A Framework for Matrix Factorization Based on General Distributions

Josef Bauer
Katholische Universität Eichstätt-Ingolstadt
Auf der Schanz 49
85049 Ingolstadt, Germany
josef.bauer@ku.de

Alexandros Nanopoulos
Katholische Universität Eichstätt-Ingolstadt
Auf der Schanz 49
85049 Ingolstadt, Germany
alexandros.nanopoulos@ku.de

ABSTRACT

In this paper we extend the current state-of-the-art matrix factorization method for recommendations to general probability distributions. As shown in previous work, the standard method called "Probabilistic Matrix Factorization" is based on a normal distribution assumption. While there exists work in which this method is extended to other distributions, these extensions are restrictive and we experimentally show on the basis of a real data set that it is worthwhile considering more general distributions which have not been used in the literature. Our contribution lies in providing a flexible and easy-to-use framework for matrix factorization with almost no limitation on the form of the distribution used. Our approach is based on maximum likelihood estimation and a key ingredient of our proposed method is automatic differentiation. This allows for the automatic derivation of the corresponding optimization algorithm, without the need to derive it manually for each distributional assumption while simultaneously being computationally efficient. Thus, with our method it is very easy to use a wide range of even complicated distributions for any data set.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

Machine Learning; Matrix Factorization; Maximum Likelihood; Automatic Differentiation

1. INTRODUCTION

Matrix factorization (MF) is a model-based collaborative filtering method that has shown its ability to build accurate prediction models for recommender systems [7]. MF generates recommendations based on latent features of users and items that determine the preferences of users for given items. The only required information to apply MF is the past behavior of users. A state-of-the-art matrix factorization method is the Probabilistic Matrix Factorization (PMF) [13], which relies on a normal distribution assumption.

Our key innovation lies in providing a flexible framework that extends PMF to almost arbitrary probability distributions. Although there exist relevant extensions of PMF to other distributions in previous work, which we discuss in Section 2, these extensions are limited. As we show by using a real quantitative implicit feedback data set, the prediction accuracy can indeed be improved by considering more general distributions. While on the one hand we provide an algorithmic framework that enables the use of general distributions, we address on the other hand the applicability of this framework by automating it, thus overcoming the need to handle each distribution manually.

One important finding is that the estimation method for all the different models with various distributional assumptions take almost the same form, in which the partial derivatives of the particular density function are the only difference. A key advantage of our approach is automatic differentiation, with which it is possible that the corresponding optimization algorithm for every distribution is derived automatically. The way we use automatic differentiation, in particular forward-mode automatic differentiation, can be seen as a source code transformation. Hereby, the relevant functional expressions in symbolical form are parsed and known differentiation rules are applied inductively. These transformed expressions can then be used to automatically generate a function which depends on the same parameters as the original function and the output of this new function will be the same as the respective derivative of the original function. This function can then be called by the program. This way, no numerical approximation of the derivatives is needed and this has to be done only once for each distribution model, which makes it efficient. Particularly, the running time and memory requirements do not increase more strongly with the size of the data then manually coded analogues. More details on its theoretical background can be found, for example, in section 8.2 of [9].

The rest of this paper is organized as follows. First, we discuss related work in Section 2 and introduce standard PMF in Section 3.1. Then we present our problem formulation and modeling approach in Section 3.2. Afterwards, we present our parameter estimation approach based on the statistical method of maximum likelihood and the corresponding gradient-based optimization method in Sections 4.1 and 4.2, respectively. After addressing how predictions can be generated in Section 4.3 and outlining promis-

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.
Copyright 2014 ACM 978-1-4503-2668-1/14/10 ...$15.00.
http://dx.doi.org/10.1145/2645710.2645735.

ing applications in Section 4.4, we focus on the experiments in Section 5. Finally, we conclude with a summary and a discussion of future work in Section 6.

2. RELATED WORK

As we mentioned previously, our proposed method can be seen as an extension of the standard matrix factorization method PMF [13], which assumes a normal distribution, to almost arbitrary distributions. There are other approaches in the literature that extended PMF in several directions. Closely related to our method is the work of [15]. There it is shown that PMF can be extended to a more general MF framework which comprises many common models. The underlying distribution does not need to be normal in this case, but is allowed to be a member of the natural exponential family with a single variable parameter. The subsumed models in this setting can be described equivalently by a few modeling choices. Basically, this approach can be seen as a combination of MF and Generalized Linear Models (GLMs). Thereby, one can obtain a unified estimation method which corresponds to the minimization of a specific Bregman divergence that depends on the distribution. However, assuming that the distribution is from the natural exponential family is still restrictive. Our general approach does not make this assumption. For instance, in our experiments we investigate (among others) the log-normal and Pareto distributions. The log-normal distribution in particular achieves the best performance in our results. It belongs to the general exponential family, but is not a member of the natural exponential family, which corresponds to the definition of a regular exponential family in [15]. Thus, this distribution is not part of the framework provided in [15] like many other distributions, including power law distributions such as the Pareto distribution. Furthermore, in [15] only predictions of the mean are addressed (by using prediction link functions as in GLMs), while we focus on the whole distribution including quantiles such as the median, which is more appropriate when dealing with quantitative implicit feedback (e.g., number of downloads for a song or amount of items sold), because it is more robust. Also, this method is not as flexible while still being easily applicable as our general maximum likelihood approach, since it does not allow for basic transformations of the user-item interactions and the biases and does not make use of computational advantages like automatic differentiation. On the other hand, the framework in [15] provides ways to deal with multi-relational data as well as with constraints on the factor matrices. Future work includes extensions of our proposed method that are capable of these additional advantages as well.

