from $1 \ldots n$). Then, the formula to compute AUC is given by:

$$\frac{1}{|B||U \setminus B|} \sum_{u \in B, v \in U \setminus B} \mathbf{1}(r(u) < r(v))$$

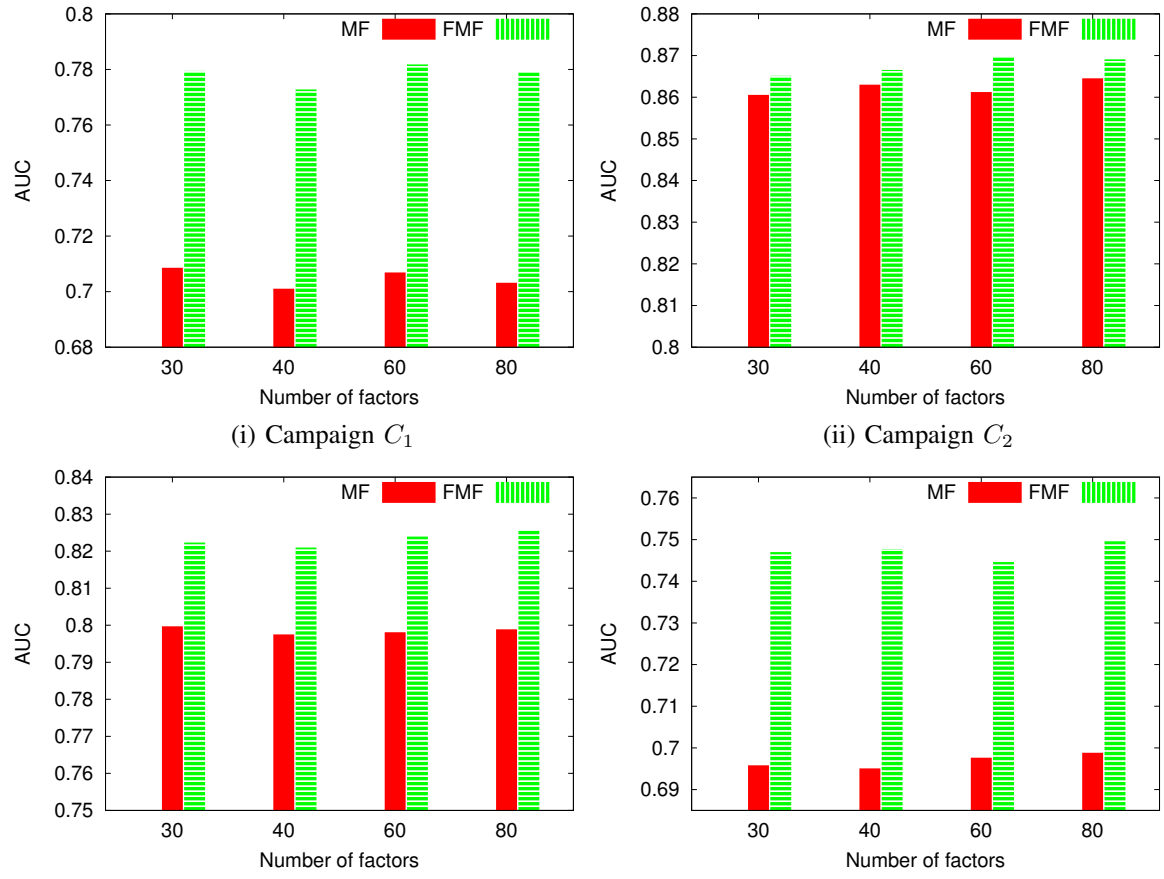
Here, $\mathbf{1}$ is the indicator function that is 1 if the condition is satisfied and 0 otherwise. An alternative metric could be to measure precision/recall at a certain rank in the list. However, different campaigns may have different requirements in terms of precision and recall. Hence, selecting a rank at which to evaluate precision such that it would be suitable for all campaigns is not possible. Instead, we use AUC since it combines the prediction performance over all ranks into a single number.

4) Cross-validation/Parameter sweep: For each of the experiments that we illustrate below, we executed a parameter sweep over our MapReduce cluster. The parameters we sweep over included λ_U, λ_I, λ_N and K, the number of factors. In addition, since our objective function is non-convex, for each setting of the parameter we evaluated 4 different initializations and picked the best initialization for each configuration, in terms of performance on the validation dataset. Finally, we chose the AUC over the test set for a given number of factors to report in our experiments.

