Scalable learning of probabilistic latent models for collaborative filtering

Langseth, Helge; Nielsen, Thomas Dyhre

Published in:
Decision Support Systems

DOI (link to publication from Publisher):
10.1016/j.dss.2015.03.006

Publication date:
2015

Document Version
Early version, also known as pre-print

Scalable learning of probabilistic latent models for collaborative filtering

Helge Langseth[a], Thomas D. Nielsen[b]

[a] *Department of Computer and Information Science, The Norwegian University of Science and Technology, Trondheim (Norway)*
[b] *Department of Computer Science, Aalborg University, Aalborg (Denmark)*

Abstract

Collaborative filtering has emerged as a popular way of making user recommendations, but with the increasing sizes of the underlying databases scalability is becoming a crucial issue. In this paper we focus on a recently proposed probabilistic collaborative filtering model that explicitly represents all users and items simultaneously in the model. This model class has several desirable properties, including high recommendation accuracy and principled support for group recommendations. Unfortunately, it also suffers from poor scalability. We address this issue by proposing a scalable variational Bayes learning and inference algorithm for these types of models. Empirical results show that the proposed algorithm achieves significantly better accuracy results than other straw-men models evaluated on a collection of well-known data sets. We also demonstrate that the algorithm has a highly favorable behavior in relation to cold-start situations.

Keywords: collaborative filtering, scalable learning, probabilistic model, latent variables, variational Bayes

1. Introduction

Recommender systems have become a well-established technology to help users cope with vast amounts of information. This is achieved by only presenting to the users the information which is deemed most relevant. Over the last years the diversity of the domains in which recommender systems have been successfully applied has increased significantly and include movies, books, news, and products in general.

Recommender systems are typically grouped into two categories: *content-based* systems make item recommendations by combining content descriptions of the items in questions with a preference model of the user (e.g. inferred using previously rated items). On the other hand, *collaborative filtering* systems provide recommendations based on the ratings of other users with similar preferences. The two types of systems exhibit different characteristics; collaborative filtering systems typically enjoy a greater flexibility in terms of the types of

items that can be recommended whereas content-based systems are usually less susceptible to cold-start problems.

Collaborative filtering systems are often further sub-divided into *model-based* and *memory-based* [4] methods, although combinations of the two have also been proposed [29]. Memory-based systems rely on a distance measure to estimate user similarity, whereas model-based approaches learn a model of the user's preferences, which is subsequently used for making predictions. The earliest model-based approaches used a multinomial mixture model [10] for either grouping the users into user groups or items into item-categories. More recently, uniform models have been proposed that treat users and items equivalently and represent them jointly in the same model. An example of such a model is the probabilistic latent variable model proposed by Langseth and Nielsen [20].

The model described in [20] bears some resemblances with relational probabilistic models [13] in that it explicitly combines all users and items directly in the model [34, 33, 12]. More specifically, the model is a special type of conditional linear Gaussian model [21], where each item and user is represented by a collection of abstract latent variables encoding intrinsic properties about the user/item in question. The rating assigned to item i by user p is in turn modeled as a linear Gaussian distribution conditioned on the corresponding latent user/item representations. This joint representation of users and items allows the model to take advantage of all the user/item information available when making recommendations. Not only does this result in high-quality recommendations, as documented in [20], but it also supports a well-founded and principled way of making group recommendations [7, 25].

In order to learn the probabilistic latent variable models, Langseth and Nielsen [20] proposed an Expectation-Maximization (EM) algorithm tailored to the specific model class. Unfortunately, the algorithm requires the calculation of the full covariance matrix for all the latent variables representing the users and items. Consequently, the algorithm does not scale to larger data sets.

In this paper we address the scalability problem by proposing approximate learning and inference algorithms based on a variational Bayes approach [1, 3]. The algorithms employ a (generalized) mean-field approximation of the variational distribution, which ensures that the complexity of the learning algorithm grows *linearly* in the number of data points/ratings. Furthermore, we show that the model fits within the general class of statistical query models that in turn supports an efficient use of the MapReduce framework [8, 6]; hence the algorithms are easily parallelizable and can exploit distributed architectures. We empirically evaluate the proposed algorithms using several well-known data sets and demonstrate that the algorithm obtains results that are significantly better than what is obtained by a collection of straw-men methods. Finally, we analyze the performance of the method under cold-start conditions [19].

The remainder of the paper is structured as follows: In Section 2 we provide background information and describe the probabilistic latent variable model by Langseth and Nielsen [20]. Section 3 describes the variational Bayes based learning algorithm (with detailed derivations included in the appendix) and in Section 4 we present the empirical results. We conclude the paper in Section 5

and outline directions for future research.

2. A Latent model for Collaborative Filtering

2.1. Bayesian network

A Bayesian network (BN) is a probabilistic graphical model that defines a compact representation of a joint probability distribution by exploiting and explicitly encoding conditional independence properties among the variables. The specification of a BN over a collection of variables $\{X_1, \ldots, X_n\}$ consists of two parts: a qualitative part and a quantitative part. The qualitative part corresponds to an acyclic directed graph $G = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V}$ represent the variables $\{X_1, \ldots, X_n\}$ through a one-to-one mapping, and the edges $\mathcal{E}$ specify the direct dependencies between the variables. For ease of exposition, we shall refer to nodes and variables interchangeably.

We shall describe the relations between the variables in a Bayesian network using graph terminology. Thus, the nodes whose outgoing edges intersect a node/variable X_i are called the parents of X_i, denoted $\boldsymbol{\pi}_{X_i}$, and the nodes to which there exists an edge emanating from X_i are called the children of X_i. If there is a directed path from a node X_i to a node X_j, then X_j is said to be a descendant of X_i. Together the edges in the graph encode the conditional independence assumptions in the Bayesian network. Specifically, a node X_i is conditionally independent of its non-descendants given its parents.

The quantitative part is defined by a collection of conditional probability distributions or density functions s.t. each node is assigned exactly one probability distribution conditioned on its parents. In the remainder of this paper we shall assume that all variables are continuous. In particular, a variable X_i with parents $\boldsymbol{\pi}_{X_i}$ is assumed to follow a conditional linear Gaussian distribution

$$f(x_i|\boldsymbol{\pi}_{x_i}) = \mathcal{N}(\mu_i + \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{\pi}_{x_i}, \sigma_i),$$

i.e., the mean value is given as a weighted linear combination of the values of the parent variables whereas the variance is fixed. The underlying conditional independence assumptions encoded in the BN allow us to calculate the joint probability function using the chain rule:

$$f(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i|\boldsymbol{\pi}_{x_i}).$$

With linear Gaussian distributions assigned to all the variables it follows that the joint distribution is a multivariate Gaussian distribution. The inverse of the covariance matrix (also called the precision matrix) for this multivariate distribution directly reflects the independencies defined by the BN; the entry defined by a pair of variables is zero if and only if the two variables are conditionally independent given the other variables in the network.

2.2. A latent variable model

The collaborative filtering method proposed in [20] relies on a Bayesian network representation that provides a joint model of all items, users, and their ratings. Before presenting the details of the model, we shall first introduce some notation.

We will denote the matrix of ratings by $\mathbf{R}$, which is of size $\#U \times \#M$. Here $\#U$ is the number of users and $\#M$ is the number of items that are rated. $\mathbf{R}$ is a sparse matrix, meaning that it contains a considerable amount of missing values (more than 99% missing observations is quite common). The observed ratings are either realizations of ordinal variables (discrete variables with ordered states, e.g., "`Dislike`", "`Neutral`", "`Like`") or real numbers. In the following we will consider only continuous ratings encoded by real numbers, and assume that ratings given as ordinal variables have been translated into a numeric scale.

We use p as the index of an arbitrary person using the system, and i is the index of an item that can be rated. Consequently, $\mathbf{R}(p, i)$ is the rating that person p gives item i. Next, we will use $\delta(p, i)$ as an indicator function to show whether or not person p has rated item i. Specifically, $\delta(p, i) = 1$ if the rating exists and $\delta(p, i) = 0$ otherwise. Furthermore, $\mathcal{I}(p)$ is the set of items that person p has rated, i.e., $\mathcal{I}(p) = \cup_{i:\delta(p,i)\neq 0}\{i\}$, and similarly $\mathcal{P}(i) = \cup_{p:\delta(p,i)\neq 0}\{p\}$ is the set of persons that have rated item i. Lowercase letters are used to signify that a random variable is observed, so $\boldsymbol{r}(p, i)$ is the rating that p has given item i (that is, $\delta(p, i) = 1$ in this case). Finally, we let $\boldsymbol{r}$ denote all observed ratings (the part of $\mathbf{R}$ that is not missing).

When doing model-based collaborative filtering from a general perspective we look for a probabilistic model that for any item i and user p defines a probability distribution over $\mathbf{R}(p, i)$ given model parameters $\boldsymbol{\rho}$ and observed ratings $\boldsymbol{r}$. Given such a probability distribution, we can make recommendations based on the expected rating or the median rating for that distribution.

The probabilistic model that is proposed in [20] defines a joint distribution over all ratings by introducing abstract latent variable representations of both the items and the users. Specifically, each item i is represented by the random variables $\mathbf{M}_i$ and each user p is represented by the random variables $\mathbf{U}_p$. In a movie context one may for example interpret the different dimensions of $\boldsymbol{m}_i$ as representing different features of movie i such as to what extend the movie uses a well-known cast and the amount of explicit violence in the movie. Similarly, the dimensions of $\boldsymbol{u}_p$ can be interpreted as corresponding to different user characteristics. Hence, since the variables are continuous, the value $\boldsymbol{u}_{p,j}$ of the jth variable $\mathbf{U}_{p,j}$ can be interpreted as representing to what extent user p has the characteristics modeled by variable j. This also means that rather than assigning a user to a single "user group", the continuous variables $\mathbf{U}_{p,j}$ encode *to what extent* a user belongs to a certain group.[1] A priori we assume that $\mathbf{U}_p \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for $1 \leq p \leq \#U$, and $\mathbf{M}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for $1 \leq i \leq \#M$.

[1] See [20] for an empirical investigation into the possible semantics of the abstract latent variable representations.

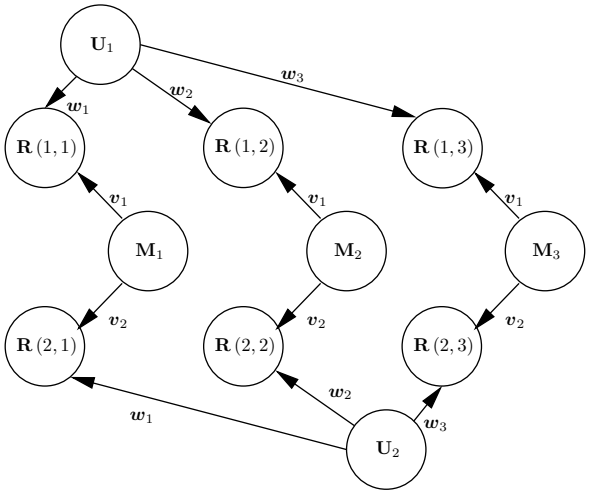

Figure 1: The full statistical model for collaborative filtering; this model has $\#M = 3$ and $\#U = 2$. The figure is from [20].