Another related generalization concept of standard MF are Factorization Machines (FMs) [11, 12] and special cases of them like in [3]. The main difference to our proposed method is the fact that FMs are developed in terms of certain loss functions and particular losses are proposed for a given task. For instance, squared loss for all kinds of regression, which corresponds to a normal distribution assumption like in PMF, regardless of the actual distribution of the data. In contrast, our method relies on flexible probability distributions than can be chosen in accordance with the data and we show that it is indeed beneficial not to make such restrictive assumptions as FMs do in this respect. On the other hand, the strength of FMs lies in combining many previously proposed MF models into one customizable framework. By feature engineering, FMs are able to mimic attribute- and context-aware approaches or to include other side information like temporal information etc. In fact, due to these advantages it is promising for future work to extend our proposed method with the benefits provided by FMs.

Furthermore, there exists work related to our application in which we focus on implicit feedback in the form of count data. According to [6], one can deal with implicit feedback in the form of frequency data by thresholding. Here, one only distinguishes whether a count value is below a fixed threshold or not, which results in binary implicit feedback. Other approaches in the literature deal directly with count data. [4] considers an advertisement problem setting, [8] addresses web site recommendations and [5] deals with location-based social networks. All these approaches extend the standard PMF model by using a Poisson distribution assumption and more precisely a variant of the GaP-model, which was developed for document data in [2]. Again, the crucial difference to our approach is that these works are based on a single model with a fixed distributional assumption while we provide a flexible framework that allows the use of almost arbitrary distributions. In our experiments we show that this is indeed valuable, since a log-normal distribution is better than a Poisson distribution for our data set. Furthermore, their way of using the Poisson distribution is different from ours. While in [4, 8, 5] gamma priors are assumed, we use L^2-norm-based regularization of the rows of the factor matrices in our application, like it is done in the standard PMF method for a normal distribution. Additionally, we use customer and item biases in our model because in [7] it is shown that this results in improvements of the prediction accuracy for the standard PMF method assuming a normal distribution. Also, we demonstrate that it is worthwhile to use a shifted version of the Poisson distribution instead of the usual one. In contrast to previous work, such adaptions can be easily done within our framework.

3. MODELING APPROACH

3.1 Probabilistic Matrix Factorization Based on a Normal Distribution Assumption

Here, we provide a short summary on the Probabilistic Matrix Factorization method [13], which assumes a normal distribution. In fact, we present an enhancement of this based on [7], which takes biases into account, since these will also be an ingredient in our general framework. This method has proved to be very successful in applications with explicit ratings like the Netflix Million Dollar Prize Challenge. Let us assume we have n users, m items and a user-item matrix $X \in \mathbb{R}^{n \times m}$, in which we identify every user by the row index $i \in \{1, ..., n\}$ and every item by the column index $j \in \{1, ..., m\}$. Each entry X_{ij} of this matrix represents the rating of item j given by user i in case it is available, e.g., measured in 1–5 stars. The goal in this setting is to infer the missing entries given the observed ones. And the key idea in this model is that users and items can be characterized by "latent features". For instance, this could be the preference for a certain genre of music and the corresponding feature of the item (song) represents the extent it belongs to this genre. However, no apriori assumptions are needed on the type of the features, but they are learned by the algorithm. Now, suppose there exist $k \in \mathbb{N}$ latent features. We store these features for all users in a matrix $U \in \mathbb{R}^{n \times k}$ and for all items in a matrix $V \in \mathbb{R}^{m \times k}$. Hereby, the interaction between user i and item j is modeled by the dot product of the corresponding feature vectors, namely $U_{i.} V_{j.}^T = \sum_{l=1}^{k} U_{il} V_{jl}$, where $U_{i.}$ denotes the i-th row of U, $V_{j.}$ the j-th row of V and V^T is the transpose of matrix V. In this model, a higher value of $U_{i.} V_{j.}^T$ means a higher preference of user i for item j, which in turn suggests a higher value of the expected rating X_{ij}. In many cases, however, this expected rating will not only depend on such an interaction, but it can partly be explained by characteristics that depend only on the user or the item. Some users tend to rate items higher than others and some items are overall more popular than the remaining ones. Thus, biases are included to capture these effects, which are modeled as variables representing the deviations between the average rating of a particular user or item and the overall mean rating μ. Let us denote by $b_{U_{i.}} \in \mathbb{R}$ the bias of user i and by $b_{V_{j.}} \in \mathbb{R}$ the bias of item j. As shown in [13], the standard method can be derived by assuming a normal distribution of all ratings. In this model, missing ratings are not included in the optimization. If we denote by $\mathcal{O}$ the set of observed rating values, by this approach the maximization of the corresponding likelihood function is equivalent to the following optimization problem:

$$\begin{aligned} \underset{U,V,b_U,b_V}{\arg\min} \sum_{(i,j):\, X_{ij} \in \mathcal{O}} \Bigg(& \left(X_{ij} - U_{i.} V_{j.}^T - \mu - b_{U_{i.}} - b_{V_{j.}} \right)^2 \\ & + \lambda \left(\sum_{l=1}^{k} \left((U_{il})^2 + (V_{jl})^2 \right) + (b_{U_{i.}})^2 + \left(b_{V_{j.}} \right)^2 \right) \Bigg) \end{aligned} \tag{1}$$

In (1), $\lambda > 0$ is a constant adjusting the influence of the regularization term in order to prevent overfitting. The model parameters that have to be estimated are given by the latent feature matrices U and V as well as the bias vectors b_U of the n user biases and b_V of the m item biases. A common method to solve this is stochastic gradient descent, which we describe below in a more general fashion.