B. Experimental Results

1) Improvement Over the Baselines: In the first experiment, we compare the GMF, MF and FMF2 techniques for different campaigns. For each campaign, we learn all the above models and evaluate the accuracy of prediction over the campaign items using the AUC metric. We report the average AUC across all the campaigns in our data set in Figure 5. As shown in the figure, the AUC for FMF2 is higher than the AUC values for the MF model. Next, we examine the performance of the FMF models over the individual campaigns. We drill down across four different campaigns C_1, C_2, C_3 and C_4 which are chosen such that they representative of all the campaigns in the dataset. We show the results for the above campaigns in Figure 6 (i-iv). We make the following observations: First, for campaigns C_1, C_3 and C_4, the improvement of FMF over the MF model is substantial (over 7% for campaign C_1), however the improvement for campaign C_2 is less than 2%, which is not statistically significant. We attribute this to the fact that the number of purchases in C_2 is high, and subsequently, the response matrix is not as sparse as those for other campaigns. Hence, the additional benefit of FMF is not as pronounced.

Second, we note that between the GMF and MF methods, there is no clear winner. In Figure 7, we plot the best AUC values across all factor sizes for each campaign. As shown in the figure, in campaign C_2, MF outperforms GMF, whereas



(i) Campaign C_1 (ii) Campaign C_2

(iii) Campaign C_3 (iv) Campaign C_4

Fig. 6: Improvement over baseline: In parts (i-iv) we show the performance of the MF and FMF models over campaigns C_1, C_2, C_3 and C_4 respectively. As shown here, FMF models consistently outperform the baseline models.

in the other campaigns, GMF outperforms MF. This occurs because the number of purchases in C_2 is high and that C_2 is a large campaign which helps MF. This figure demonstrates that blind sharing of information as performed by GMF does not always help, as in the case of campaigns C_2.

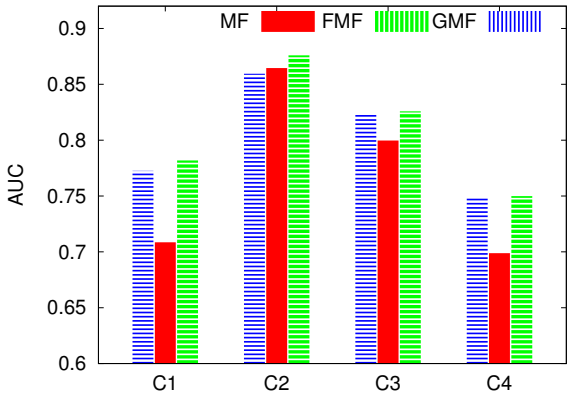


Fig. 7: We summarize the best performing model (across factor sizes) for each case. Note that between GMF and MF, there is no clear winner, i.e., in campaign C_2, MF outperforms GMF

Note that our FMF2 model performs fairly well for every campaign, since it can adapt in terms of how the information from non-target campaigns is borrowed, the extent of information that needs to be borrowed, etc. We further dissect the performance behavior of FMF models in detail in Section VI-B.2.

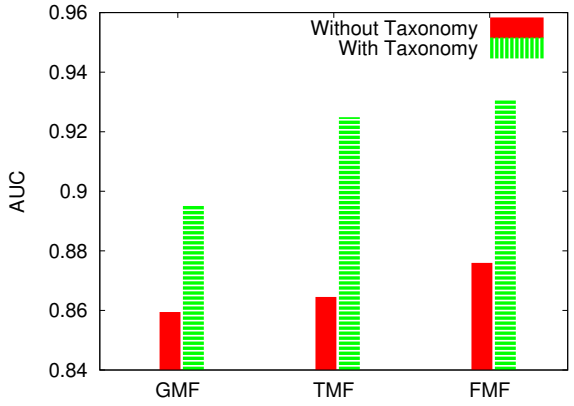


Fig. 8: We show the improvement obtained by all the models using the taxonomy, and the improvement of the FMF(t) models over the GMF(t) and MF(t) baselines.

Improvement Using Taxonomy

Next, we compare the FMF(t) model (i.e., the FMF model

augmented with information about the taxonomy) against the taxonomy-aware MF(t) and GMF(t) models. As before, we execute a parameter sweep and determine the best AUC values for each of the models, for different values of the number of factors. We show the results in Figure 8 for campaign C_2. As shown in the figure, all models benefit from using information about the product taxonomy, just as expected. However, the FMF(t) model still outperforms the GMF(t) and MF(t) models. Also, it is worth noting that MF draws the largest benefit from the taxonomy – this can be explained based on the fact that MF factorizes the target campaign matrix only and hence, is affected by sparsity the most. The taxonomy-based smoothing of item factors alleviates this issue, thus leading to large improvement in MF.