The rating assigned to item i by user p is modeled by assuming the existence of a linear mapping from the space describing users and items to the numerical rating scale:

$$\mathbf{R}(p,i) \,|\{\mathbf{M}_i = \boldsymbol{m}_i, \mathbf{U}_p = \boldsymbol{u}_p\} = \boldsymbol{v}_p^{\mathrm{T}} \boldsymbol{m}_i + \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{u}_p + \phi_p + \psi_i + \epsilon. \tag{1}$$

The rating in Equation (1) is thus determined as an additive combination of user p's preferences $\boldsymbol{v}_p$ for (or attitude towards) the features describing item i and item i's disposition $\boldsymbol{w}_i$ towards the different user groups.[2] The constants ϕ_p and ψ_i in Equation (1) can be interpreted as representing the average rating of user p and the average rating of item i (after compensating for the user average), respectively. Furthermore, ϵ represents "sensor noise", i.e., the variation in the ratings the model cannot explain. For mathematical convenience, we assume that $\epsilon \sim \mathcal{N}(0, \theta)$. It follows that the marginal distribution for $\mathbf{R}(p,i)$ can be written as

$$\mathbf{R}(p,i) \sim \mathcal{N}\left(\phi_p + \psi_i, \boldsymbol{v}_p^{\mathrm{T}} \boldsymbol{v}_p + \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{w}_i + \theta\right).$$

Finally it should be emphasized that we have the same number of latent variables for all users (i.e., $|\mathbf{U}_o| = |\mathbf{U}_p|$) and for all movies (i.e., $|\mathbf{M}_r| = |\mathbf{M}_i|$). Note, however, that $|\mathbf{M}_i|$ and $|\mathbf{U}_p|$ may differ. Figure 1 (taken from [20]) shows a BN representation of the proposed model for a domain with two users and three items.

As described in [20] this model has several desirable properties. It allows for a semantic interpretation of the features/variables characterizing users and items,

[2] Note that the relative importance of the movie features and the user group can be encoded in the weight vectors $\boldsymbol{v}_p$ and $\boldsymbol{w}_i$.

and it naturally provides support for making group recommendation. Furthermore, the model provides a generative perspective encompassing all ratings, users, and items simultaneously by entertaining a *global view* of the recommendation task. To see this, let us follow a chain of reasoning in the model depicted in Figure 1: Assume User 1 has already rated Item 1, so that $\boldsymbol{r}\,(1,1)$ has been observed by the system. When he also rates Item 2 (that is, $\mathbf{R}\,(1,2) = \boldsymbol{r}\,(1,2)$ is observed), one immediate effect is that the posterior distributions over $\mathbf{U}_1$ and $\mathbf{M}_2$ are updated to take the new information into account. Note that changing $\mathbf{U}_1$ gives the model a new perspective towards all ratings User 1 has given, in particular $\boldsymbol{r}\,(1,1)$: If $\mathbf{U}_1$ is changed, we get a new understanding of how that particular rating came to be, and this may in turn shed new light on Item 1. Thus, the encoding of Item 1, represented by the distribution over $\mathbf{M}_1$, should be altered. Next, the new posterior over $\mathbf{M}_1$ makes the model reconsider its representation of all users who have already rated Item 1, and thus the internal representation of those users must also be updated. This will again change the model's interpretation of all ratings that these users have given, and so on, quickly resulting in correlations between all ratings of all users. In general, the global perspective is both a desired property of the model as well as the source of a problem: From a predictive point of view, the model efficiently leverages information from *all* observed ratings when generating new recommendations. This results in high quality recommendations, as demonstrated in [20], where the recommendation accuracy of the model significantly outperform other collaborative filtering systems when evaluated using the MovieLens 100k data set [15]. On the other hand, since all latent variables in the model quickly become entangled, this leads to computational challenges that must be further addressed to ensure that the model scales to reasonably sized data sets. This is the main topic of the next section.

3. Learning the model from data

The learning algorithm described in [20] is based on the EM-algorithm [9]. Unfortunately, the computational complexity of that approach makes learning prohibitive for many real-world sized data sets. To see the problem, consider the rule for learning the weight $\boldsymbol{v}_p$, where the M-step [20] amounts to calculating

$$\begin{aligned}\boldsymbol{v}_p \leftarrow &\left[\tau/\theta \cdot \mathbf{I} + \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_i \mathbf{M}_i^{\mathrm{T}}\right]\right]^{-1} \times \\ &\left[\sum_{i \in \mathcal{I}(p)} \left(\boldsymbol{r}\,(p,i)\,\mathbb{E}\left[\mathbf{M}_i\right] - \mathbb{E}\left[\mathbf{M}_i \mathbf{U}_p^{\mathrm{T}}\right] \boldsymbol{w}_i - \mathbb{E}\left[\mathbf{M}_i\right](\psi_i + \phi_p)\right)\right],\end{aligned} \tag{2}$$

where τ is a parameter introduced by Tikhonov regularization of the learned weights. Notice that the terms $\mathbb{E}\left[\mathbf{M}_i \mathbf{M}_i^{\mathrm{T}}\right]$ and $\mathbb{E}\left[\mathbf{M}_i \mathbf{U}_p^{\mathrm{T}}\right]$ are required. They are established during the algorithm's E-step, where the covariance matrix over all latent variables must be calculated. To find the correlation between two

latent vectors we must first determine the full covariance matrix of all latent variables before the non-relevant variables can be marginalized out. The size of the full covariance matrix quickly becomes problematic; a modest data set containing $\#U = 10.000$ users and $\#M = 1.000$ items results in a covariance matrix with more than 10^8 entries that need to be calculated. The source for this computational problem is the model's global perspective (discussed in Section 2.2), which generally introduces a posteriori correlations among all the latent variables.

In what follows we propose two approaches for speeding up the learning of the parameters. Firstly, an alternative learning algorithm based on a variational Bayes method is developed. As we shall see, by relying on this learning algorithm, the complexity problem mentioned above is mitigated (in fact, each iteration of the algorithm is linear in the number of ratings, users, and items). Secondly, we position the proposed learning algorithm within a MapReduce context, thereby also achieving a framework that can exploit distributed architectures and is scalable with the size of the data.

3.1. Variational Bayes approximations

Our starting-point for the subsequent developments is the definition of the *variational Bayes* [2, 32] framework. An introduction to the variational Bayes procedure can be found in [16]. In its general form, one considers the random variables $(\boldsymbol{X}, \boldsymbol{Z})$, where $\boldsymbol{X} = \boldsymbol{x}$ is observed and we want to approximate $f(\boldsymbol{z}|\boldsymbol{x})$. We call the approximation $q(\boldsymbol{z})$, where we for simplicity of notation suppress that $q(\boldsymbol{z})$ depends on the observation $\boldsymbol{x}$. We measure the quality of the approximation by the KL distance from q to f, and obtain

$$\begin{aligned} D(q \,\|\, f) &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log\left(\frac{q(\boldsymbol{z})}{f(\boldsymbol{z}|\boldsymbol{x})}\right) d\boldsymbol{z} \\ &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log\left(\frac{q(\boldsymbol{z})}{f(\boldsymbol{z},\boldsymbol{x})}\right) d\boldsymbol{z} + \log(f(\boldsymbol{x})). \end{aligned}$$

By simple rearrangement, defining $\mathcal{F}(q) = -\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log\left(\frac{q(\boldsymbol{z})}{f(\boldsymbol{z},\boldsymbol{x})}\right) d\boldsymbol{z}$ and noting that $D(q \,\|\, f) \geq 0$, we get that

$$\log(f(\boldsymbol{x})) \geq \mathcal{F}(q).$$

It follows that finding the $q(\boldsymbol{z})$ that minimizes $D(q \,\|\, f)$ is equivalent to maximizing $\mathcal{F}(q)$, under the constraints that $q(\boldsymbol{z})$ is to be a probability density. Note that $\mathcal{F}(q) = -\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log\left(\frac{q(\boldsymbol{z})}{f(\boldsymbol{z},\boldsymbol{x})}\right) d\boldsymbol{z} = \mathbb{E}_q\left[\log\left(\frac{f(\boldsymbol{z},\boldsymbol{x})}{q(\boldsymbol{z})}\right)\right] = \mathcal{H}(q) + \mathbb{E}_q[\log f(\boldsymbol{z},\boldsymbol{x})]$, where $\mathcal{H}(q) = -\mathbb{E}_q[\log(q(\boldsymbol{z}))]$ is the entropy of $q(\cdot)$.

It turns out that we can maximize $\mathcal{F}(q)$ in a tractable way if we make structural assumptions about $q(\cdot)$. One popular strategy is to assume that $q(\boldsymbol{z})$ factorizes into smaller factors, like for instance its separate variables, $q(\boldsymbol{z}) = \prod_i q_i(z_i)$. This approach is commonly known as the *mean-field* approximation.

In this case, through calculus of variations, we find that $\mathcal{F}(q)$ is maximized by setting

$$\log q_i(z_i) = \mathbb{E}\left[\log\left(f(\boldsymbol{z}, \boldsymbol{x})\right)\right] + c, \tag{3}$$

where c is a constant and the expectation is wrt. all $Z_j \sim q_j$ s.t. $j \neq i$ [32]. In principle, this allows us to calculate the distribution for Z_i if we assume that the distribution for each Z_j, $j \neq i$, is known.

However, an iterative procedure will have to be followed in practice. When utilizing Equation (3) to find the optimal approximation for $q_1(z_1)$, say, the right-hand side of the equation will include expected values of (functions of) the other variables, that are calculated according to their respective approximate distributions. Thus, if, say, $q_2(z_2)$ is wrongly assessed, the error will in turn affect the estimate of $q_1(z_1)$, and so on. Fortunately, the contraction theorem applies, ensuring an eventual convergence to a local optimum [32]. To this end, $\mathcal{F}(q)$ is monitored and used to determine the termination of the iterations.