3.2 Problem Statement and Modeling

Now, we focus on generalizing PMF to a framework that allows for the automatic use of general distributions. As mentioned in Section 2, previous work extended standard PMF to natural exponential family distributions with one real-valued parameter. However, this is still a major limitation on the possible distributions. In contrast, in our framework there is almost no restriction on the distribution. And in Section 4.2 we show that this generalization can be accomplished without additional costs by using automatic differentiation to derive the corresponding optimization algorithm. We keep the same notation as before and now treat each rating X_{ij} of item j by user i as a realization of a random variable whose distribution is determined by the corresponding probability density function $f(. \mid \theta_{ij}, \vartheta)$, with a hyperparameter ϑ and a user and item dependent parameter θ_{ij} which is specified below. The specific value θ_{ij} of the variable parameter θ determines the form of the distribution for the rating of item j by user i. In general, θ can represent a vector, so that we have in fact multiple user and item dependent parameters. This way, almost any distribution can be modeled. In our application with quantitative implicit feedback, we use for example the log-normal and the Pareto distribution, which are not from the natural exponential family. One promising example for future work with explicit feedback is the skew normal distribution, which is a generalization of the normal distribution that is able to deal with positive and negative skewness. For simplicity of presentation, in the following we focus on a real-valued parameter θ. For a vector-valued parameter θ, the same framework applies, whereby each component of this vector is treated the same way as described below. In this case, in each step of the algorithm the updates of the parameters corresponding to each component can be done successively while the remaining parameters are fixed. In our experiments, we have also used a real-valued parameter θ. Future work will investigate more complex distributions with θ being a parameter vector. While in classical PMF the ratings usually represent explicit ratings from a discrete set, we consider real-valued X_{ij} that can, for example, arise implicitly. In particular, in our experiments we focus on the case of quantitative implicit feedback data in the form counts, where $X_{ij} \in \mathbb{N}_0$. (For consistency with existing literature we still refer to such counts as ratings, also assuming that higher count values –e.g., number of downloads for a song– express higher preference.) We assume that all X_{ij} are independent and we require that $\theta \mapsto f(x \mid \theta, \vartheta)$ is differentiable for every fixed x, although some nondifferentiable points could be dealt with by subgradient methods. By differentiability, the existence of the corresponding partial derivatives in the gradient descent-based optimization method in Section 4.2 is ensured. This assumption is satisfied for most distributions in general and for all that are used in the present paper.

For the specification of each parameter θ_{ij}, we use the same building blocks as in the standard method, namely the user-item interaction $U_{i.} V_{j.}^T$ and the bias term $\mu + b_{U_{i.}} + b_{V_{j.}}$, where in our case the number μ can problem dependent represent an average (like the mean or median) or zero. However, we will not only focus on the sum of both parts, but rather consider a transformation of both ingredients. For this purpose, in the following let τ denote a transformation function which maps the bias term and the user-item interaction to a real number, i.e., $\tau : \mathrm{dom}(\tau) \subseteq \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto \tau(x, y)$, and we require that τ is differentiable with respect to x and y. This transformation function can be used for reparameterization purposes, for instance to allow the convenient calculation of a target measure of interest. Appropriate choices of τ for different distributions and targets will become clear below in Section 4.3. Often, for example in case of a normal distribution, we can simply set $\tau(x, y) = x + y$, which leads to the same form as in standard PMF with biases. We then model every θ_{ij} by this transformation, therefore having

$$\theta_{ij} = \tau\left(U_{i.} V_{j.}^T, \mu + b_{U_{i.}} + b_{V_{j.}} \right). \tag{2}$$

4. MAXIMUM LIKELIHOOD ESTIMATION BASED ON AUTOMATIC DIFFERENTIATION AND GENERATION OF PREDICTIONS

In the following, we first derive our maximum likelihood-based objective function and afterwards we develop an efficient (stochastic) gradient-based learning algorithm which uses automatic differentiation as a key ingredient. Finally, we address the generation of predictions.

4.1 Maximum Likelihood-based Problem Formulation

Under the assumption that the X_{ij} are independent, the joint density factorizes into the product of all marginal densities. Thus, in this general fashion, this leads at first to the following maximum likelihood optimization problem:

$$\underset{U,V,b_U,b_V}{\arg\max}\left(\prod_{\substack{i\in\{1,..,n\},\\ j\in\{1,..,m\}}} f\left(X_{ij}\,|\theta_{ij},\vartheta\right)\right), \tag{3}$$

where θ_{ij} is given by (2) and we are initially assuming that all X_{ij} are observed and equally important. In (3), the X_{ij} denote observed values of the corresponding random variables, e.g., concrete explicit or implicit ratings. For the parameter estimation, we minimize the negative log-likelihood function, since this is equivalent to maximizing the likelihood function and the resulting sums are more convenient. Furthermore, we also use regularization in order to prevent overfitting. In a more general formulation, let $R_{ij} \in \mathbb{R}_0^+$ denote the regularization term for all parameters associated with X_{ij}, which are given by $U_{i.}$, $b_{U_{i.}}$, $V_{j.}$ and $b_{V_{j.}}$. Additionally, we use observation weights $W_{ij} \in \mathbb{R}_0^+$ for $i \in \{1,..,n\}$ and $j \in \{1,..,m\}$ in order to handle cases when certain observations are more important than others. By this approach, the negative log-likelihood minimization problem becomes:

$$\underset{U,V,b_U,b_V}{\arg\min}\left(-\sum_{(i,j)\colon W_{ij}>0}\left(W_{ij}\log\left(f\left(X_{ij}\,|\theta_{ij},\vartheta\right)\right)-R_{ij}\right)\right) \tag{4}$$

Here, the summation is done over all user-item pairs (i,j) for which the corresponding observations X_{ij} have non-zero weights. In our experiments, where the X_{ij} are count data, we have used either $W_{ij} = 0$ (in case of missing values, noise and outliers) or $W_{ij} = 1$, meaning that we include the observation X_{ij} and treat all of the included observations equally. Furthermore, for simplicity we have used the common way of L^2-norm-based regularization given by

$$R_{ij} := \lambda\left(\sum_{l=1}^{k}\left((U_{il})^2+(V_{jl})^2\right)+\left(b_{U_{i.}}\right)^2+\left(b_{V_{j.}}\right)^2\right), \tag{5}$$

with one regularization constant $\lambda > 0$.

