Recommendations Using Coupled Matrix Factorization

No Author Given
No Institute Given

Abstract. Consider the task of recommending movies to users, such as the Netflix challenge. Given additional information for the movies, e.g. features derived from Internet Movie Database, are we able to improve the recommendation quality? In this work, we pose the aforementioned problem (which encompasses a variety of different scenarios, all under the family of recommender systems/collaborative filtering) as a regularized Coupled Matrix Factorization instance. We demonstrate the effectiveness of our approach over traditional collaborative filtering approaches that employ matrix factorization. In particular, we show that for two different recommendation tasks viz. movie recommendation and question recommendation for “ask an expert” web site, our model outperforms state of the art matrix factorization models, in terms of root mean squared error.

1 Introduction

Given a set of users, a set of movies, as well as a rating (typically a number from 1-5) that each user gave to every movie they watched, the objective is to predict the rating that a particular user would give to a movie, if they were to watch it; this prediction will effectively be used as a movie recommendation, if the predicted rating exceeds a certain threshold. This problem belongs in the general family of recommender systems, a problem family rich in applications in the real world.

One of the most prominent approaches of tackling this problem is Collaborative Filtering, a class of algorithms which attempt to use information from all users, and thereby come up with a prediction. In particular, doing so leverages similarities amongst users and movies, and has been shown to be very effective. A very popular solution to Collaborative Filtering is through Matrix Factorization; drawing from the movie recommendation example, we form a matrix whose rows correspond to movies and whose columns correspond to users; each element of the matrix contains either the rating of a user for a particular movie, or zero if there is no rating provided. The end goal is to fill in the missing entries of that matrix in such a way that a particular cost function is minimized.

Suppose now that we gather additional data for each movie; for instance, we might have genre information, or rating information from some online repository such as Internet Movie Database (IMDb). Can this additional information prove helpful in the aforementioned task? And if so, how can we put this additional
information into play, framing the problem as an instance of collaborative filtering? Recently, there has been an emerging interest in this type of information fusion, e.g. in [26], the authors combine user friendship information (user-user matrix) with user interests (user-service matrix), in order to enhance the recommendation task at both ends. In this work, we propose to express this problem as an instance of Coupled Matrix Factorization (CMF).

Additionally, this problem formulation proves to be highly generic, so that it accommodates a variety of other applications; in particular, we also apply the model to perform a collaborative filtering for users of stackoverflow.com [3]. This is a forum where registered users can post questions and assign relevant tags to it. Multiple users provide answers to a question, and one of them may be chosen as the best answer. Irrespective of whether an answer is chosen as best or not, all answers receive votes/points from the community. These points along with tags marked in the question, are added to the user’s profile who provided the answer. Hence, tags and points aggregated over all answers provided by the user, makes a set of technology tags of interest to the user. It will be preferred that questions with related tags which could not be attached to a user profile should also be shown to him/her. We use a similar approach, as for Netflix, to provide question recommendations using tags to a user. The rows of the matrix correspond to users and columns to tag points; each element of the matrix contains either the points for a tag given to a user, or zero if none were awarded. The final goal is, as in the case of Netflix, to fill missing entries. Although, here additional information is collected from stackoverflow.com itself rather than from a different source as IMDb; additional data for each user is collected from a different user who has similar and more tags attached to his/her profile.

Our main contributions are:

– We formulate the problem of recommendation with additional information as a Coupled Matrix Factorization with two different types of regularization, namely on the $\ell_1$ and $\ell_2$ norms, respectively.
– Our approach is agnostic of the type of the additional information, thus generic.
– We experiment on two diverse, real world applications and demonstrate the effectiveness of our approach, in terms of rating prediction.

In Section 2, we describe cost function along with the method formulation and algorithm. Section 3 describes the mapping of raw data into our method formulation (described in Section 2). Experiments and evaluation with standard techniques are shown in Section 4. Section 5 introduces related work on how fusion of additional information helps in improving the objective. Moreover, we also survey the current state of the art, in terms of leveraging additional information for recommendations. In Section 6, we conclude, and provide future directions for improvement.
2 Proposed Method

Notation

Scalars are denoted by lowercase, italic letters, e.g. \( x \). Matrices are denoted by uppercase, boldface letters, e.g. \( X \). We use Matlab notation for matrix indexing, i.e. the \((i, j)\)-th element is denoted by \( X(i, j) \).