3.2. Variational Bayes in our model

The specification of the full generative model over $(\mathbf{R}, \mathbf{U}, \mathbf{M})$ given the parameters $\boldsymbol{\rho} = (\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{v}, \boldsymbol{w}, \theta)$ can be expressed as

$$f(\boldsymbol{r}, \boldsymbol{u}, \boldsymbol{m}|\boldsymbol{\rho}) = f(\boldsymbol{r}|\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho}) f(\boldsymbol{m}|\boldsymbol{\rho}) f(\boldsymbol{u}|\boldsymbol{\rho}),$$

where

$$\begin{aligned} f(\boldsymbol{r}|\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho}) &= \prod_{p=1}^{\#U} \prod_{i \in \mathcal{I}(p)} \sqrt{\frac{\theta}{2\pi}} \exp\left(-\frac{\theta}{2}(\boldsymbol{r}(p,i) - (\boldsymbol{v}_p^{\mathrm{T}} \boldsymbol{m}_i + \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{u}_p + \phi_p + \psi_i))^2\right); \\ f(\boldsymbol{m}_i|\boldsymbol{\rho}) &= \mathcal{N}(\mathbf{0}_s, \mathbf{I}_{s\times s}); \\ f(\boldsymbol{u}_p|\boldsymbol{\rho}) &= \mathcal{N}(\mathbf{0}_t, \mathbf{I}_{t\times t}). \end{aligned}$$

Further, in a Bayesian formulation of the problem, we give the following prior distributions to our parameters

$$\begin{gathered} f(\theta) = \mathrm{Gamma}(a, b), \quad f(\psi_i) = \mathcal{N}(\mu_\psi, 1/\kappa_\psi), \quad f(\phi_p) = \mathcal{N}(\mu_\phi, 1/\kappa_\phi) \\ f(\boldsymbol{w}_i) = \mathcal{N}(\mathbf{0}_s, 1/\tau \cdot \mathbf{I}_{s\times s}), \quad f(\boldsymbol{v}_p) = \mathcal{N}(\mathbf{0}_t, 1/\tau \cdot \mathbf{I}_{t\times t}). \end{gathered}$$

This allows us to, in principle, calculate $f(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}|\boldsymbol{r})$. Note that we have kept $\mu_\phi = 0$ fixed in the experiments reported in this paper, and for each experiment we have defined μ_ψ as the mid-point of the relevant rating-scale. Furthermore, we have found the behavior of the model to be rather robust wrt. the values of κ_ψ, κ_ϕ, a and b, and have for simplicity set each of them equal to one.

To cast the present problem into the formulation of Equation (3), we let $\boldsymbol{Z}$ denote all the latent variables and model parameters $\boldsymbol{Z} = (\mathbf{M}, \mathbf{U}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{v}, \boldsymbol{w}, \theta)$ and $\boldsymbol{X}$ be the part of $\mathbf{R}$ that is observed (the other ratings are *barren*, and can be disregarded during parameter learning).

Two flavors of the variational Bayes framework will be examined. The first one, which we will name *generalized mean-field* (GMF), takes as its starting point that the variational approximation of the full joint factorizes according to

$$q\left(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}\right) = q\left(\theta\right) \prod_{p=1}^{\#U} q\left(\phi_p\right) q\left(\boldsymbol{u}_p\right) q\left(\boldsymbol{v}_p\right) \prod_{i=1}^{\#M} q\left(\psi_i\right) q\left(\boldsymbol{m}_i\right) q\left(\boldsymbol{w}_i\right).$$

Thus, the posterior distribution over the variables of interest given the ratings are approximated by assuming independence between vectors of latent variables. For instance, the generalized mean-field approximation prescribes that $\mathbf{M}_i \perp\!\!\!\perp \mathbf{U}_p | \{\mathbf{R} = \boldsymbol{r}\}$, and $\mathbf{M}_i \perp\!\!\!\perp \mathbf{M}_j | \{\mathbf{R} = \boldsymbol{r}\}$. On the other hand, note that the dimensions of each random vector are seen as correlated in the posterior, e.g., $\mathbf{M}_{i,k} \not\perp\!\!\!\perp \mathbf{M}_{i,l} | \{\mathbf{R} = \boldsymbol{r}\}$ for a specific item i.

The second formulation, which is the standard mean-field (later to be referred to by MF), assumes that the posterior factorizes over all variables, giving us

$$\begin{aligned} q\left(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}\right) = q\left(\theta\right) & \prod_{p=1}^{\#U} \left(q\left(\phi_p\right) \prod_{j=1}^{|\mathbf{U}_p|} q\left(\boldsymbol{u}_{p,j}\right) q\left(\boldsymbol{v}_{p,j}\right) \right) \times \\ & \prod_{i=1}^{\#M} \left(q\left(\psi_i\right) \prod_{j=1}^{|\mathbf{M}_i|} q\left(\boldsymbol{m}_{i,j}\right) q\left(\boldsymbol{w}_{i,j}\right) \right), \end{aligned}$$

which introduces the additional set of assumptions that $\mathbf{M}_{i,k} \perp\!\!\!\perp \mathbf{M}_{i,l} | \{\mathbf{R} = \boldsymbol{r}\}$ and $\mathbf{U}_{p,k} \perp\!\!\!\perp \mathbf{U}_{p,l} | \{\mathbf{R} = \boldsymbol{r}\}$ further to those previously discussed. The details of the developments are given in Appendix A, so here we will only show one example and comment on the relevant time complexity of the calculations, namely the weight $\boldsymbol{v}_p$, used to model user p's preferences towards the items' latent representations.

Using the (generalized) variational Bayes machinery, $\mathbf{V}_p$ is now modeled as a random variable with posterior distribution $q\left(\boldsymbol{v}_p\right)$, which has the form of a multivariate Gaussian distribution, $q\left(\boldsymbol{v}_p\right) = \mathcal{N}\left(\mu_{\boldsymbol{v}_p}, \boldsymbol{Q}_{\boldsymbol{v}_p}^{-1}\right)$, where

$$\begin{aligned} \boldsymbol{Q}_{\boldsymbol{v}_p} &\leftarrow \tau \mathbf{I} + \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_i \mathbf{M}_i^{\mathrm{T}}\right] \qquad (4) \\ \mu_{\boldsymbol{v}_p} &\leftarrow \boldsymbol{Q}_{\boldsymbol{v}_p}^{-1} \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_i\right] \left(\boldsymbol{r}\left(p, i\right) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Psi_i\right] - \mathbb{E}\left[\Phi_p\right] \right). \end{aligned}$$

As can be seen, the calculation of $\mu_{\boldsymbol{v}_p}$ strongly resembles the calculation of the point-estimate when using the EM algorithm (Equation (2)), but slightly simplified by utilizing that $\mathbf{M}_i$ is constrained to be independent of the other random variables in the posterior $q\left(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}\right)$. The real benefit from a calculation point

of view, though, is due to the simplifications of the expectations coming into the calculations. Where the EM-algorithm using exact inference demanded that the covariance matrix over all latent variables was calculated to find $\mathbb{E}\left[\mathbf{M}_i\mathbf{M}_i^{\mathrm{T}}\right]$ and $\mathbb{E}\left[\mathbf{M}_i\mathbf{U}_p^{\mathrm{T}}\right]$, it is in the generalized mean-field model sufficient to look at each latent vector at a time (as they are assumed to be independent a posteriori); $\mathbb{E}\left[\mathbf{M}_i\mathbf{M}_i^{\mathrm{T}}\right]$ can be found directly from the posterior variational distribution for $\mathbf{M}_i$.

Finally, the standard MF approach results in $q\left(\boldsymbol{v}_{p,k}\right) = \mathcal{N}\left(\mu_{\boldsymbol{v}_{p,k}}, Q^{-1}_{\boldsymbol{v}_{p,k}}\right)$, where

$$
\begin{aligned}
Q_{\boldsymbol{v}_{p,k}} &\leftarrow \tau + \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_{i,k}^2\right] \qquad (5)\\
\mu_{\boldsymbol{v}_{p,k}} &\leftarrow Q^{-1}_{\boldsymbol{v}_{p,k}} \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_{i,k}\right] \Bigg(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Psi_i\right] - \mathbb{E}\left[\Phi_p\right] \\
&\qquad\qquad - \sum_{\ell \neq k} \mathbb{E}\left[\mathbf{M}_{i,\ell}\right] \mathbb{E}\left[\mathbf{V}_{p,\ell}\right] \Bigg).
\end{aligned}
$$

The logic behind the updating-rule remains the same, but the calculations are further simplified completely removing the need to invert matrices during the calculations. A Matlab implementation of the algorithm can be downloaded from `http://people.cs.aau.dk/~tdn/VB-CF/`.

3.3. Parallelization

MapReduce [8, 6] is a paradigm and framework to efficiently distribute data-intensive calculations in parallel over multiple computational cores/CPUs. This parallelization is most efficient when calculations are done concurrently and require no or little communication between the different calculation units. It has been shown [6] that *Statistical Query Models* (SQMs) [17] fit the MapReduce framework well. The mean-field approximation described above lead to calculation steps as in Equation (5), where – for a fixed p – the calculations amount to calculations of (weighted) sums over subsets of the data. Parallelization is immediate with subsets of data being summarized at each computational core. This comes as no surprise as the mean-field model is indeed an SQM. We have run our experiments on a configuration with relatively simple computational nodes (separate memory, shared disk). Here, data transfer and memory usage can be kept low by loading a separate subset of the data at each node, letting the mappers calculate the relevant statistics from that subset of the data, and having the reducer summarize the partial sums and finalize the calculations.

To see how this works in detail, assume that the rating data is separated into $\#\Gamma$ parts. Each part of the data is defined as a sparse matrix over the same domain as the full data set would occupy, but holding only a subset of the ratings. For simplicity of notation, assume that the calculations are done on a cluster with $\#\Gamma$ cores, and that each core is able to hold its dedicated subset of

the data in memory. When calculating $\mu_{\boldsymbol{v}_{p,k}}$ in Equation (5), we let each core γ calculate

$$\texttt{partial}(\gamma) \leftarrow \sum_{i \in \mathcal{I}(p|\gamma)} \mathbb{E}\left[\mathbf{M}_{i,k}\right] \Bigg(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Psi_i\right] - \mathbb{E}\left[\Phi_p\right] - \sum_{\ell \neq k} \mathbb{E}\left[\mathbf{M}_{i,\ell}\right] \mathbb{E}\left[\mathbf{V}_{p,\ell}\right] \Bigg),$$

where we use $\mathcal{I}(p|\gamma)$ to signify the set of items that person p has rated in the data set held by core γ. Then, the reducer simply calculates

$$\mu_{\boldsymbol{v}_{p,k}} \leftarrow Q^{-1}_{\boldsymbol{v}_{p,k}} \mathbb{E}\left[\Theta\right] \sum_{\gamma=1}^{\#\Gamma} \texttt{partial}(\gamma),$$

and the result is identical to Equation (5). Similar parallelization is readily available for all the remaining learning rules of the recommender model.