4.2 Gradient-based Optimization Algorithm

First, we derive the gradients with respect to all parameters of our objective function in (4) for a variable density f. Afterwards, we will use these for a gradient descent-based optimization algorithm that can make use of automatic differentiation. Let us denote the objective function in (4) by

$$F\left(U,V,b_U,b_V\right) := \tag{6}$$
$$-\sum_{(i,j)\colon W_{ij}>0}\left(W_{ij}\log\left(f\left(X_{ij}\,|\theta_{ij},\vartheta\right)\right)-R_{ij}\right)$$

After taking partial derivatives and applying the chain rule, we obtain the following gradients (in matrix calculus notation) for every i for which there exists at least one j such that $W_{ij} > 0$ and every j for which exists at least one i such that $W_{ij} > 0$:

$$\frac{\partial}{\partial U_{i.}}F\left(U,V,b_U,b_V\right) = -\sum_{j\colon W_{ij}>0}\left(W_{ij}D_{ij}V_{j.}-\frac{\partial}{\partial U_{i.}}R_{ij}\right) \tag{7}$$

$$\frac{\partial}{\partial V_{j.}}F\left(U,V,b_U,b_V\right) = -\sum_{i\colon W_{ij}>0}\left(W_{ij}D_{ij}U_{i.}-\frac{\partial}{\partial V_{j.}}R_{ij}\right) \tag{8}$$

$$\frac{\partial}{\partial b_{U_{i.}}}F\left(U,V,b_U,b_V\right) = -\sum_{j\colon W_{ij}>0}\left(W_{ij}\widetilde{D}_{ij}-\frac{\partial}{\partial b_{U_{i.}}}R_{ij}\right) \tag{9}$$

$$\frac{\partial}{\partial b_{V_{j.}}}F\left(U,V,b_U,b_V\right) = -\sum_{i\colon W_{ij}>0}\left(W_{ij}\widetilde{D}_{ij}-\frac{\partial}{\partial b_{V_{j.}}}R_{ij}\right), \tag{10}$$

where the coefficients D_{ij} and $\widetilde{D}_{ij}$ are given by

$$D_{ij} = \frac{1}{f\left(X_{ij}\,|\theta_{ij},\vartheta\right)}\cdot\frac{\partial}{\partial\theta}f\left(X_{ij}\mid\theta,\vartheta\right)\Big|_{\theta=\theta_{ij}} \cdot\frac{\partial}{\partial x}\tau\left(x,\mu+b_{U_{i.}}+b_{V_{j.}}\right)\Big|_{x=U_{i.}V_{j.}^T} \tag{11}$$

$$\widetilde{D}_{ij} = \frac{1}{f\left(X_{ij}\,|\theta_{ij},\vartheta\right)}\cdot\frac{\partial}{\partial\theta}f\left(X_{ij}\mid\theta,\vartheta\right)\Big|_{\theta=\theta_{ij}} \cdot\frac{\partial}{\partial y}\tau\left(U_{i.}V_{j.}^T,y\right)\Big|_{y=\mu+b_{U_{i.}}+b_{V_{j.}}}, \tag{12}$$

where θ_{ij} is determined by (2) and $\frac{\partial}{\partial\theta}f\left(X_{ij}\mid\theta,\vartheta\right)\big|_{\theta=\theta_{ij}}$ means that the derivative of $f\left(X_{ij}\mid\theta,\vartheta\right)$ with respect to θ is evaluated at $\theta = \theta_{ij}$. If we use (5) for regularization, we furthermore have $\frac{\partial}{\partial U_{i.}}R_{ij} = 2\lambda U_{i.}$, $\frac{\partial}{\partial V_{j.}}R_{ij} = 2\lambda V_{i.}$, $\frac{\partial}{\partial b_{U_{i.}}}R_{ij} = 2\lambda b_{U_{i.}}$ and $\frac{\partial}{\partial b_{V_{j.}}}R_{ij} = 2\lambda b_{V_{j.}}$.

In order to solve (4) approximately, we use stochastic/mini-batch gradient descent with momentum. Gradient descent is based on the fact that the negative of a gradient of a multivariate differentiable function points in the direction of steepest descent, thus suggesting a good search direction for a minimum. Stochastic/mini-batch gradient descent are extensions of basic gradient descent that exhibit improved performance and are useful for big data sets. In this case, instead of looping over all the training data indices in a single step, one only loops successively in each step $s \in \mathbb{N}$ of the algorithm over one of some randomly selected disjoint subsets B_s whose union is the full training index set $\{(i,j) \mid W_{ij} > 0\}$ and repeats this after one complete pass through the training data. In the so-called "heavy-ball method" [10], an additional momentum term is introduced to increase the speed of convergence. This requires the determination of a suitable learning rate $\varepsilon_s > 0$ and momentum coefficient $\nu_s > 0$. Now, in our case this leads to the following update equations, that are repeated in each step s of the algorithm:[1]

[1]Here, $U^{(s-1)}, V^{(s-1)}, b_U^{(s-1)}, b_V^{(s-1)}$ mean that all the relevant rows/values of step $s-1$ are used. And formally we set $(-1) := (0)$ to make the notation consistent for step 1 of the algorithm.