2.1 Method Description

We have focused on a broad spectrum of aspects related to the project, and more specifically; on the algorithm development; the acquisition and the preprocessing of data; as well as coming up with a comprehensive means of comparison with existing baselines; and applying it to different application domains.

We propose to jointly factorize \( X \) and \( Y \) into a set of low rank components, such that an appropriate loss function is minimized. The most commonly used loss function is the Frobenius norm, which reflects the sum of squared errors. We additionally regularize the loss function in two different ways. Our first model is a Coupled Matrix Factorization with \( \ell_2 \) regularization on the factors:

\[
\min_{A,B,C} \alpha \|X - AB^T\|_F^2 + \beta \|Y - AC^T\|_F^2 + \mu (\|A\|_F^2 + \|B\|_F^2 + \|C\|_F^2) \quad (1)
\]

where \( \alpha + \beta = 1 \). Note that the left factor matrix for both \( X \) and \( Y \) is the same, and this reflects the joint nature of our factorization. Main reason for the \( \ell_2 \) regularization is to avoid uncontrollably large values in the factors, and thus, to avoid overfitting.

The second means of regularization is adding \( \ell_1 \) norm penalties to the three factor matrices. This penalization is done for two reasons: 1) as before, in order to overfitting, and 2) by using the \( \ell_1 \) norm, we are producing a sparse latent embedding of the data into a lower dimension, which effectively suppresses noise, and retains the signal which is useful for collaborative filtering. The objective function is the following:

\[
\min_{A,B,C} \alpha \|X - AB^T\|_F^2 + \beta \|Y - AC^T\|_F^2 + \lambda \left( \sum_i \sum_f |A(i, f)| + \sum_j \sum_f |B(j, f)| + \sum_l \sum_f |C(l, f)| \right) \quad (2)
\]

A similar model has been used in [2], on an entirely different setting, that of exploratory biomarker identification in metabolomics.

As stated initially, our work revolves around Equations 1 and 2. By using the Frobenius norm as a loss function, we are able to solve the problem using Alternating Least Squares, a block coordinate descent optimization technique; in a nutshell, block coordinate descent fixes all data but a specific block, and
optimizes over that specific block. This is done in an alternating fashion, until no change in the cost function is observed.

In our particular case, we may obtain a locally optimal solution for Eq. 1 and 2 by fixing two of the three matrices at each step, optimizing over the third matrix and doing so in a "circular" way. The optimization problem is linear on any of the three matrix variables, thus, each step of the block coordinate descent is simply either a Ridge Regression [8] (for the $\ell_2$ penalty), or a Lasso regression problem [22] (i.e. Least Squares regression with $\ell_1$ regularization on the variable that we regress on).

Our algorithm is guaranteed to converge to a locally optimal solution, since each step is solved optimally for the specific block that we optimize for, and thus, it is guaranteed to decrease the cost monotonically. Finally, we incorporate non-negativity constraint on all three factor matrices $A, B, C$ by projecting the solution of each step of the algorithm to the $[0, \infty)$ interval.

In Figure 1, we illustrate our proposed method. Using Netflix as an example, say that $X$ is the movie by user matrix. After we obtain the low rank factors $A, B, C$, we may reconstruct $X \approx AB^T$ as a matrix of lower rank. As we review in the Related Work section, low rank factorization models are behind many state of the art collaborative filtering algorithms.

By decomposing $X$ into a small number of rank one factors, what we essentially accomplish, is that we express the data as a sum of parts; consequently, we express each user as a weighted sum of latent movie concepts, defined by the columns of $A$. We, thus, may compute this weighted sum of latent movie concepts.
concepts, and obtain a new vector representation for each user. Essentially, as shown in Fig. 1, we need to reconstruct $X$, by multiplying out $AB^T$.

After we reconstruct $X$, we may simply look at the coefficients of interest and obtain an estimate of the missing values. Furthermore, we may also post-process these predicted values in a way that it makes them more suitable for interpretation, e.g. quantizing them in $\{1, 5\}$. In [14], the interested reader may find a concise and intuitive tutorial behind the logic of using latent factor models in collaborative filtering.