Note that the quantities entering into the calculations are either scalars ($\mathbb{E}\left[\mathbf{M}_{i,\ell}\right]$, $\mathbb{E}\left[\mathbf{V}_{p,\ell}\right]$, $\mathbb{E}\left[\Psi_i\right]$, $\mathbb{E}\left[\Phi_p\right]$, $Q_{\boldsymbol{v}_{p,k}}$, $\mathbb{E}\left[\Theta\right]$) or modestly sized vectors ($\mathbb{E}\left[\mathbf{U}_p\right]$, $\mathbb{E}\left[\mathbf{W}_i\right]$), and that the reducer step is computationally simple. The overhead due to the parallelization is therefore negligible, and in practice one will observe that the computational time is close to inversely proportional to $\#\Gamma$. This is in contrast to the maximum-likelihood learning [20]: In that case, the reducer would have to calculate $\mathbb{E}\left[\mathbf{M}_i \mathbf{U}_p^{\mathrm{T}}\right]$ (see Equation (2)). These expectations are found by calculating the inverse of the precision matrix containing all latent variables, which is a computationally daunting task.

4. Empirical results

In this section we will report on the results of the proposed collaborative filtering algorithm when run on a number of standard data sets. For the smaller data sets, where more computationally expensive straw-men can be employed, we will also report on those results for comparison.

4.1. Empirical setup

When learning the parameters of our model with a fixed model structure, we use the MF learning scheme described in Section 3. For the results reported in this section, we terminated the algorithm when the relative increase in the likelihood bound was less than 10^{-5} from one iteration to the next, or when the algorithm had run for 100 iterations. As we have already discussed, the GMF method is computationally more expensive than the standard MF. When learning a model using the generalized mean-field method we therefore initialize the learning algorithm by first learning a model using the MF algorithm, and then using the learned model as the starting-point for the GMF algorithm. Following this approach, GMF learning typically terminated after only 10–25 iterations.

Name	#ratings	#users	#items	Sparsity	Range
MovieLens 100k [15]	100.000	943	1.682	93.7%	$1\star - 5\star$
MovieLens 1M [15]	1.000.209	6.040	3.952	95.8%	$1\star - 5\star$
MovieLens 10M [15]	10.000.054	71.567	10.681	98.7%	$1\star - 5\star$
Last.fm	92.834	1.892	18.745	98.4%	$1\star - 5\star$
Jester [14]	4.136.630	73.421	100	43.7%	$[-10, +10]$
BookCrossing	433.671	77.805	185.973	99.9%	$1\star - 10\star$
Yahoo!	717.872.016	1.823.179	136.736	99.7%	$1\star - 5\star$

Table 1: Summary statistics for the data sets included in our study.

Deciding upon the model *structure* amounts to determining the number of latent variables to describe both users and items as well as the value for τ. When doing so, we used the same greedy strategy as described in [20]: The idea is to start from the simplest model structure, i.e., setting $|\mathbf{U}_p| = 1$, $|\mathbf{M}_i| = 1$, for all i and p, and setting $\tau = 1$.[3] τ is then gradually increased until the results, calculated using the *wrapper approach*, show that this is harmful for the predictive performance; for the experiments reported in this section we used four folds. Next we iteratively considered the neighboring models ($|\mathbf{U}_p| = 2, |\mathbf{M}_i| = 1$) and ($|\mathbf{U}_p| = 1, |\mathbf{M}_i| = 2$) in the same fashion, and at each step select the best scoring model as the current candidate model. When learning using the GMF algorithm, we first use the standard MF algorithm for structure learning, and then use the GMF algorithm only to calculate the posterior over the fixed structure.

4.2. The data sets

Before presenting the experimental results we first give some summary statistics for the data sets used in the experiments, see Table 1. For each data set we report its most commonly used name together with the number of ratings, users, and items. We also report the sparsity level, calculated as the fraction of item-user combination not having a rating, as well as the range of legal ratings. For the latter, numbers postfixed by stars denote integer ratings, whereas the interval for the Jester data set indicates real-valued ratings.

The MovieLens data sets [15] contain user supplied ratings of movies. There are three versions of the MovieLens data sets, namely *MovieLens 100k*, *MovieLens 1M*, and *MovieLens 10M*. The MovieLens 100k data set is supplied with five predefined folds for cross-validation, and these folds were also used during testing. For the two other MovieLens sets, we have randomly partitioned the data sets in a training set with 80% of the ratings and a test set with the remaining 20% of the ratings.[4]

[3] We found that the behaviour of the system was quite robust wrt. the values of a, b, κ_ψ and κ_ϕ. All these parameters were therefore rather arbitrarily set to 1 in the experiments reported in this paper.

[4] The MovieLens 10M data set also includes a five fold partitioning of the data. However,

The Last.fm data set[5] documents the number of times a user of the Last.fm service has listened to different music artists. In its original form, the data set contains information about the social networking activities, tagging, and music listening information of the users. In order to use the data within our framework, we have made a stripped down and re-coded version of the data set, focusing only on the amount of time that a user has listened to a particular artist: For each user, the 10% of the artists that the user has listened to the least are rated with "One star". The artists that appear between the 10% and 30% percentiles of the listening time are coded as "Two stars". The artists with a listening time between the 30% and 70% percentiles are recoded as "Three stars", and the artists the user have listened to from the 70% percentile and up to (and including) the 90% percentile are given "Four stars". The artists above the 90% percentile are given "Five stars". The encoding scheme introduces some particularities in the data. Firstly, all users have the same average rating and the same observed variance; in fact they have *identical* empirical distributions for their ratings. Secondly, the empirical distributions are symmetric; there are equally many ratings of one and five stars and similarly for two and four stars. We note that the simple structure of the ratings does not give the proposed model an unfair advantage over the straw-men models as it, e.g., still explicitly tries to capture potential differences in the average ratings between users through the user offset ϕ_p.[6]

The Jester data [14] contains ratings of jokes. This data set is not as sparse as the other data sets; 19.2% of the users have rated all the jokes, approximately 17% of the items have been rated by more than 90% of the users, and in total the sparsity level is 43.7%.

The original BookCrossing data set contains 1.149.780 ratings from 278.858 users with demographic information regarding a total of 271.379 books. For the empirical results reported in this paper, we have disregarded the demographic information. Further, the data set contains both explicit and implicit ratings. We have only considered the explicit ratings, leaving us with a smaller data set (see Table 1).

Finally, the Yahoo! data set contains users' ratings of songs. The data set also includes information about artist, album, and genre attributes, but this information has been disregarded for the experiments in the present paper.

4.3. Accuracy results

In the following subsections we report on the accuracy results of the models learned by the proposed algorithms using the data sets described above. For comparison we also evaluate the accuracy using the following straw-men methods:

---

the training/test partitions define disjoint user sets, and they are therefore not suitable for the present type of model learning and testing.

[5] `http://grouplens.org/datasets/hetrec-2011/`

[6] A script used to recode the data set is available from `http://people.cs.aau.dk/~tdn/VB-CF/`.

Pearson(k) denotes a memory-based approach, where the predicted rating of the active item is calculated as a weighted sum of the ratings given to the k items deemed most important (measured using Pearson correlation) wrt. the active item [15].

Euclidean(k) is the k-nearest neighbors algorithm, where the distance is calculated using the Euclidean norm [24].

SVD(λ) performs a singular value decomposition, where λ is the regularization weight. For each setting of λ we ran experiments with the number of dimensions ranging from one to twenty-five, and we present the best of these results here. Note that when choosing the number of dimensions based on the obtained results on the test set, we slightly favor the SVD algorithm over the other algorithms. Two options were considered for λ: $\lambda = 0$, resulting in a non-regularized model, and $\lambda = 0.01$ (as done by [30]). The learning was implemented with an adaptive learning rate. It was terminated when the relative improvement in error was lower than 10^{-5} or when the algorithm had run for 10.000 iterations.

The quality of recommendations are measured using the *Mean Absolute Error* (MAE). For the calculation of the MAE results in this section we rounded off the predicted ratings to the nearest integer value between one and five as this slightly improved the results.

4.3.1. The MovieLens data sets

In addition to the straw-men methods listed above, we also compared the accuracy results with the collaborative filtering model learned using the method described in [20], denoted EM in Table 2.

The results for MovieLens 100k are shown in Table 2, where we see that both the regular and generalized mean-field models outperform the straw-men models. The results in the scientific literature are not directly comparable to ours, mainly because the experimental settings are different. Many researchers using the MovieLens 100k data set have made their own training and test sets without further documentation. However, the reported MAE values are typically about 0.73 – 0.74 or poorer [15, 31, 26, 23, 27, 18, 5, 28, 22, 35] although results as low as 0.690 have also recently been reported [12]. See [20] for further discussion of the performance of these straw-men models.

The data sets MovieLens 1M and 10M do not come with predefined cross-validation folds and were instead randomly divided into two sets each: 80% for training and 20% for testing. The results of the comparison can also be seen in Table 2. For the MovieLens 10M data set we have only made a comparison based on SVD; the size of the data set makes direct use of the other straw-men models intractable.

Based on the MovieLens 10M data set, the mean-field learning algorithm produced a model with two latent variables representing the users and nineteen latent variables representing the items; the prior precision was set to 100. In comparison, the best SVD model uses $C = 15$ dimensions; the SVD model was

selected from a set of candidate models with dimensions $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20\}$. The mean-field model learned for the MovieLens 1M data set contains two latent variables representing users and 10 latent variables representing items.

	MovieLens		
	100k	1M	10M
Pearson(10)	0.7295	0.7433	—
Euclidean(10)	0.7446	0.7809	—
Pearson(25)	0.7080	0.7161	—
Euclidean(25)	0.7244	0.7532	—
Pearson(50)	0.7110	0.7046	—
Euclidean(50)	0.7328	0.7389	—
Pearson(all)	0.7122	0.7081	—
Euclidean(all)	0.7220	0.7229	—
SVD($\lambda = 0$)	0.6916	0.6829	0.6223
SVD($\lambda = 0.01$)	0.6869	0.6563	0.6099
EM	0.6848	—	—
MF	0.6745	0.6412	0.5953
GMF	0.6736	0.6412	0.5953

Table 2: The table shows the MAE results for the data sets MovieLens 100k, MovieLens 1M, and MovieLens 10M. Note that the values can only be compared vertically and not horizontally.

4.3.2. The Last.fm, Jester, and BookCrossing data sets

Due to the relatively small size of the modified Last.fm data set we have been able to compare the proposed method with all the straw-men methods. On the other hand, we were only able to compare the results of the proposed method with that of SVD for the Jester and BookCrossing data sets. The results can be found in Table 3.

4.3.3. The Yahoo data set

The size of the Yahoo! 700M music data set prohibit a full structural learning using the equipment at our disposal. Instead we have, somewhat arbitrarily, chosen to learn a regular mean-field model with four latent variables for both the users and the items; in total the learned model contains approximately 7.7 million latent variables. The learned model provides an MAE of 0.7942 based on the predefined training/test set division. In comparison, the best MAE result reported by MyMediaLite [11] is 0.81445 using a factorized matrix approach;[7] due to the size of the data set we have not been able to compare the method with the other straw-men methods listed above.