2) Understanding Focused Matrix Factorization: In this section we conduct a thorough analysis to investigate the performance behavior of the FMF models. We aim to identify key characteristics that affect the FMF models.

Comparison of the different FMF Models

We start by comparing the FMF1, FMF2, FMF3 models with each other. For each of the four campaigns we train the FMF1, FMF2 and FMF3 models. We show the AUCs metrics, as before, over the test dataset in Figure 9 (i-iv). As shown in the figure, the models FMF1 and FMF2 perform much better than FMF3. We reason this is because FMF3 is much more constrained than the other two models (as α_j values are constrained to be between 0 and 1). However, with FMF2, the α_j values are unconstrained leading to greater freedom of learning. Indeed, in our analysis we observed some α_j values were positive and others negative, reinforcing our belief that some campaigns may be negatively correlated. For brevity, we focus on FMF2 in the remaining experiments in this section.

Effect of Campaign Size

In this experiment, we understand the performance of FMF2 model as a function of the target campaign size, i.e., the number of items in the target campaign. We select 8 different campaigns with varying sizes (ranging from a few hundreds to many thousands of items). We show the AUCs achieved by the FMF2 model in Figure 9(i). The x-axis denote the campaign index in increasing order of campaign size. Note that the performance of FMF2 is robust and largely unaffected by the campaigns size.

Effect of Intra-Campaign relationship (Campaign Homogeneity)

In this experiment, we explore the performance of FMF2 models as a function of the homogeneity (user purchase pattern similar across all products) of the target campaign. To test this, we create a set of hypothetical target campaign $C'(p)$ from an existing campaign C for different values of p. The campaign $C'(p)$ is constructed in the following way: with probability p, we select an item from C and with probability $1-p$, we select a random item from the complete collection of items. The size of $C'(p)$ is kept the same as C. In the experiment, we used p = { 0, 0.34, 0.60, 1}. We train the FMF2 model over these campaigns for four different values of K (number of factors), and plot the AUCs in Figure 9(ii). As shown in the figure, the AUC scores increase as we increase the homogeneity of the campaign. This is expected since information transfer to a campaign is much easier (through α_j) if the campaign is much more coherent and homogeneous.

Effect of Inter-Campaign Relationship (Information Transfer)

Next, we illustrate the effect of inter-campaign relationship for information transfer in the FMF2 model. Intuitively, we expect that as we increase the positive correlation between the target and the non-target campaigns, we should see more information transfer and improved AUC values. We ran a controlled experiment to verify this as follows: We picked a fairly homogeneous campaign X (e.g., electronics) and split it randomly into two parts X_1 and X_2. Then we picked another campaign Y and constructed two configurations using X_1, X_2 and Y as follows. In both configurations, called *config 1* and *config 2*, X_1 is made the target campaign, while X_2 is the non-target campaign in *config 1* and Y is the non-target campaign in *config 2*. We ignore the rest of the campaigns from the data for this experiment. After running the FMF2 model with these configurations, we plot their AUCs in Figure 9 (iii). As shown in the figure, config 1 has much higher AUC than config 2 since it has X_2 as the non-target campaign which is highly similar to X_1. In other words, our FMF2 model successfully manages to transfer the information from the X_2 campaign to achieve better performance.