$$U_{i.}^{(s)} := U_{i.}^{(s-1)} - \varepsilon_s \frac{\partial}{\partial U_{i.}} F(U, V, b_U, b_V)\Big|_{\substack{U=U^{(s-1)}, V=V^{(s-1)},\\ b_U=b_U^{(s-1)}, b_V=b_V^{(s-1)}}} + \nu_s \left(U_{i.}^{(s-1)} - U_{i.}^{(s-2)}\right) \tag{13}$$

$$V_{j.}^{(s)} := V_{j.}^{(s-1)} - \varepsilon_s \frac{\partial}{\partial V_{j.}} F(U, V, b_U, b_V)\Big|_{\substack{U=U^{(s-1)}, V=V^{(s-1)},\\ b_U=b_U^{(s-1)}, b_V=b_V^{(s-1)}}} + \nu_s \left(V_{j.}^{(s-1)} - V_{j.}^{(s-2)}\right) \tag{14}$$

$$b_{U_{i.}}^{(s)} := b_{U_{i.}}^{(s-1)} - \varepsilon_s \frac{\partial}{\partial b_{U_{i.}}} F(U, V, b_U, b_V)\Big|_{\substack{U=U^{(s-1)}, V=V^{(s-1)},\\ b_U=b_U^{(s-1)}, b_V=b_V^{(s-1)}}} + \nu_s \left(b_{U_{i.}}^{(s-1)} - b_{U_{i.}}^{(s-2)}\right) \tag{15}$$

$$b_{V_{j.}}^{(s)} := b_{V_{j.}}^{(s-1)} - \varepsilon_s \frac{\partial}{\partial b_{V_{j.}}} F(U, V, b_U, b_V)\Big|_{\substack{U=U^{(s-1)}, V=V^{(s-1)},\\ b_U=b_U^{(s-1)}, b_V=b_V^{(s-1)}}} + \nu_s \left(b_{V_{j.}}^{(s-1)} - b_{V_{j.}}^{(s-2)}\right) \tag{16}$$

These updates can be computed using the relations (7)–(10). If stochastic/mini-batch gradient descent is used instead of full gradient descent, the summation in (7)–(10) is done over all i respectively j with $(i, j) \in B_s$ instead of all i respectively j with $W_{ij} > 0$. Furthermore, often a constant learning rate ε and momentum coefficient ν can be used, as we do in our experiments. The first important aspect to note about these updates is the fact that they all have the same form for all possible densities f. At this point, the power of automatic differentiation can be utilized. Given selected functions f and τ, the partial derivatives $\frac{\partial}{\partial \theta} f(X_{ij} \mid \theta, \vartheta)\big|_{\theta=\theta_{ij}}$, $\frac{\partial}{\partial x}\tau\left(x, \mu + b_{U_{i.}} + b_{V_{j.}}\right)\big|_{x=U_{i.}V_{j.}^T}$ as well as $\frac{\partial}{\partial y}\tau\left(U_{i.}V_{j.}^T, y\right)\Big|_{y=\mu+b_{U_{i.}}+b_{V_{j.}}}$ (and the partial derivatives of the R_{ij} if they are chosen to be of a different form than (5)) appearing in (11) and (12) can be determined automatically and efficiently by automatic differentiation. Thus, apart from specifying those functions no additional work for derivation is required from the user and the whole optimization can be done by a computer program. Regarding the initial values, we obtained good results by setting all $b_{U_{i.}}^{(0)}$ and $b_{V_{j.}}^{(0)}$ to zero and all vectors $U_{i.}^{(0)}$ and $V_{j.}^{(0)}$ as random perturbations around zero, with a positivity constraint for distributions for which only positive parameter values are allowed.

4.3 Generation of Predictions

After the model parameters have been estimated in the way described in the previous section, one can calculate statistics of interest and generate predictions for unknown ratings X_{ij}. Let us denote by $\widehat{X}_{ij}$ the estimate of X_{ij}. One reasonable choice would be to use the expected value (mean) of the fitted distribution. If we use the same notation for the corresponding random variable, we have

$$\widehat{X}_{ij} = \mathbb{E}[X_{ij}] = \int_{\mathbb{R}} x f(x \mid \theta_{ij}, \vartheta)\, dx \tag{17}$$

in the continuous case. In the discrete case the integral is replaced by a sum. In (17), θ_{ij} is given by (2), whereby for $U_{i.}$, $V_{j.}$, $b_{U_{i.}}$ and $b_{V_{j.}}$ the values are used that were determined by the algorithm in Section 4.2. However, using the expected value is not always the best choice, and the median of the distribution can be a better target in some applications, since it is more robust. While the sample mean is a minimizer of the mean square error, the sample median minimizes the mean absolute error. Thus, it is plausible that using the median of the fitted distribution instead of the mean will yield better results with respect to mean absolute error as a performance measure. And as we discuss below, in our application in which we deal with count data, mean absolute error is more suitable for evaluation purposes than root mean square error. In the continuous case, if we choose $\widehat{X}_{ij}$ to be the median of the distribution, we can instead calculate it by the following relation:

$$\int_{-\infty}^{\widehat{X}_{ij}} f(x \mid \theta_{ij}, \vartheta)\, dx = \frac{1}{2} \tag{18}$$

An analogous result applies for sums in case of a discrete density. Now the benefit of using an appropriate transformation function τ comes into play. For many distributions analytical expressions for the mean and the median dependent on the parameters are available. By a suitable choice of τ we can often use these and calculate the mean or the median easily and efficiently. In particular, let us define τ as a composition $\tau = g \circ l$ with functions $l : (x, y) \mapsto l(x, y)$ and $g : x \mapsto g(x)$. Hereby, the function l describes how the interplay between bias term and user-item interaction is modeled. The most straightforward choice is $l(x, y) := x + y$, meaning that the user-item interaction is modeled as an additive deviation from the bias baseline. A different ansatz could be, for example, $l(x, y) := \exp\left(\frac{x}{c}\right) \cdot y$, for some constant $c > 0$, whereby the user-item interaction is modeled as a multiplicative deviation from the bias term. In case no user-item interaction is present, i.e., $U_{i.}V_{j.}^T = 0$, both choices simplify to the bias term alone. The function g now has the purpose of being the solution function that is determined by solving the equation of the mean or the median, respectively, for θ_{ij} (i.e., the inverse function of the mean function or median function with respect to θ_{ij}). Therefore, based on the analytical expressions for the mean or median, respectively, that depend on θ_{ij}, we seek for g such that $l\left(U_{i.}V_{j.}^T, \mu + b_{U_{i.}} + b_{V_{j.}}\right)$ is identical to the target we are interested in (the mean or median, respectively). As a specific example consider the log-normal distribution, which we use in our experiments. In this case θ_{ij} is the log-scale parameter and dependent on this parameter, the median is given by $\exp(\theta_{ij})$. We choose $l(x, y) := x + y$ and thus we see that the function $g(x) := \log(x)$ satisfies $\exp\left(g\left(l\left(U_{i.}V_{j.}^T, \mu + b_{U_{i.}} + b_{V_{j.}}\right)\right)\right) = U_{i.}V_{j.}^T + \mu + b_{U_{i.}} + b_{V_{j.}}$, so that $U_{i.}V_{j.}^T + \mu + b_{U_{i.}} + b_{V_{j.}}$ indeed is the estimated user-item dependent median, whereby μ is the global median.