[7] `http://mymedialite.net/examples/datasets.html`

	Last.fm	Jester	BookCrossing
Pearson(10)	0.8915	—	—
Euclidean(10)	0.9485	—	—
Pearson(25)	0.8664	—	—
Euclidean(25)	0.9291	—	—
Pearson(50)	0.8601	—	—
Euclidean(50)	0.9209	—	—
Pearson(all)	0.8547	—	—
Euclidean(all)	0.9131	—	—
SVD($\lambda = 0$)	0.8405	3.3669	1.8362
SVD($\lambda = 0.01$)	0.8460	3.2545	1.7813
MF	0.8077	3.1335	1.1576
GMF	0.8077	3.1191	1.1576

Table 3: The tables shows the MAE results for the Last.fm, Jester and BookCrossing data sets. Note that MAE is not normalized wrt. the *range* of the ratings. The MAE values for Jester, which contains ratings between -10 and +10, are therefore larger than those for Last.fm (ranging from one to five) and BookCrossing (between one and ten).

4.4. Run-time performance

In this section we compare the run-time performance of the regular mean-field implementation (that does *not* exploit the MapReduce architecture) with the EM-algorithm described in [20] based on the MovieLens 100k data set. The results of the comparison can be seen in Figure 2, which shows a log-log plot of the learning time for sixteen different collaborative filtering models using the mean-field approach as well as the EM-algorithm described in [20]. The sixteen different models vary in the number of latent variables (ranging from one to four) used to descried the users and the items in the domain. As can be seen from the figure, the mean-field approach achieves a substantial performance improvement compared to the EM-algorithm, and, as demonstrated in Table 2, this improvement is obtained without a loss in precision.

4.5. Cold-start

In this section we investigate how vulnerable our model is to cold-start problems. For the investigation, we have used the MovieLens 100k data set with the pre-defined cross-validation folds, but with the following changes: For each cross-validation iteration, we modify the training set by randomly removing all but κ ratings from one fifth of the users (called the *cold-start users*). A model is then learned from this modified training set, and evaluated using MAE on the associated predefined test set (retaining only the cold-start users). We repeat the process five times to minimize the stochasticity of the results due to the random selection of which ratings to retain for the chosen users. Next, the process is repeated, by choosing a new set of cold-start users, and calculating the MAE for the five models learned when *their* ratings are partly removed. We continue in this way a total of five times, making sure that each user has been selected as a cold-start user exactly once. The MAE results from these 25 repetitions are

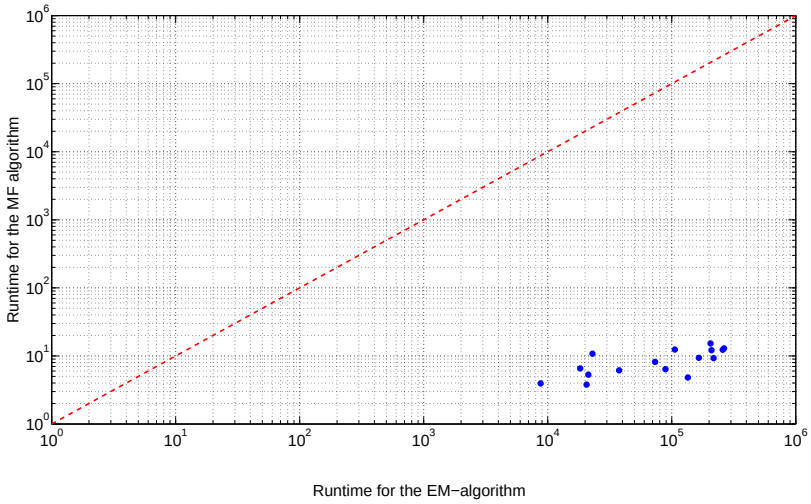

Figure 2: Log-log plot of the comparison of the runtime between the mean-field approach and the learning scheme described in [20]. The numbers relate to the MovieLens 100k data set [15].

averaged, and stored as the MAE on the first cross-validation fold. The same procedure is then repeated for the other four folds, thus in total requiring 125 runs of the learning algorithm.

For each cross-validation fold, we chose the structure ($|\mathbf{U}_p|$, $|\mathbf{M}_i|$, τ) by structural learning. For simplicity, the structure was kept fixed for all 25 replications inside one cross-validation fold, and was chosen to fit the *original* data set, i.e., the data set we had prior to the removal of any ratings. The results aggregated over all five cross-validation folds are reported in Figure 3. On the x-axis we show κ, the number of ratings left in the training set for the cold-start-users. On the y-axis we give the *cold-start efficiency*, which for a particular κ value is defined as the MAE of the full data set divided by the MAE obtained as above when the cold-start users had only κ ratings. An efficiency close to one thus means that the system has been able to learn almost all the information available about the cold-start users using only κ of these users' ratings. The figure shows the results for the mean-field algorithm in solid line with circular markers and the results of the SVD algorithm (dashed line and crosses). We note that not only does the mean-field algorithm obtain better results for the whole data set (as reported in Table 2), it is also better suited for cold-start, with an efficiency above 0.8 for $\kappa = 1$. The results for the generalized mean-field model was similar to the results of the mean-field model, but are omitted for clarity of the figure.

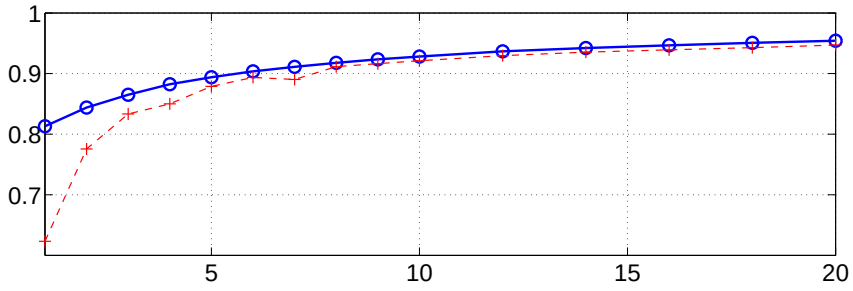

Figure 3: Efficiency vs κ for MF and SVD.

4.6. Summary of results

The first thing to notice from the results reported in this section is that the latent variable model described in Section 2.2 consistently and significantly outperforms the collection of straw-men methods on a wide range of data sets (see Table 2 and Table 3). Furthermore, strong results were documented in cold-start situations (Figure 3). Three different algorithms for learning the latent variable model have been evaluated:

- The EM algorithm relies on exact inference and is described in [20] for the latent variable model considered in this paper. The computational complexity of this algorithm is, however, problematic when considering realistically sized data sets. In this paper it thus serves as a point of reference for the two approximate algorithms (MF and GMF) being proposed.

- The MF algorithm is an approximate inference algorithm that assumes that every latent variable is independent of all the other latent variables a posteriori. This assumption, which improves the run-time performance with several orders of magnitude (Figure 2), violates the inherent modeling premises of the model. Still, as reported in Table 2, this apparent inconsistency does not lead to a loss in precision. In fact, a small improvement is observed. One possible explanation for this is the regularization effect produced by the prior distributions over the model parameters, thus reducing the risk/effect of over-fitting.

- The GMF alternative is a natural intermediate solution, where only some of the latent variables (namely those representing different aspects of a single item or person) are correlated a posteriori. The application of the GMF algorithm had only a modest impact on the results when compared to the MF approach, thus indicating that the extra modelling flexibility was not significant when evaluated on the data sets we considered. On the other hand, its computational complexity is higher than that of the MF algorithm, because it requires inversion of matrices that are in gen-

eral non-diagonal whereas the MF algorithm works with scalers (compare Equation (4) to Equation (5)).

We conclude that out of the three approaches examined, the MF algorithm appears to find the best balance between computational complexity and predictive ability for the data sets we have considered in this study.

5. Conclusion and future work

In this paper we have proposed two scalable algorithms for learning probabilistic collaborative filtering models that explicitly represents all users and items simultaneously [20]. The algorithms are based on the variational Bayes framework and differ in terms of the complexity of the variational distributions being applied. The computational complexity of the algorithms is linear in the number of ratings. Furthermore, both algorithms support a seamless parallel implementation that can easily be exploited in a MapReduce architecture. This allows for the processing of extremely large data sets, which we have illustrated by evaluating and comparing the algorithm based on the Yahoo 700M data set. The algorithms have also been evaluated on a collection of other publicly available collaborative filtering data sets and compared with well-known straw-men methods. The empirical results demonstrate that not only do the algorithms significantly outperform the straw-men methods, but we also observe a very favorable performance in cold-start scenarios. We observe only minor differences between the two algorithms wrt. prediction quality, and therefore do not find support for selecting the computationally more complex algorithm (GMF) over its simplified counterpart (MF) in our analysis. In particular, the simpler version is recommended for use in big data situations.

As part of our future work, we are currently exploring methods for extending the model and the learning algorithm to also include content information about users and items. We expect that this type of information will be encoded using discrete variables, thus producing a particular type of hybrid probabilistic collaborative filtering model.

Appendix A. Developments of variational Bayes inference

Appendix A.1. Model definition

In this appendix we will derive the updating rules for the variational Bayes learning and inference algorithms.

The specification of the full generative model over $(\mathbf{R}, \mathbf{U}, \mathbf{M})$ given the parameters $\boldsymbol{\rho} = (\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{v}, \boldsymbol{w}, \theta)$ can be expressed as

$$f(\boldsymbol{r}, \boldsymbol{u}, \boldsymbol{m}|\boldsymbol{\rho}) = f(\boldsymbol{r}|\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho}) f(\boldsymbol{m}|\boldsymbol{\rho}) f(\boldsymbol{u}|\boldsymbol{\rho}),$$

where

$$f(\boldsymbol{r}|\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho}) = \prod_{p=1}^{\#U}\prod_{i\in\mathcal{I}(p)}\sqrt{\frac{\theta}{2\pi}}\exp\left(-\frac{\theta}{2}(\boldsymbol{r}\,(p,i) - (\boldsymbol{v}_p^{\mathrm{T}}\boldsymbol{m}_i + \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{u}_p + \phi_p + \psi_i))^2\right)$$
$$f(\boldsymbol{m}_i|\boldsymbol{\rho}) = \mathcal{N}(\mathbf{0}_s, \mathbf{I}_{s\times s})$$
$$f(\boldsymbol{u}_p|\boldsymbol{\rho}) = \mathcal{N}(\mathbf{0}_t, \mathbf{I}_{t\times t}).$$

Further, in a Bayesian formulation of the problem, we give the following prior distributions to our parameters

$$f(\theta) = \mathrm{Gamma}(a,b), \quad f(\psi_i) = \mathcal{N}\left(\mu_\psi, 1/\kappa_\psi\right), \quad f(\phi_p) = \mathcal{N}\left(\mu_\phi, 1/\kappa_\phi\right)$$
$$f(\boldsymbol{w}_i) = \mathcal{N}\left(\mathbf{0}_s, 1/\tau\cdot\mathbf{I}_{s\times s}\right), \quad f(\boldsymbol{v}_p) = \mathcal{N}\left(\mathbf{0}_t, 1/\tau\cdot\mathbf{I}_{t\times t}\right).$$

This allows us to, in principle, calculate $f(\boldsymbol{u},\boldsymbol{m},\boldsymbol{\rho}|\boldsymbol{r})$. Note that we have kept $\mu_\phi = 0$ fixed in the experiments reported in this paper.

