Erasmus University Rotterdam

Erasmus School of Economics

Seminar Case Studies in QMarketing FEM21001

Logistic matrix factorization with minimization by majorization: a sparse plus low-rank approach

Abstract

With the rise of the internet and corresponding explosion of available content, efficient and accurate recommender systems are now in high demand. In an effort to help customers find their way, implicit customer feedback, such as clicking behavior, is used in so called collaborative filtering methods to provide customers with recommendations. One way of modelling the clicking behavior of customers is through the probabilistic logistic matrix factorization model. Often, websites deal with a vast amount of products and customers, which can result in significant computational and memory issues. In this paper, we present a new way to optimize and obtain estimates for the logistic matrix factorization model by means of a majorization algorithm which implements an efficient and novel sparse plus low-rank data structure. We demonstrate that the method outperforms dense matrix structures in a simulation setting for large matrices. Additionally, the effectiveness of the logistic matrix factorization model is compared to, and hybridized with, a content-based recommender system that is based on a dataset composed of clicking behavior on offers in promotional mails for a large online tour operator.

Thijs de Bruin	Colin Huliselan	Sanne Lin	Máté Váradi
427882	428287	426416	495556

April 20, 2020

Contents

1 Introduction

With the emergence of the wide-spread usage of the internet, the amount of accessible content and information has rapidly increased. In a commercial context, this translates to the countless products present-day consumers can choose from. For example, whereas the product range in the movie-rental industry used to be confined by the physical capacity and supply chain of stores, services such as Netflix now provide consumers with thousands of movies to choose from, all from the comfort of one's own home. Naturally, this explosion in information comes with challenges regarding the navigation of this sea of content or products. Companies may partly alleviate this problem by filtering information and recommending content in which the consumer is likely to be interested. This benefits both sides of the transaction, as customers are more likely to be satisfied with the service and companies enjoy higher probabilities of purchases. This act of filtering information may be done using 'recommender systems'; techniques that provide suggestions for items to be of use to a user (Resnick and Varian, 1997).

One can distinguish many different such techniques, where differences in large part depend on the type of data that is available[1]. In general, these techniques attempt to use information regarding a user's past interactions (e.g. clicks or ratings) to gain a notion of the user's preferences. Furthermore, these interactions may be either implicit or explicit, where the former refers to an indirect measure of preference, such as a user's browsing history (measured through clicks on a website), and the latter refers to a direct and more informative measure of preference, such as a 1 to 5 rating. In the most restrictive case, all we have are these interactions, without information about the items or users.

In this application we look at holiday offers sent through e-mail by Sunweb, on which recipients can click for more information. Often, implicit feedback includes data on the inferred positive interest of the customer, but cannot be used to infer negative interest (Hu et al., 2008). For instance, when a user watches a certain TV show, they are presumably interested in it, but not having seen it does not necessarily imply dislike for the show. In our dataset, we can use a click on an offer to infer that the customer has observed the offer *and* is interested in it. However, our dataset differs from many other cases with implicit feedback, since it also distinguishes between offers that were not seen by the customer and offers that were seen but not clicked on, implying disinterest in the offer. Therefore, although implicit in nature, our data is more similar to a sparse binary dataset.

A common category of recommendation system methods is collaborative filtering, which is used to predict which items a user may be interested in, based on the interests of other users that have similar preferences. These methods often rely on matrix factorization techniques, which may result in significant computational and memory issues. We derive, for the first time, a solution for optimizing the 'logistic matrix factorization' (Johnson, 2014) problem for binary data using the sparse plus low-rank data structure as proposed in Mazumder et al. (2010), a data structure which has been shown to be very efficient in the context of large-scale matrix completion problems such as the Netflix prize problem (Bennett et al.,

[1] See Burke (2002) for a useful overview of the general methods and ways to combine them.

2007). Besides this new method, we apply the logistic matrix factorization method based on Johnson (2014), and build on it by implementing an iterated stochastic gradient ascent (ISGA) procedure. We compare and contrast the efficiency of the two matrix factorization methods.

Content-based techniques form another subset within recommender systems, and use user and/or item attributes to make recommendations. We implement a simple content-based model that estimates a logistic regression model for a subset of users. The benefit of this model in this application is that it is able use the available characteristics of the holiday offers. In our final model, we use both the click behavior of users and the holiday offer characteristics to gain a notion of user preferences by combining collaborative filtering and content-based techniques in a hybrid method. The predictions can be interpreted as the probability that a user will click on a certain offer. This leads to the research question of this paper: *To what extent can we accurately and efficiently predict the clicking behavior of customers*? Here, we state the following hypotheses. First, we expect that we can beat simple baseline cases by specifying a bi-additive model which has the ability to encapsulate the baseline, but adds additional components to it. Second, we expect that through simple calculations of the majorization function we achieve relatively short iteration times. Third, we expect that the sparse plus low-rank addition further increases the iteration speed and thereby the overall efficiency. Last, as for the comparison between the majorization method and ISGA, we expect that the methods converge to roughly similar optima, if hyperparameters are set appropriately.