The role of g here is similar to the link function in GLMs, but in contrast to the latter it is not limited to natural exponential family distributions and is also allowed to be a map to a different target than the mean (like the median as discussed above).

Now, a big advantage of our approach is that by using computer algebra we can automate the determination of g, too. We only need to specify the expression for the target and by routines for the symbolical solution of equations we can obtain g automatically. In other cases, like when no such a solvable closed-form expression of the target is available, we can instead use numerical integration to approximately solve (17) or (18), respectively, to find $\widehat{X}_{ij}$.

4.4 Flexible Loss Functions as Special Cases

A conceptually different way to approach the problem of generating best recommendations is by using goal-oriented loss functions. In this use case we might not be interested in inferring the precise underlying distribution, but rather impose flexible losses in the form of cost functions that model the problem we have to solve. For example, our objective might arise from a business-oriented perspective. For instance, it might not be as important if an algorithm makes more errors on less valuable customers or products with low profit margin as long as it performs well from a business point of view. There has been related work, such as [1], where profit maximization is addressed, although not in a MF context. A related approach is to directly optimize for ranking functions. The idea behind this is that indicator functions which are hard to deal with can be approximated smoothly like in artificial neural networks, e.g., by a logistic function. There have been particular related approaches in the literature such as [14], in which mean reciprocal rank is addressed.

Problems of this kind can be described by the minimization of a certain distance function, which in turn can be seen as a special case of our framework. Suppose that we construct a distance function $d : \mathbb{R}^2 \to \mathbb{R}_0^+$, $(x, y) \mapsto d(x, y) \geq 0$ which is differentiable with respect to y for every fixed x. Assuming $d(x, x) = 0$ is plausible, but we do not restrict d to be a metric, so it does not have to fulfill symmetry or the triangle inequality. Our goal is then to minimize the deviation $d(X_{ij}, \theta_{ij})$ over all observations (i, j), with θ_{ij} given by (2). This can be described within our probabilistic framework by defining the artificially constructed density as

$$f(x \mid \theta_{ij}) := \frac{1}{C} \exp(-d(x, \theta_{ij})),$$

whereby we require $C := \int_{\mathbb{R}} \exp(-d(x, \theta_{ij}))\, dx < \infty$. So in this case (4) is equivalent to the following problem:

$$\underset{U,V,b_U,b_V}{\arg\min} \sum_{(i,j):\, W_{ij}>0} \left(W_{ij}\, d(X_{ij}, \theta_{ij}) + R_{ij}\right) \qquad (19)$$

Again, while the approaches in the literature are focusing on particular losses and devise a corresponding algorithm manually, the key advantage of our framework is that one can just specify a loss and the optimization algorithm is created automatically.

Discussing and evaluating possibilities of such flexible loss function approaches is beyond the scope of this paper, but will be addressed in future work. Related to this approach are distributions which are closely connected to common losses, such as the Laplace distribution, which will also be investigated in the future.

5. EXPERIMENTAL EVALUATION

5.1 Implementation Details

We have developed our implementation in R whereby we used just-in-time compilation as well as C++ via the Rcpp library for the most computationally intensive part of our code in order to increase performance. For our implementation of automatic differentiation and the symbolic solution of equations, we used the interface to the Python computer algebra library SymPy. A promising alternative to SymPy is the investigation of Theano, which offers advantageous performance. However, its symbolic computing capabilities and its interface to R are still issues that have to be taken into consideration. In the experiments, we made use of several R built-in quantile functions as we describe in Section 5.3.

5.2 Experimental Setup

For the experimental evaluation of our proposed framework, we use a random sample of the commonly available "Echo Nest Taste Profile Subset". This data set consists of play count records of songs by users. In this case, the value X_{ij} represents the number of times user i listened to song j and can be seen as an implicit rating in the form of count data. We can assume that the higher a play count, the more a user likes the corresponding song. We created a sample by initially randomly selecting 20% of the users of this data set and then restricting to play count values between 3 and 300. The purpose of this filtering is to clean the data set from noise and outliers. Low play counts can occur, for instance, because a user is unaware which type of song is behind a link. Very high play counts, on the other hand, are likely the consequence of some automated mechanism like Web crawlers. Additionally, we filtered out all users and songs for which less than 10 play count records exist in order to strengthen the collaborative effect. After applying this preprocessing, we obtained a random sample consisting of 1,557,337 (user, song, play count)-triplets with 68,119 unique users and 34,032 unique songs. A histogram showing the frequency of the different play count values is depicted in Figure 1. The number of users depending on the number of observations (different songs) per user, as well as the number of songs depending on the observations per song are shown in Figure 2.