There are two, particularly interesting observations for our proposed approach, which differentiate it from the existing work that uses factorization models: First, by coupling matrices $X, Y$, as in Fig. 1, we incorporate the additional information, contained in $Y$, to the latent factor $A$. Thus, our low rank representation is richer in information, compared to a factorization performed solely on $X$. Second, under our proposed formulation, we can perform two-way ”recommendation”/completion, since we obtain a joint low rank factorization for both data matrices. Therefore, in the very same way, we may approximate matrix $Y$ as $AC^T$ and obtain our estimates. In the Experimental section, we demonstrate this two-way collaborative filtering property of our method.

3 Data Description

As we mentioned in the Introduction, we apply our algorithm to two, diverse, recommendation problems. The reason we do so, is because our algorithm is agnostic of the type of additional information given. In case of movie recommendations to Netflix users, the additional information from IMDb contains extra features like genres, and ratings from a different set of audience. The set of movies in both Netflix and IMDb is invariant. On the other hand, for stackoverflow.com there is no other source like IMDb. Here additional information contains a different set of users having similar and/or more interests than the set of users for which recommendations are to be provided.

We shall see in Experimental Evaluation section how the selection of the type of additional information reflects on recommendations.

**Netflix & IMDb Data** The Netflix dataset contains ratings of anonymized users, to movies. The number of movies in this dataset is about 18K movies and the number of users who provided ratings (on a scale from 1-5) is about 2.5 million. The Netflix data corresponds to matrix $X$ in our formulation, whose rows correspond to movies, and whose columns correspond to users.

In terms of matrix $Y$, we use information crawled from IMDb, ultimately on a shared subset of movies that exist in the Netflix data. About 200k movies have been crawled from the IMDb website. For each movie, we collect the rating (on a scale from 1-10), as well as genre information, which is a binary feature.
Stack Overflow Data  Here we look at the publically available data dump for stackoverflow.com[1]. Collection of data is divided into two steps viz. Data Aggregation and Data Cleaning.

Data Aggregation: To make recommendations for a user, $U_i$, we first look at all answers provided by the user and aggregate all points/up-votes received for each answer. Every question posted on stackoverflow.com has a set of tags. A tag is a keyword which acts like metadata to describe the question and allows for easy browsing or searching. Hence, when a random registered user of stackoverflow.com provides an answer to a question and the answer gets upvoted, then each tag, along with number of votes for the answer, is attached to the user profile to project his/her set of interests. Algorithm 1 below formally describes the aggregation of interest sets from the data dump.

Algorithm 1: Create interest set $IntS_i$ for a user $U_i$

Require: Set of answers $A_i$ posted by the user $U_i$ and the corresponding set of questions $Q$ for each answer in $A_i$

1: Set the interest set, $IntS_i$, for $U_i$ as a null vector
2: for each $a$ in $A_i$ do
3: Find set of tags $T$ for the question to which user($U_i$) provided an answer($a$)
4: for each $t$ in $T$ do
5: $IntS_i(t) = IntS_i(t) + Upvotes(a)$
6: end for
7: end for
8: return vector $IntS_i$

Data Cleaning: Since a lot of tags are found in not more than a couple of hundred questions, we restrict the vector size of interest set for each user to be at most 200 based on popularity, since we observed that tags beyond 200th mark, if ranked based on popularity, were mentioned in less than 0.2% of questions.

To form the $X$ matrix, tag points correspond to the columns and users correspond to rows. We randomly make $X$ from our pool of users($P$). For $Y$ we keep the number of columns same as $X$. Each row $Y_i$ is a row in $P$ such that the following function is maximized:

$$Y_i = \arg\max_{P \setminus X_i} P_j \cdot X_i$$

4 Experimental Evaluation

For the purposes of evaluating our approach we are applying the following procedure: choose, uniformly at random, a set of data entries (movie ratings in the case of the Netflix matrix, or genre and rating information in the case of the IMDb matrix) and arbitrarily set it to zero (bearing in mind that zero in the case of ratings doesn’t have any physical interpretation, other than the fact that
a particular entry is missing). After having hidden a set of elements, we solve Eq. 1 or 2 we predict them as we described earlier. Additionally, we are testing the algorithm’s performance on the reverse setting, where we the roles of matrix $X$ and $Y$ are reversed (we are using $X$ as additional data for $Y$).