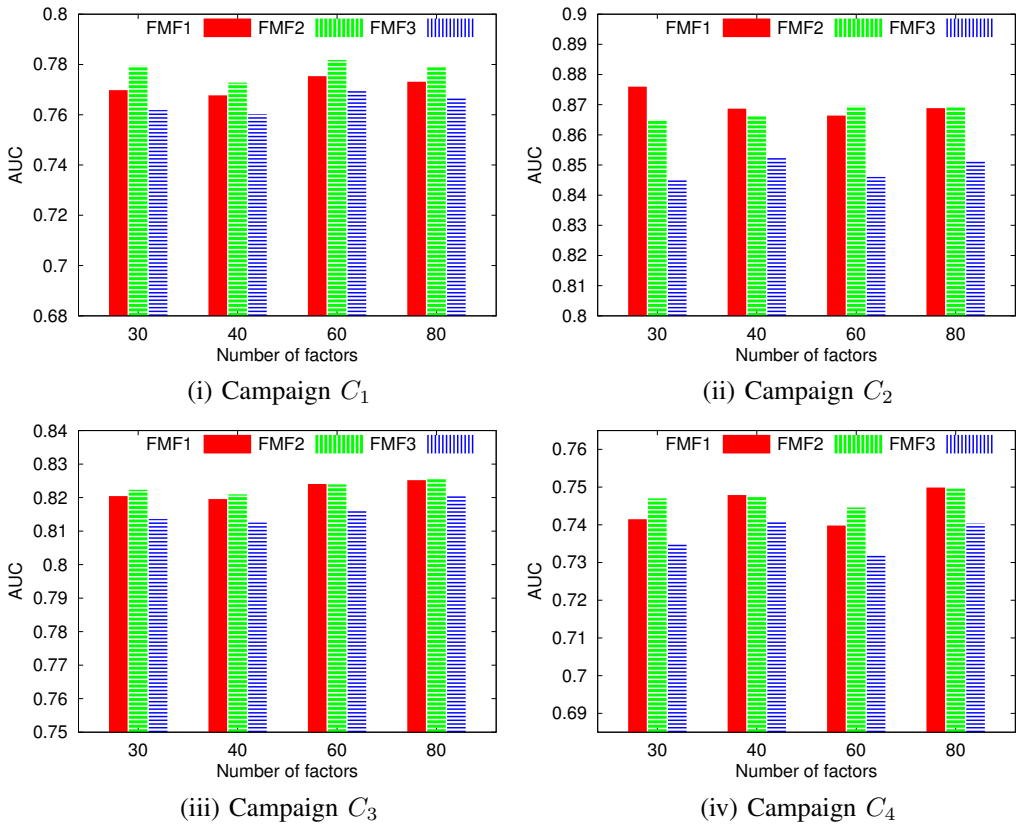
3) Efficiency: As described in Section V, we develop a multicore implementation of the training and evaluation algorithms. To deal with lock contention over frequently accessed matrix entries, we use our caching techniques. In this experiment, we demonstrate the trade-offs that is obtained by using the caching technique. Recall that we update the global variable only if we exceed a specified threshold value. When this threshold is set to 0, there is complete synchronization. As the threshold is increased, the synchronization with the global copy is performed less often, resulting in faster runtime but less accuracy. In other words, caching allows us to trade-off accuracy for efficiency.

For the FMF2 model, we measure the time taken per *epoch* of training with caching enabled and without caching. The results are shown in Figure 9(iv). We make two observations here: First, we obtain substantial improvement using caching. For instance, the run-time is cut to almost half by enabling caching. Second, even with a reasonably large value of threshold = 0.001, we did not observe any significant loss in the accuracy of the model.

Related Work

Recommender Systems

Classical approaches in collaborative filtering are based on item-item/user-user similarity, these are nearest-neighbor methods where the response for a user-item pair is predicted based on a local neighborhood mean [26], [24]. However, modern methods based on matrix factorization have shown to be very successful and outperform nearest neighbor methods [23]. In this paper we showed how matrix factorization can be used for audience selection in display advertising.



(i) Campaign C_1 (ii) Campaign C_2

(iii) Campaign C_3 (iv) Campaign C_4

Fig. 9: Study of FMF models: In part(i-iv), we study the relationship between the various FMF models that we propose. Essentially, we see that FMF1 and FMF2 models outperform FMF3.

Transfer Learning
The idea of using focused collaborative filtering draws inspiration from transfer learning / multi-task learning. Zhang et al. [27], however our work differs from these work since we only care about increasing the performance of the focused task rather than learning the task structure between all tasks as in [27], which reduces the number of correlation parameters to be learned from a quadratic number of parameters between all tasks to just a linear number of parameters between the focused task and all other tasks. Another related area is collective matrix factorization [25] which shares structure between multiple matrix factorization tasks, however, our focus here is on efficient way of training the resulting model.