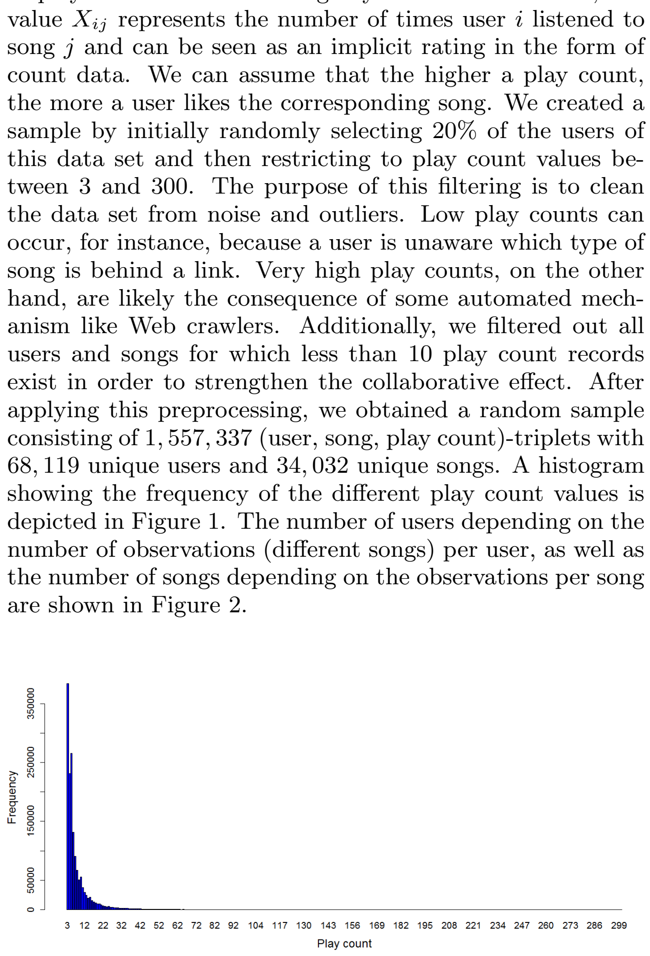

Figure 1: Frequencies of the play count values

We applied our proposed method for different distributions, namely the normal (corresponding to standard PMF), the Poisson, the gamma, the Pareto and the log-normal distribution. For the evaluation, we randomly split the data set into a training part used for model building and a validation

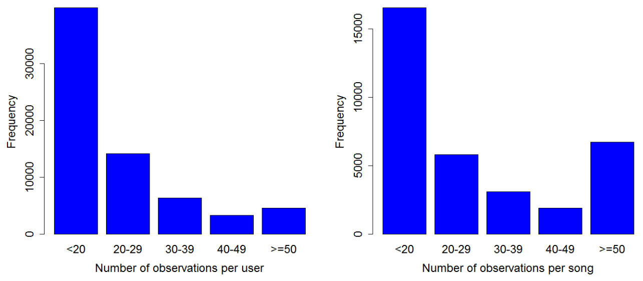

Figure 2: Number of observations per user and song

part used for testing purposes. We used $1,200,000$ randomly chosen (user, song, play count)-triplets for the training part and the remaining $357,337$ for validation. For all distributions we used 4 randomly selected and equally sized batches. We determined the values of the learning rate ε, the momentum coefficient ν, the regularization parameter λ as well as hyperparameters for distributions which provided the best result on the validation set by the commonly used method of manually adapted grid search. This procedure consists of searching through the joint parameter space with refinements in regions in which good values were found. In the same manner, the initial values were selected as described in Section 4.2. We used the best values obtained for the comparison in the next section. We evaluated the performance of our method for all of the distributions with respect to the frequently used mean absolute error (MAE). If the number of all user-item pairs (i,j) in the validation index set $\mathcal{V}$ is denoted by N and the predicted value of every X_{ij} by $\widehat{X}_{ij}$, this is defined as $\text{MAE}\left(\widehat{X}\right) := \frac{1}{N}\sum_{(i,j)\in\mathcal{V}}\left|X_{ij}-\widehat{X}_{ij}\right|$. In our setting, this is a better choice than root mean square error (RMSE). While RMSE is suitable in cases where the rating range is small, such as 1–5 stars, it is not appropriate in the case of count data where the values can vary in several orders of magnitude. Since in this case the squaring in the RMSE definition implies that only small relative changes in the values can result in large errors.

5.3 Experimental Results

Here, we present the results we obtained in our experiments. We have tried various options and tunings for each of the distributions and report the best values we found. Figure 3 shows the performance with respect to MAE on the validation set for each of the distributions dependent on the epoch, whereby epoch refers to the number of full passes through the training data. The minimum MAE values are summarized in Table 1. The value for the mean corresponds to the MAE when predictions for the validation set are made based on the training data mean. The number of features k is 20 for all distributions. In the case of a normal distribution assumption, for which the mean is identical to the median, we used standard PMF as described in Section 3.1. The best parameter settings in this case are $\varepsilon = 1.7\cdot 10^{-4}$, $\nu = 0.4$ and $\lambda = 0.25$, which we obtained by the grid search approach described in Section 5.2. Regarding the Poisson distribution, we found that it is beneficial to fit a shifted version by first subtracting the minimum play count value 3. This means that $f_{Pois}\left(X_{ij}-3 \mid \theta_{ij} = U_{i.}V_{j.}^T + b_{U_{i.}} + b_{V_{j.}}\right)$ (with $\mu = 0$, since θ_{ij} does not correspond to the median in this case) is fitted to determine the parameters and herewith the median is generated by using R's `qpois(0.5,.)` to which 3 is added again for the predictions. In this case, the corresponding best parameter settings are $\varepsilon = 1.7\cdot 10^{-3}$, $\nu = 0.4$ and $\lambda = 0.2$. For the gamma distribution we used the scale parameter as variable parameter θ_{ij} and treated the additional shape parameter κ as a hyperparameter, whereby the same form for θ_{ij} as in the Poisson case worked best. Analogously, we used the R function `qgamma` to predict each of the medians, since as in the case of a Poisson distribution no closed-form expression for the median exists. The respective best parameter settings are $\varepsilon = 6.7\cdot 10^{-3}$, $\nu = 0.4$, $\lambda = 0.01$ and $\kappa = 1$. For the Pareto distribution once again the same ansatz for θ_{ij} worked best, this time representing the shape parameter, while we used a value of 3 for the scale parameter. This choice is plausible, because for this distribution the scale parameter is the minimum possible value and this is 3 for our preprocessed data set. Like for the previous distributions, we used R's corresponding function `qpareto` to calculate the medians. Here, the best parameter choices are $\varepsilon = 3.3\cdot 10^{-4}$, $\nu = 0.4$, $\lambda = 0.15$. Finally, for the log-normal distribution we made use of the fact that in this case there exists a convenient closed-form expression for the median and we used the parameter transformation described in Section 4.3 with which the variable log-scale parameter θ_{ij} is identical to the median. Additionally, we found that using log-scaled biases leads to slight improvements of the MAE for this distribution, although it still performs better than the remaining distributions in the standard form without log-scaled biases. The corresponding parameter settings in this case are $\varepsilon = 1.7\cdot 10^{-3}$, $\nu = 0.4$, $\lambda = 0.01$, whereby the additional shape parameter is 0.5. Also, if alternatively the same ansatz for θ_{ij} as for the other distributions is used and the predictions are analogously generated by the corresponding quantile function `qlnorm` instead of using the transformation function, it still performed better than the remaining distributions (MAE 3.9586). But using this transformation function approach improved the performance.