Appendix A.2. The generalized mean-field model

We will first assume a full variational joint distribution of the form

$$q\left(\boldsymbol{u},\boldsymbol{m},\boldsymbol{\rho}\right) = q\left(\theta\right)\prod_{p=1}^{\#U} q\left(\phi_p\right) q\left(\boldsymbol{u}_p\right) q\left(\boldsymbol{v}_p\right) \prod_{i=1}^{\#M} q\left(\psi_i\right) q\left(\boldsymbol{m}_i\right) q\left(\boldsymbol{w}_i\right),$$

i.e., the distribution factors into terms so that, e.g., $\mathbf{M}_i \perp\!\!\!\perp \mathbf{U}_p|\mathbf{R}$. On the other hand, note that $\mathbf{M}_{i,k} \not\perp\!\!\!\perp \mathbf{M}_{i,l}|\mathbf{R}$, etc.

Based on Equation (3), we get the following for $\mathbf{M}_i$, where $i \in \{1,\ldots,\#M\}$ is fixed:

$$\log q\left(\boldsymbol{m}_i\right) = \int_{\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}} q\left(\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}\right)\log f(\boldsymbol{r},\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho}) d\boldsymbol{u}\, d\boldsymbol{m}_{-i}\, d\boldsymbol{\rho} + \text{ const.}, \tag{A.1}$$

where $\boldsymbol{m}_{-i}$ is used as a shorthand for the collection of all $\boldsymbol{m}_j$ with $j \neq i$ and $\boldsymbol{u}$ denotes the collection of all $\boldsymbol{u}_p$, $p = 1,\ldots,\#U$.

The integral can be expanded as:

$$\begin{aligned}
&\int_{\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}} q\left(\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}\right)\log f(\boldsymbol{r},\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho}) d\boldsymbol{u}\, d\boldsymbol{m}_{-i}\, d\boldsymbol{\rho} \\
&= \int_{\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}} q\left(\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}\right)\sum_{j=1}^{\#M}\sum_{p\in\mathcal{P}(j)}\log f(\boldsymbol{r}\,(p,j)\,|\boldsymbol{m}_j,\boldsymbol{v}_p,\boldsymbol{u}_p,\boldsymbol{w}_j,\phi_p,\psi_j,\theta) d\boldsymbol{u}\, d\boldsymbol{m}_{-i}\, d\boldsymbol{\rho} \\
&\qquad + \log f(\boldsymbol{m}_i) + \text{ const.} \\
&= \int_{\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}} q\left(\boldsymbol{u},\boldsymbol{m}_{-i},\boldsymbol{\rho}\right)\sum_{p\in\mathcal{P}(i)}\log f(\boldsymbol{r}\,(p,i)\,|\boldsymbol{m}_i,\boldsymbol{v}_p,\boldsymbol{u}_p,\boldsymbol{w}_i,\phi_p,\psi_i,\theta) d\boldsymbol{u}\, d\boldsymbol{m}_{-i}\, d\boldsymbol{\rho} \\
&\qquad + \log f(\boldsymbol{m}_i) + \text{ const.},
\end{aligned}$$

where the constant is used to continuously collect all terms that do not depend on $\boldsymbol{m}_i$. Next, $\log f(\boldsymbol{r}\,(p,i)\,|\boldsymbol{m}_i, \boldsymbol{v}_p, \boldsymbol{u}_p, \boldsymbol{w}_i, \phi_p, \psi_i, \theta)$ can be written as follows (when all terms not involving $\boldsymbol{m}_i$ are continuously collected into the constant):

$$\begin{aligned}
&\log f(\boldsymbol{r}\,(p,i)\,|\boldsymbol{m}_i, \boldsymbol{v}_p, \boldsymbol{u}_p, \boldsymbol{w}_i, \phi_p, \psi_i, \theta) \\
&= -\frac{\theta}{2}\left(\boldsymbol{r}\,(p,i) - (\boldsymbol{m}_i^{\mathrm{T}}\boldsymbol{v}_p + \boldsymbol{u}_p^{\mathrm{T}}\boldsymbol{w}_i + \phi_p + \psi_i)\right)^2 + \text{ const.} \\
&= -\frac{\theta}{2}\boldsymbol{m}_i^{\mathrm{T}}\boldsymbol{v}_p\boldsymbol{v}_p^{\mathrm{T}}\boldsymbol{m}_i + \theta\cdot\boldsymbol{m}_i^{\mathrm{T}}\boldsymbol{v}_p\left(\boldsymbol{r}\,(p,i) - \boldsymbol{u}_p^{\mathrm{T}}\boldsymbol{w}_i - \phi_p - \psi_i)\right) + \text{ const.}
\end{aligned} \tag{A.2}$$

Consider now an r-dimensional Gaussian variable $\boldsymbol{X} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{Q}^{-1}\right)$, where $\boldsymbol{\mu}$ is the expected value and $\boldsymbol{Q}$ is the inverse variance, or *precision*. By simple calculation, and letting all terms that are not depending on $\boldsymbol{x}$ continuously disappear into the constant, we find that

$$\begin{aligned}
\log f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{Q}) &= \log\left((2\pi)^{r/2}\,|\boldsymbol{Q}|^{1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}-\boldsymbol{\mu})\right)\right) \\
&= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}-\boldsymbol{\mu}) + \text{ const.} \\
&= -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\mu} + \text{ const.}
\end{aligned} \tag{A.3}$$

Since $\mathbf{M}_i \sim \mathcal{N}(\mathbf{0}_s, \mathbf{I}_{s\times s})$, it follows that $\log f(\boldsymbol{m}_i) = -\frac{1}{2}\boldsymbol{m}_i^{\mathrm{T}}\boldsymbol{m}_i + \text{ const.}$ Utilizing Equation (A.2), the integral in Equation (A.1) can therefore be written as

$$\begin{aligned}
&\log q\,(\boldsymbol{m}_i) \\
&= -\frac{1}{2}\boldsymbol{m}_i^{\mathrm{T}}\left(\mathbf{I} + \mathbb{E}\left[\Theta\right]\sum_{p\in\mathcal{P}(i)}\mathbb{E}\left[\mathbf{V}_p\mathbf{V}_p^{\mathrm{T}}\right]\right)\boldsymbol{m}_i \\
&\quad + \mathbb{E}\left[\Theta\right]\boldsymbol{m}_i^{\mathrm{T}}\left(\sum_{p\in\mathcal{P}(i)}\mathbb{E}\left[\mathbf{V}_p\right]\left(\boldsymbol{r}\,(p,i) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Phi_p\right] - \mathbb{E}\left[\Psi_i\right]\right)\right) \\
&\quad + \text{ const.}
\end{aligned}$$

Comparing the terms of Equation (A.4) with those of Equation (A.3), we find that $q\,(\boldsymbol{m}_i)$ must be a Gaussian with precision

$$\boldsymbol{Q}_{\boldsymbol{m}_i} = \mathbf{I} + \mathbb{E}\left[\Theta\right]\sum_{p\in\mathcal{P}(i)}\mathbb{E}\left[\mathbf{V}_p\mathbf{V}_p^{\mathrm{T}}\right]$$

and expectation

$$\boldsymbol{Q}_{\boldsymbol{m}_i}^{-1}\mathbb{E}\left[\Theta\right]\sum_{p\in\mathcal{P}(i)}\mathbb{E}\left[\mathbf{V}_p\right]\left(\boldsymbol{r}\,(p,i) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Phi_p\right] - \mathbb{E}\left[\Psi_i\right]\right).$$

Using the same procedure as above we find that $q\,(\boldsymbol{v}_p)$ also has the form of a multivariate Gaussian distribution, $q\,(\boldsymbol{v}_p) = \mathcal{N}\left(\mu_{\boldsymbol{v}_p}, \boldsymbol{Q}_{\boldsymbol{v}_p}^{-1}\right)$, where

$$
\begin{aligned}
\boldsymbol{Q}_{\boldsymbol{v}_p} &= \tau\mathbf{I} + \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_i \mathbf{M}_i^{\mathrm{T}}\right]; \\
\mu_{\boldsymbol{v}_p} &= \boldsymbol{Q}_{\boldsymbol{v}_p}^{-1} \mathbb{E}\left[\Theta\right] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_i\right] \Big(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Phi_p\right] - \mathbb{E}\left[\Psi_i\right] \Big).
\end{aligned}
$$

Similar update rules are obtained for $q(\boldsymbol{u}_p)$ and $q(\boldsymbol{w}_i)$.

Next, we move on to Ψ_i, which can be interpreted as the a priori rating for item i. In our Bayesian formulation, $\Psi_i \sim \mathcal{N}\left(\mu_\psi, \kappa_\psi^{-1}\right)$, where μ_ψ and κ_ψ are the hyper-parameters, denoting the expectation and precision, respectively. Starting again from Equation (3) we have that

$$
\begin{aligned}
\log q\left(\psi_i\right) = & \int_{\boldsymbol{u},\boldsymbol{m},\theta,\boldsymbol{\phi},\boldsymbol{\psi}_{-i}} q\left(\boldsymbol{u},\boldsymbol{m},\theta,\boldsymbol{\phi},\boldsymbol{\psi}_{-i}\right) \log f(\boldsymbol{r},\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho})\, d\boldsymbol{u}\, d\boldsymbol{m}\, d\theta\, d\boldsymbol{\phi}\, d\boldsymbol{\psi}_{-i} \\
& + \text{ const.}
\end{aligned}
$$