### 4.1 Baselines

Since our method is based on matrix factorization, our baselines are two basic, albeit, strong matrix factorization methods, which may be applied to only one of the two matrices $X, Y$ at each time. Since both baselines operate under the factorization regime, the prediction of missing values is done in the same fashion, after we obtain the low rank factors. The two baseline methods are:

**Singular Value Decomposition** The SVD of $X$ in a lower rank can be shown (by the Eckart-Young theorem) that is the best low rank approximation of $X$ in the least squares sense. The model is written as:

$$X = U\Sigma V^T$$

However, optimality in the least squares sense does not give any guarantees for the completion quality, when values from $X$ are missing. As we review in the Related Work section, many state of the art techniques are based on the SVD.

**Non-negative Matrix Factorization:** This model was popularized by [15] and is a small but very insightful modification of the bilinear, SVD, factorization. In particular, the additional constraint subjects the two factors to non-negativity constraints, which make a lot of sense, especially for interpretation purposes (but not restricted to them), when the original data is non-negative (as in our case, and in many real world scenarios).

$$\min_{A, B \geq 0} \|X - AB^T\|_F^2$$

### 4.2 Datasets

The three datasets we used were the following:

**Netflix/IMDb:** The $X$ matrix is the Netflix movie-user matrix. Matrix $Y$ is the IMDb matrix that contains ratings and genres. Because this dataset is relatively large for Matlab to handle, we take samples of 100 users at a time; we, then, execute our method (and the baselines) 10 times, in order to estimate their performance.

**Stack Overflow:** Matrices $X$ and $Y$ are the ones shown previously for the stack overflow data.

**Synthetic:** We create synthetic data as follows: First we draw random sparse matrices $A$ of size $I \times F$, $B$ of size $J \times F$ and $C$ of size $I \times L$. We then take $X = AB^T$ and $Y = AC^T$. 
4.3 Root Mean Squared Error

We measure our method’s performance, compared to the baselines, by recording the Root mean Squared Error (RMSE) on the missing portion of the matrix, i.e. we only count the error on the ratings that we wish to predict. In Figure 2, we show the RMSE as a function of the rank of the factorization (i.e. the number of latent factors that we extract). In all three cases, we see that our method outperforms the baselines, indicating that additional information is beneficial for collaborative filtering. The figures we show are for $p = 10\%$ of the ratings missing; for higher percentages of missing ratings, we observed a similar trend of our approach outperforming the baselines; however, as the number of missing values increases, the prediction quality decreases. We determined the parameters via trial-and-error: in particular, for Netflix/IMDb $\alpha = 0.88, \lambda = 0.005$ (CMF-$\ell_1$) and $\alpha = 0.8, \mu = 0.05$ (CMF-$\ell_2$); for StackOverflow $\alpha = 0.4, \lambda = \mu = 1$, and for Synthetic $\alpha = 0.8, \lambda = 5, \text{and } \mu = 1$.

![Figure 2: Completion RMSE for 10% missing values. In all three cases, both our approaches (CMF-$\ell_1$ & CMF-$\ell_2$) outperform the two baselines, in terms of RMSE.](image)

Taking our analysis a step further, we assess whether reversing the roles of $X$ and $Y$ yields same, better, or worse completion quality. We, indeed, observe that the roles of $X, Y$, for the IMDb/Netflix dataset, are not reciprocal. That is, trying to complete values of $Y$ using $X$ proves to be as effective as just completing the values of $Y$ via traditional techniques.

**Parameter Sensitivity Analysis** In our model, we introduce parameters $\alpha, \beta, \lambda, \mu$ which may be useful in smoothing out the differences of the two pieces of data we combine together ($\alpha, \beta$), control the regularization/overfitting ($\mu$) and control the sparsity levels in the factor matrices ($\lambda$). Determining a set of appropriate values for $\alpha, \beta$ proves to be a very challenging research subject where there is some amount of work on, e.g. [24].