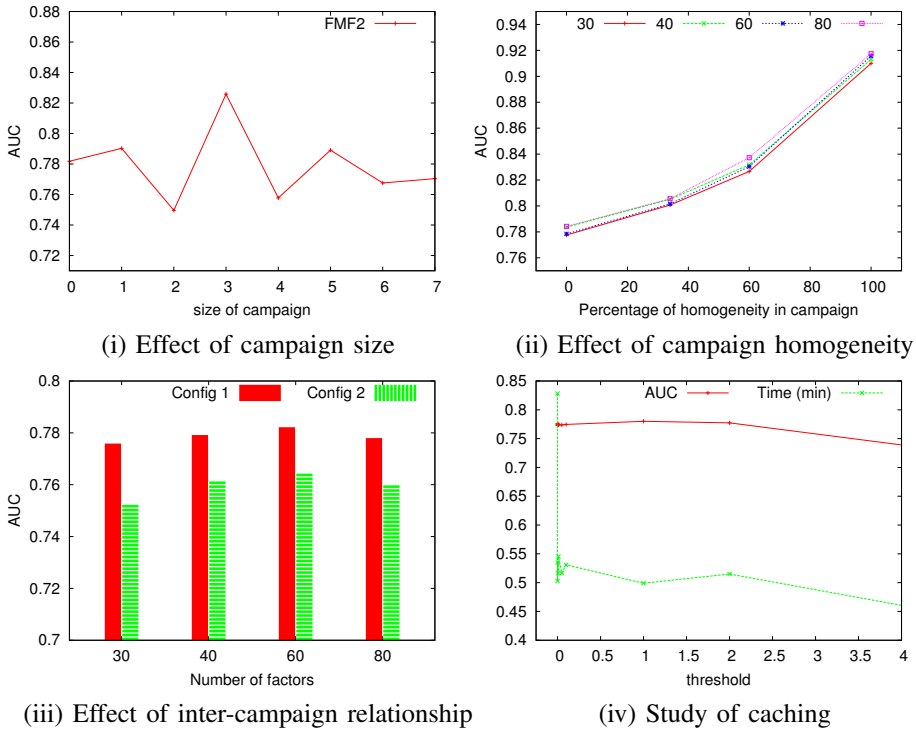
Audience Selection in Display Advertising Display advertising is increasingly becoming more and more performance oriented where the goal is to identify and target customers most suited to the advertising campaigns. Existing work on this topic focuses on building models to characterize user interests based on their past activities, e.g., search queries, pages browsed [18], [5]. The advertising network can track and construct user history to build these models for targeting purposes. In contrast, our work does not require user online activities to be given, instead we mine the past purchase records to bring together similar users and advertisers. To the best of our knowledge, this is the first work to apply matrix factorization approach to display targeting. Also, it is possible to directly extend this work for the case where user history is given. For instance, similar to [4], we can derive user factors by regressing on the user features, and thus combine the purchase history as well as past online activities of the user.

Conclusions

Display advertising has grown into a multi-billion dollar industry in recent years. To this end, there has been extensive work on behavioral targeting and user personalization to solve the key problem of audience selection for ad campaigns. Much of the previous work use users' search logs, browsing history and other features for targeting ads to user. In this paper, we propose to use information from previous campaigns, which contains information about a user's actual purchases and is hence feature rich. We propose a novel *focused matrix factorization* model, in which we develop techniques to borrow information from related campaigns while ignoring information from unrelated campaigns. As shown in our extensive empirical study, the FMF model consistently outperforms the traditional matrix factorization techniques over all kinds of campaigns. In addition, in our experimental study, we characterized the conditions under which we can obtain significant improvements from our approach.

References

[1] Boost c++ libraries. http://www.boost.org/.
[2] Pricegrabber. http://www.pricegrabber.com/.



(i) Effect of campaign size

(ii) Effect of campaign homogeneity

(iii) Effect of inter-campaign relationship

(iv) Study of caching

Fig. 10: In part(i), we study the effect of campaign size over the models. In part(ii), we show that as the campaigns are more homogeneous, we improve the recommendation accuracy. In part (iii), we show the effects of inter-campaign relationships over the recommendation. In part (iv), we show the accuracy-time trade-offs provided by our caching technique.

[3] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[4] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[5] A. Bagherjeiran, A. Hatch, A. Ratnaparkhi, and R. Parekh. Large-scale customized models for advertisers. *Data Mining Workshops, International Conference on*, 0:1029–1036, 2010.

[6] D. P. Bertsekas and D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[7] L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, pages 146–168, 2003.

[8] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

[9] L. Bottou and Y. LeCun. Large scale online learning. In *NIPS*, 2003.

[10] J. Eisenstein, A. Ahmed, and E. P. Xing. Sparse additive generative models of text. In *ICML*, 2011.

[11] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC Texts in Statistical Science, 2003.

[12] M. I. Jordan. *Learning in Graphical Models (ed)*. MIT Press, 1998.

[13] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[15] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[16] Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. 2011.

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

[18] Y. Liu, S. Pandey, D. Agarwal, and V. Josifovski. Finding the right consumer: optimizing for conversion in display advertising campaigns. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 473–482. ACM, 2012.

[19] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-thieme. L.s.: Bpr: Bayesian personalized ranking from implicit feedback. In *In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[21] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 2010.

[22] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, 2010.

[23] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[24] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[25] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD*, KDD '08, pages 650–658. ACM, 2008.

[26] J. Wang, A. P. D. Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *In SIGIR 06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[27] Y. Zhang, B. Cao, and D.-Y. Yeung. Multi-domain collaborative filtering. In *UAI*, 2010.