As before we expand the integral, and simplify by continuously moving all terms not depending on ψ_i into the constant:

$$
\begin{aligned}
& \int_{\boldsymbol{u},\boldsymbol{m},\theta,\boldsymbol{\phi},\boldsymbol{\psi}_{-i}} q\left(\boldsymbol{u},\boldsymbol{m},\theta,\boldsymbol{\phi},\boldsymbol{\psi}_{-i}\right) \log\left[f(\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho}) f(\boldsymbol{r}|\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho})\right]\, d\boldsymbol{u}\, d\boldsymbol{m}\, d\theta\, d\boldsymbol{\phi}\, d\boldsymbol{\psi}_{-i} \\
& = \log f(\psi_i) + \int_{\boldsymbol{u},\boldsymbol{m},\theta,\boldsymbol{\phi},\boldsymbol{\psi}_{-i}} q\left(\boldsymbol{u}\right) q\left(\boldsymbol{m}\right) q\left(\theta\right) q\left(\boldsymbol{\phi}\right) q\left(\boldsymbol{\psi}_{-i}\right) \cdot \\
& \quad \frac{\theta}{2} \sum_{p \in \mathcal{P}(i)} \left(\boldsymbol{r}\left(p,i\right) - \left(\boldsymbol{m}_i^{\mathrm{T}} \boldsymbol{v}_p + \boldsymbol{u}_p^{\mathrm{T}} \boldsymbol{w}_i + \phi_p + \psi_i\right)\right)^2 d\boldsymbol{u}\, d\boldsymbol{m}\, d\theta\, d\boldsymbol{\phi}\, d\boldsymbol{\psi}_{-i} + \text{ const.} \\
& = \log f(\psi_i) - \frac{\mathbb{E}\left[\Theta\right]}{2} \sum_{p \in \mathcal{P}(i)} \Big[2\psi_i \mathbb{E}\left[\mathbf{M}_i\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{V}_p\right] + 2\psi_i \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] + \\
& \qquad\qquad 2\psi_i \mathbb{E}\left[\phi_p\right] + \psi_i^2 - 2\boldsymbol{r}\left(p,i\right)\psi_i \Big] + \text{ const.}
\end{aligned}
$$

Thus, we get

$$
\begin{aligned}
\log q\left(\psi_i\right) = & -\frac{1}{2}\psi_i^2 (\kappa_\psi + |\mathcal{P}(i)|\, \mathbb{E}\left[\Theta\right]) \\
& + \psi_i\, \mathbb{E}\left[\Theta\right] \sum_{p \in \mathcal{P}(i)} \left(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{M}_i\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{V}_p\right] - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}} \mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\phi_p\right]\right) \\
& + \psi_i\, \kappa_\psi \mu_\psi + \text{ const.}
\end{aligned} \tag{A.4}
$$

Comparing Equation (A.4) to Equation (A.3), we recognize that $q\left(\psi_i\right)$ is a

Gaussian with mean μ_{ψ_i} and variance $\sigma^2_{\psi_i}$, where:

$$\sigma^2_{\psi_i} = 1/(\kappa_\psi + |\mathcal{P}(i)|\,\mathbb{E}\left[\Theta\right]);$$
$$\mu_{\psi_i} = \sigma^2_{\psi_i}\left(\kappa_\psi\mu_\psi + \mathbb{E}\left[\Theta\right]\sum_{p\in\mathcal{P}(i)}\left(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{M}_i\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{V}_p\right] - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Phi_p\right]\right)\right).$$

Using the same procedure, we also find that $q\left(\phi_p\right)$ is a Gaussian distribution. We utilize that ϕ_p has a priori mean $\mu_\phi = 0$ to simplify the results slightly, and obtain that

$$\sigma^2_{\phi_p} = 1/(\kappa_\phi + |\mathcal{I}(p)|\,\mathbb{E}\left[\Theta\right]);$$
$$\mu_{\phi_p} = \sigma^2_{\phi_p}\mathbb{E}\left[\Theta\right]\sum_{i\in\mathcal{I}(p)}\left(\boldsymbol{r}\left(p,i\right) - \mathbb{E}\left[\mathbf{M}_i\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{V}_p\right] - \mathbb{E}\left[\mathbf{U}_p\right]^{\mathrm{T}}\mathbb{E}\left[\mathbf{W}_i\right] - \mathbb{E}\left[\Psi_i\right]\right).$$

Lastly, the distribution for the precision, $q\left(\theta\right)$ is to be calculated. The prior distribution of Θ is assumed to be a Gamma distribution with hyper-parameters a and b:

$$f(\theta) = \frac{b^a}{\Gamma(a)}\theta^{a-1}\exp(-b\theta).$$

As usual, we start from Equation (3) and obtain that

$$\begin{aligned}
&\log q\left(\theta\right)\\
&= \int_{\boldsymbol{u},\boldsymbol{m},\boldsymbol{\phi},\boldsymbol{\psi}} q\left(\boldsymbol{u},\boldsymbol{m},\boldsymbol{\phi},\boldsymbol{\psi},\theta\right)\log f(\boldsymbol{r},\boldsymbol{m},\boldsymbol{u},\boldsymbol{\phi},\boldsymbol{\psi},\theta)\,d\boldsymbol{u}\,d\boldsymbol{m}\,d\boldsymbol{\phi}\,d\boldsymbol{\psi} + \text{ const.}\\
&= \log f(\theta) + \int_{\boldsymbol{u},\boldsymbol{m},\boldsymbol{\phi},\boldsymbol{\psi}} q\left(\boldsymbol{u}\right)q\left(\boldsymbol{m}\right)q\left(\boldsymbol{\phi}\right)q\left(\boldsymbol{\psi}\right)\cdot\\
&\qquad \sum_{i=1}^{\#M}\sum_{p\in\mathcal{P}(i)}\log f(\boldsymbol{r}\left(p,i\right)|\boldsymbol{m}_i,\boldsymbol{v}_p,\boldsymbol{u}_p,\boldsymbol{w}_i,\phi_p,\psi_i,\theta)\,d\boldsymbol{u}\,d\boldsymbol{m}\,d\boldsymbol{\phi}\,d\boldsymbol{\psi} + \text{ const.}
\end{aligned}$$

Next, observe that when #Obs is defined as the total number of observed ratings,

$$\begin{aligned}
&\log f(\boldsymbol{r}|\boldsymbol{m},\boldsymbol{u},\boldsymbol{\rho})\\
&\quad= \log\left(\prod_{p=1}^{\#U}\prod_{i\in\mathcal{I}(p)}\sqrt{\frac{\theta}{2\pi}}\exp\left(-\frac{\theta}{2}(\boldsymbol{r}\left(p,i\right) - (\boldsymbol{v}_p^{\mathrm{T}}\boldsymbol{m}_i + \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{u}_p + \phi_p + \psi_i))^2\right)\right)\\
&\quad= \frac{\#\text{Obs}}{2}\log\left(\theta\right) - \frac{\theta}{2}\left(\boldsymbol{r}\left(p,i\right) - (\boldsymbol{m}_i^{\mathrm{T}}\boldsymbol{v}_p + \boldsymbol{u}_p^{\mathrm{T}}\boldsymbol{w}_i + \phi_p + \psi_i)\right)^2 + \text{ const.},
\end{aligned}$$

where the constant includes all terms not involving θ. Similarly, observe that

$$\begin{aligned}
\log f(\theta) &= a\log(b) - \log(\Gamma(a)) + (a-1)\log(\theta) - b\theta\\
&= (a-1)\log(\theta) - b\theta + \text{ const.}
\end{aligned} \tag{A.5}$$

It follows that

$$\log q(\theta) = \log(\theta)\left(a - 1 + \frac{\#\text{Obs}}{2}\right)$$
$$- \theta\left(b + \frac{1}{2}\sum_{p=1}^{\#U}\sum_{i\in\mathcal{I}(p)}\mathbb{E}\left[\left(\boldsymbol{r}(p,i) - \mathbf{U}_p^{\mathrm{T}}\mathbf{W}_i - \mathbf{M}_i^{\mathrm{T}}\mathbf{V}_p - \Phi_p - \Psi_i\right)^2\right]\right) + \text{ const.},$$

and by comparing this expression with the Gamma distribution (Equation (A.5)) we find that $q(\theta)$ is a Gamma distribution with parameters:

$$a^* = a + \frac{\#\text{Obs}}{2}$$
$$b^* = b + \frac{1}{2}\sum_{p=1}^{\#U}\sum_{i\in\mathcal{I}(p)}\mathbb{E}\left[\left(\boldsymbol{r}(p,i) - \mathbf{U}_p^{\mathrm{T}}\mathbf{W}_i - \mathbf{M}_i^{\mathrm{T}}\mathbf{V}_p - \Phi_p - \Psi_i\right)^2\right].$$

The convergence of the iterative learning scheme is controlled by monitoring the lower-bound of the marginal likelihood of the data, defined by $\mathcal{F}(q) = \mathcal{H}(q) + \mathbb{E}_q[\log f(\boldsymbol{r}, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho})]$, where $\mathcal{H}(q)$ is the entropy of the variational distribution. The calculation of this lower-bound is straightforward given the developments above.

Appendix A.3. Standard mean-field

The developments of the previous subsection were based on the assumption that the variational approximation factorizes according to

$$q(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}) = q(\theta)\prod_{p=1}^{\#U} q(\phi_p)\, q(\boldsymbol{u}_p)\, q(\boldsymbol{v}_p)\prod_{i=1}^{\#M} q(\psi_i)\, q(\boldsymbol{m}_i)\, q(\boldsymbol{w}_i).$$

We now take this one step further, assuming that the full joint $q(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho})$ has the form

$$q(\boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho}) = q(\theta)\prod_{p=1}^{\#U}\left(q(\phi_p)\prod_{j=1}^{|\mathbf{U}_p|} q(\boldsymbol{u}_{p,j})\, q(\boldsymbol{v}_{p,j})\right) \times$$
$$\prod_{i=1}^{\#M}\left(q(\psi_i)\prod_{j=1}^{|\mathbf{M}_i|} q(\boldsymbol{m}_{i,j})\, q(\boldsymbol{w}_{i,j})\right),$$

which introduces the set of additional assumptions that $\mathbf{M}_{i,k} \perp\!\!\!\perp \mathbf{M}_{i,l} | \mathbf{R}$ and $\mathbf{U}_{p,k} \perp\!\!\!\perp \mathbf{U}_{p,l} | \mathbf{R}$ in addition to those previously discussed. It turns out that these additional assumptions simplify the calculations of the approximate posteriors for $\mathbf{M}_i$, $\mathbf{U}_p$, $\mathbf{W}_i$, and $\mathbf{V}_p$ even further, while the developments for Φ_p, Ψ_i, and Θ remain unchanged.