As we can see from the experimental results in Table 1, all distributions that are more common to model skewed count data show better performance than the standard PMF method which assumes a normal distribution. The different results for the four remaining distributions can be explained by their different tail characteristics. For instance, the Pareto distribution is a heavy-tailed power law distribution. For given (implicit) rating distributions with a higher average rating this would imply that high values still occur with relatively high probability. But according to the histogram in Figure 1, such high count values do not occur often. Therefore it is reasonable that the Pareto distribution performs worse than the other three distributions. In contrast, a log-normal distribution with a bit lower shape parameter (and thus lower variance) models these characteristics better. Of the four non-normal distributions, the Poisson and the gamma distribution are from the natural exponential family, while the Pareto and the log-normal distribution are not. As discussed in Section 2, previous approaches have been limited to natural exponential family distributions with one variable parameter. The fact that the best result is obtained for a log-normal distribution thus demonstrates that it is indeed worthwhile to consider more general distributions. Furthermore, with our framework it is

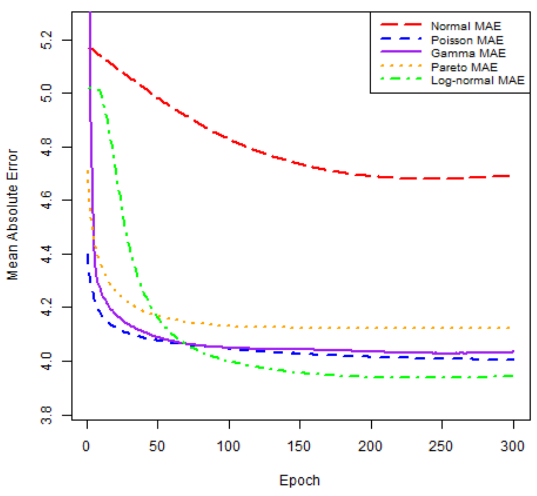

Figure 3: Mean absolute error performance

easy to use modified variants of usual distributions, like predicting the median of a shifted Poisson distribution, which is a different approach from what has been used previously in the literature and indeed turned out to be advantageous.

Table 1: MAE comparison for MF based methods

Method	Minimum MAE achieved
Normal distribution	4.6804
Poisson distribution	4.0038
Gamma distribution	4.0282
Pareto distribution	4.1208
Log-normal distribution	3.9366
Mean	5.1738

6. CONCLUSION

In this paper we have proposed a flexible framework for matrix factorization that allows the use of general distributions. The automatic derivation of the corresponding learning algorithm by means of techniques like automatic differentiation make our framework particularly lucrative in practice. In an application with quantitative implicit feedback we have shown that it is indeed worthwhile to consider broader classes of distributions while different choices can be tested easily and efficiently. Our method opens many directions for future work. This includes the investigation of more complex distributions with multiple parameters, for example in applications with explicit feedback where a normal distribution is likely to be outperformed. This could be the case in online dating, where distributions are often U-shaped, or in settings where the rating distributions are significantly skewed. Furthermore, as we described in Section 4.4, flexible cost functions can be used that allow, for instance, the consideration of business rules. A more extensive evaluation is also needed, e.g., with ranking measures. From a theoretical point of view we plan to extend our framework to allow for constrained optimization, full bayesian inference and Factorization Machines as a building block, whereby side information and temporal dynamics can be incorporated as well as a generalization to tensors. Additionally, we plan to derive theoretical statistical properties of our framework, including the generation of confidence bounds, as well as automated methods to preliminarily determine distributions that are likely to provide good results.

7. REFERENCES

[1] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 121–128. ACM, 2013.

[2] J. Canny. Gap: a factor model for discrete data. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 122–129, 2004.

[3] T. Chen, Z. Zheng, Q. Lu, W. Zhang, and Y. Yu. Feature-based matrix factorization. *arXiv preprint arXiv:1109.2271*, 2011.

[4] Y. Chen, M. Kapralov, J. Canny, and D. Pavlov. Factor modeling for advertisement targeting. In *Advances in Neural Information Processing Systems*, pages 324–332, 2009.

[5] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI'12: Proceedings of the 26th Conference on Artificial Intelligence*, pages 17–23, 2012.

[6] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

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

[8] H. Ma, C. Liu, I. King, and M. R. Lyu. Probabilistic factor models for web site recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 265–274. ACM, 2011.

[9] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[10] B. T. Polyak. *Introduction to optimization*. Optimization Software New York, 1987.

[11] S. Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[12] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[13] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in neural information processing systems 20*, pages 1257–1264, 2008.

[14] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.

[15] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *ECML PKDD*, pages 358–373. Springer, 2008.