For the previous experiments, we used hand tuned parameters, however, we conducted an experiment in order to evaluate the behaviour of our models with respect to the $\alpha = 1 - \beta$ parameters, in terms of RMSE. Figure 3 shows the results for the Netflix/IMDb and StackOverflow datasets.
Fig. 3: Parameter sensitivity analysis for our two datasets. We vary $\lambda$ and $\alpha$ and measure the RMSE, for 10% of the values missing. We observe that in both cases, very small values of $\lambda, \mu$ perform better.

5 Related Work

Collaborative Filtering The public release of the netflix data [4] resulted in a number of approaches developed in a short time for improving recommendation systems. In general most of the current approaches to collaborative filtering are either neighborhood based methods or latent factor based approaches. We mostly focus in factorization methods, however, we provide a brief overview of neighborhood methods too. In neighborhood based approaches, the goal revolves around estimating the similarity between two pairs of entities [18]. Here, similarities could be computed between user-user pairs or item-item pairs. In a user based recommendation scenario, we first try to identify the most similar users for a given target user [17]. The similarity measure between any two users could be computed in a number of ways such as vector cosine similarity, Pearson coefficient etc [21][19][9][20].

In factorization based approaches, the use of SVD has been proposed in the literature [5], [7]. One of the major drawbacks however in using SVD type approaches is that we have to deal with the sparsity of the user-item matrix because
several of the user-item ratings are unknown. Also, modelling only based on the observed ratings could result in serious overfitting. Approaches like imputation have been used to make the user-item matrix more dense [10], however, this procedure might insert noise in the data. Several of the recent works have focused on modelling the observed user-item matrix directly through a regularized model, such as [11] [25]. By employing regularization, these methods attenuate the effects of overfitting.

An overview of Matrix Factorization techniques for Collaborative Filtering can be found in [14]. Most of the previous works are however based on the explicit feedback that was already provided by the users. But one of the major advantages of our approach is the fusion of side information from additional sources. By jointly factorizing all the matrices, we hope to improve the accuracy of the predicted ratings obtained by using netflix ratings alone. Also one of the major advantages of our approach is that, our joint factorization model is completely generic and could be applied to any domain where additional side information can be garnered (Stackoverlow, Friendship recommendation).

The Netflix Prize winner applied matrix factorization with temporal dynamics[13] which allows to capture time dependent behaviours of users. Moreover, in the final year of the competition, they also add predictors [12] which capture effects with respect to a user or movie. For example, a user Bob may have a tendency to rate a movie with a certain percentage less than the average rating for all movies. Similar effects which are independent of user interactions were incorporated in the final model for prediction. Our matrix factorization approach, does not consider such extra parameters and thereby reducing the overall computational cost.

Incorporating Additional Information

There exists a fairly recent line of work which, like in the present paper, incorporates some form of additional information, in order to boost collaborative filtering. In [14], the authors mention the use of implicit feedback as additional information for collaborative filtering. The authors of [23] study collaborative filtering for marketing, and also conclude that additional information improves predicted ratings. In [27], the authors attempt to model the context of a users choice of a movie (e.g. preference of a certain actor vs. preference of action movies in general), into the collaborative filtering using latent factorizations framework. In [26], the authors combine friendship information with user interests to improve recommendations. The authors of [16] introduce three successful ways of combining social network information to MF collaborative filtering, outperforming SVD. Finally, in [6], the authors provide a tensor based method of incorporating social data into cross domain collaborating filtering.

The advantage of our approach is that it is agnostic to the additional information, thus flexible; namely matrix $X$ whose values we need to complete, along with another matrix which matches one of $X$’s dimensions, suffice for our method to operate, and as we demonstrated, to outperform the baselines in diverse, real world applications.
6 Conclusions

In this paper:

- We formulate the problem of recommendation with additional information as a regularized Coupled Matrix Factorization; one of our proposed methods includes sparse modelling of the entities participating in the collaborative filtering process.
- The way we formulate the problem is generic enough, to accommodate diverse applications;
- We experiment on two such real world applications and show that our approach outperforms the two baselines.

As future work, we may consider incorporating temporal information (handling tensors instead of matrices), as well as experiment with different loss functions, which might be more appropriate for certain applications, where ratings are highly skewed.

References