Let us consider how to calculate $q(\boldsymbol{m}_{i,k})$, i.e., the distribution of the k'th element of the latent vector describing item i. Starting from Equation (3), we need to calculate the expectation of $\log f(\boldsymbol{r}, \boldsymbol{u}, \boldsymbol{m}, \boldsymbol{\rho})$ wrt. all random variables except $\mathbf{M}_{i,k}$. Continuously collecting all terms that are independent of $\boldsymbol{m}_{i,k}$ into the constant, and using the shorthand $\boldsymbol{m}_{-ik}$ for the collection of all $\boldsymbol{m}$-variables except $\boldsymbol{m}_{i,k}$, we obtain

$$
\begin{aligned}
&\log q(\boldsymbol{m}_{i,k}) \\
&= \int_{\boldsymbol{u},\boldsymbol{m}_{-ik},\boldsymbol{\rho}} q(\boldsymbol{u}, \boldsymbol{m}_{-ik}, \boldsymbol{\rho}) \log f(\boldsymbol{r}, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho}) d\boldsymbol{u}\, d\boldsymbol{m}_{-ik}\, d\boldsymbol{\rho} + \text{ const.} \\
&= \int_{\boldsymbol{u},\boldsymbol{m}_{-ik},\boldsymbol{\rho}} q(\boldsymbol{u}, \boldsymbol{m}_{-ik}, \boldsymbol{\rho}) \sum_{p \in \mathcal{P}(i)} \log f(\boldsymbol{r}(p,i) | \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{\rho}) d\boldsymbol{u}\, d\boldsymbol{m}_{-ik}\, d\boldsymbol{\rho} \\
&\qquad + \log f(\boldsymbol{m}_{i,k}) + \text{ const.} \\
&= -\frac{1}{2}\boldsymbol{m}_{i,k}^2 \left(1 + \mathbb{E}[\Theta] \sum_{p \in \mathcal{P}(i)} \mathbb{E}\left[\mathbf{V}_{p,j}^2\right] \right) \\
&\qquad + \boldsymbol{m}_{i,k} \mathbb{E}[\Theta] \sum_{p \in \mathcal{P}(i)} \mathbb{E}[\mathbf{V}_{p,j}] \Bigg(\boldsymbol{r}(p,i) - \\
&\qquad\qquad \mathbb{E}[\mathbf{U}_p]^{\mathrm{T}} \mathbb{E}[\mathbf{W}_i] - \sum_{\ell \neq k} \mathbb{E}[\mathbf{M}_{i,\ell}] \mathbb{E}[\mathbf{V}_{p,\ell}] - \mathbb{E}[\Phi_p] - \mathbb{E}[\Psi_i] \Bigg) \\
&\qquad + \text{ const.}
\end{aligned}
$$

Again, we find that $q(\boldsymbol{m}_{i,k})$ must be a Gaussian, this time with parameters

$$
\begin{aligned}
\sigma^2_{\boldsymbol{m}_{i,k}} &= \left(1 + \mathbb{E}[\Theta] \sum_{p \in \mathcal{P}(i)} \mathbb{E}\left[\mathbf{V}_{p,k}^2\right] \right)^{-1}; \\
\mu_{\boldsymbol{m}_{i,k}} &= \sigma^2_{\boldsymbol{m}_{i,k}} \mathbb{E}[\Theta] \sum_{p \in \mathcal{P}(i)} \mathbb{E}[\mathbf{V}_{p,k}] \Bigg(\boldsymbol{r}(p,i) - \mathbb{E}[\mathbf{U}_p]^{\mathrm{T}} \mathbb{E}[\mathbf{W}_i] - \\
&\qquad\qquad \sum_{\ell \neq k} \mathbb{E}[\mathbf{M}_{i,\ell}] \mathbb{E}[\mathbf{V}_{p,\ell}] - \mathbb{E}[\Phi_p] - \mathbb{E}[\Psi_i] \Bigg).
\end{aligned}
$$

In the previous sub-section we found that the variance of $q(\boldsymbol{m}_i)$ was given by $\left(\mathbf{I} + \mathbb{E}[\Theta] \sum_{p \in \mathcal{P}(i)} \mathbb{E}[\mathbf{V}_p] \mathbb{E}[\mathbf{V}_p]^{\mathrm{T}}\right)^{-1}$, hence the inversion of one $s \times s$ matrix (per item) was required to calculate the variational approximation. The matrixes are not diagonal in general, so if $\#M$ is large, the computational savings of the present result can be noteworthy, even for small values of s.

Using the same procedure as above we find that $q(\boldsymbol{v}_{p,k})$ also has the form of a multivariate Gaussian distribution, $q(\boldsymbol{v}_{p,k}) = \mathcal{N}\left(\mu_{\boldsymbol{v}_{p,k}}, Q^{-1}_{\boldsymbol{v}_{p,k}}\right)$, where $Q_{\boldsymbol{v}_{p,k}} = \tau + \mathbb{E}[\Theta] \sum_{i \in \mathcal{I}(p)} \mathbb{E}\left[\mathbf{M}_{i,k}^2\right]$ and $\mu_{\boldsymbol{v}_{p,k}} = Q^{-1}_{\boldsymbol{v}_{p,k}} \mathbb{E}[\Theta] \sum_{i \in \mathcal{I}(p)} \mathbb{E}[\mathbf{M}_{i,k}] (\boldsymbol{r}(p,i) - \mathbb{E}[\mathbf{U}_p]^{\mathrm{T}} \mathbb{E}[\mathbf{W}_i] - \sum_{\ell \neq k} \mathbb{E}[\mathbf{M}_{i,\ell}] \mathbb{E}[\mathbf{V}_{p,\ell}] - \mathbb{E}[\Phi_p] - \mathbb{E}[\Psi_i])$.

Finally, $q(\boldsymbol{u}_{p,k})$ and $q(\boldsymbol{w}_{i,k})$ can be found using the same set-up.

[1] Attias, H., 2000. A variational Bayesian framework for graphical models. Advances in neural information processing systems 12 (1-2), 209–215.

[2] Beal, M. J., 2003. Variational algorithms for approximate Bayesian inference. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

[3] Beal, M. J., Ghahramani, Z., 2006. Variational Bayesian learning of directed graphical models with hidden variables. Bayesian Analysis 1 (4), 793–831.

[4] Breese, J. S., Heckerman, D., Kadie, C., 1998. Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers, pp. 43–52.

[5] Chen, J., Yin, J., 2006. Recommendation based on influence sets. In: Proceedings of the Workshop on Web Mining and Web Usage Analysis.

[6] Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G. R., Ng, A. Y., Olukotun, K., 2006. Map-Reduce for machine learning on multicore. In: Schölkopf, B., Platt, J. C., Hoffman, T. (Eds.), Advances in Neural Information Processing Systems 19. MIT Press, pp. 281–288.

[7] de Campos, L. M., Fernández-Luna, J. M., Huete, J. F., Rueda-Morales, M. A., 2009. Managing uncertainty in group recommending processes. User Modeling and User-Adapted Interaction 19, 207–242.

[8] Dean, J., Ghemawat, S., 2004. MapReduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6. USENIX Association, pp. 137–150.

[9] Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39, 1–38.

[10] Duda, R., Hart, P., Stork, D., 2001. Pattern classification. Wiley Interscience.

[11] Gantner, Z., Rendle, S., Freudenthaler, C., Schmidt-Thieme, L., Oct. 2011. MyMediaLite. In: Proceedings of the fifth ACM conference on Recommender systems - RecSys '11. ACM Press, New York, New York, USA, p. 305.
URL `http://dl.acm.org/citation.cfm?id=2043932.2043989`

[12] Georgiev, K., Nakov, P., May 2013. A non-IID framework for collaborative filtering with restricted Boltzmann machines. In: Dasgupta, S., Mcallester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning (ICML-13). Vol. 28. JMLR Workshop and Conference Proceedings, pp. 1148–1156.
URL http://jmlr.org/proceedings/papers/v28/georgiev13.pdf

[13] Getoor, L., Taskar, B., 2007. Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press.

[14] Goldberg, K., Roeder, T., Gupta, D., Perkins, C., 2002. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4, 133–151.

[15] Herlocker, J., Konstan, J., Borchers, A., Riedl, J., 1999. An algorithmic framework for performing collaborative filtering. In: Proceedings of the ACM 1999 Conference on Research and Development in Information Retrieval. pp. 230–237.

[16] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., Saul, L. K., 1999. An introduction to variational methods for graphical models. Machine Learning 37, 183–233.

[17] Kearns, M., 1998. Efficient noise-tolerant learning from statistical queries. Journal of the ACM 45 (6), 983–1006.

[18] Kim, D., Yum, B.-J., 2005. Collaborative filtering based on iterative principal component analysis. Expert Systems with Applications 28 (4), 823 – 830.

[19] Kim, H.-N., El-Saddik, A., Jo, G.-S., Jun. 2011. Collaborative error-reflected models for cold-start recommender systems. Decision Support Systems 51 (3), 519–531.

[20] Langseth, H., Nielsen, T. D., June 2012. A latent model for collaborative filtering. International Journal of Approximate Reasoning 53 (4), 447–466.

[21] Lauritzen, S. L., 1992. Propagation of probabilities, means and variances in mixed graphical association models. Journal of the American Statistical Association 87 (420), 1098–1108.

[22] Lekakos, G., Caravelas, P., 2008. A hybrid approach for movie recommendation. Multimedia Tools and Applications 36, 5570.

[23] Li, Q., Kim, B. M., 2003. Clustering approach for hybrid recommender system. In: WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence. IEEE Computer Society, Washington, DC, USA, pp. 33–38.

[24] Marlin, B., 2004. Collaborative filtering: A machine learning perspective. Master of Science Thesis, Graduate Department of Computer Science, University of Toronto.

[25] Masthoff, J., 2011. Group recommender systems: Combining individual models. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. B. (Eds.), Recommender Systems Handbook. Springer US, pp. 677–702.

[26] Melville, P., Mooney, R., Nagarajan, R., 2002. Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence. The AAAI Press, pp. 178–192.

[27] Mobasher, B., Jin, X., Zhou, Y., 2003. Semantically enhanced collaborative filtering on the web. In: Web Mining: From Web to Semantic Web, First European Web Mining Forum, EMWF 2003. No. 3209 in Lecture Notes in Computer Science. pp. 57–76.

[28] Park, S.-T., Pennock, D., O. Madani, N. G., DeCoste, D., 2006. Naïve filterbots for robust cold-start recommendations. In: KDD 06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 699–705.

[29] Pennock, D. M., Horvitz, E., Lawrence, S., Giles, C. L., 2000. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers, pp. 473–480.

[30] Salakhutdinov, R., Mnih, A., Hinton, G., 2007. Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the Twenty-fourth International Conference on Machine Learning. Vol. 24. pp. 791–798.

[31] Schein, A., Popescul, A., Ungar, L., Pennock, D., 2001. Generative models for cold-start recommendations. In: Proceedings of the 2001 SIGIR Workshop on Recommender Systems.

[32] Šmídl, V., Quinn, A., 2006. The variational Bayes method in signal processing. Springer-Verlag.

[33] Truyen, T. T., Phung, D. Q., Venkatesh, S., 2009. Ordinal Boltzmann machines for collaborative filtering. In: Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence. pp. 548–556.

[34] Xu, Z., Tresp, V., Yu, K., Kriegel, H.-P., 2006. Infinite hidden relational models. In: Proceedings of the Twenty-second Conference on Uncertainty in Artificial Intelligence. pp. 544–551.

[35] Yoshii, K., Goto, M., Komatani, K., Ogata, T., Okuno, H., 2008. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. IEEE Transaction on Audio, Speech and Language Processing 16, 435–447.