Using simulated data we show that the sparse plus low-rank addition to the majorization framework provides considerable speed increases, being up to 4 times faster and providing significant benefits especially for large and highly sparse datasets. As for our final results, we find that a hybridization method that uses a weighted average to combine predictions coming from logistic matrix factorization with majorization and the content-based method performs best and achieves RMSE values that are below the best baseline method that we define. This hybrid method also performs best in classification metrics, such as the area under the curve of the Receiver Operating Characteristic curve and the Precision-Recall plot.

In the next section, we give a short overview of the relevant literature available on recommender systems. In Section 3, we give a detailed description of the datasets used for both the collaborative filtering and the content-based methods. Thereafter, in Section 4, we continue with a specification and derivation of the used methods and describe a series of implementation details that speed up our algorithms. In Section 5, we compare the performance of all individual models first, after which we present results for our best performing (hybrid) model. Lastly, in Section 6 we summarize and discuss our overall findings.

2 Literature

The majority of research dealing with recommender systems is centered around explicit feedback. In this section we explore the techniques that are suitable for implicit feedback data. Two common approaches

for recommendation systems are collaborative and content-based filtering. Content-based filtering (CBF) treats recommendations as a user-specific classification problem and learns to recommend items that are similar to the ones that the user liked in the past. Here, item features are used to compare their similarities. The advantage of CBF is that it produces recommendations that are explainable. Additionally, and opposed to other methods, CBF can also be used to recommend new or unpopular items based on similar characteristics. One thing to note, however, is that recommendation quality is limited by the selected features of the recommended items (Çano and Morisio, 2017). This is one possible disadvantage of content-based methods, along with the fact that CBF only recommends items that fit a user's observed taste profile, which may be too restrictive.

Collaborative filtering (CF) is the most popular and most widely implemented recommendation system technique. CF arises from the rationale that if, in the past, a certain user agreed with other users, then recommendations coming from these similar users should be of relevance. Many CF algorithms belong to one of two types: neighborhood methods and latent factor models (Koren et al., 2009). In neighborhood methods, the neighborhood of a given user or item is computed based on some similarity measure and recommendations are generated using this neighborhood. Techniques in which user neighborhoods are computed are referred to as user-based recommendation systems (Ricci et al., 2015). They estimate unknown ratings based on past ratings of similar users. Item-oriented or item-to-item methods estimate ratings using past ratings made by the given user on similar items (Hu et al., 2008). Item-to-item methods are generally favorable to user-to-user methods due to their better scalability and higher accuracy (Linden et al., 2003). However, according to Hu et al. (2008) in the case of implicit feedback, item-oriented methods lack the required flexibility to distinguish between user preferences.

Latent factor models try to explain ratings by a characterization of both items and users on some latent factors inferred from the rating patterns. Some of the most successful latent factor CF methods for implicit feedback are based on matrix factorization (Mnih and Teh, 2012). Matrix factorization methods seek to find a low-rank approximation to the sparse matrix of ratings by decomposing the matrix of observations into two low-rank matrices and minimizing a loss function on the known ratings. One such method created for implicit feedback is introduced in Hu et al. (2008). In this study, implicit user observations are transformed into preferences and confidence levels. Pan et al. (2008) propose to overcome problems arising from the sparsity and uncertainty of implicit data by applying negative example sampling (drawing observations from the missing data and labeling these as negative examples, i.e. items that the user is not interested in) in a weighted matrix factorization method, where they employ different weighing schemes of the error terms in the objective function. Johnson (2014) proposes a probabilistic matrix factorization model, where the probability of a user interacting with a certain item is modelled as a logistic function parameterized by the interaction of user and item latent factor vectors as well as user and item biases.

Matrix factorization methods can be optimized using different learning algorithms. One such algorithm is gradient descent, which optimizes the loss function iteratively by updating the parameters in each iteration with a value proportional to the gradient (Ruder, 2016). However, a limitation of this algorithm

is that a calculation of the gradient on the entire data can be computationally exhaustive for large datasets. Johnson (2014) uses gradient ascent to optimize an objective function and suggests selecting the gradient step sizes using AdaGrad (Duchi et al., 2011) in order to decrease the number of iterations necessary for convergence. An alternative to gradient descent is stochastic gradient descent (SGD), which uses the gradient of a random subset of the data as an approximation of the true gradient. In recent years, extensions to SGD have been proposed that aim to improve the prediction performance and/or scalability of the algorithm (Gemulla et al., 2011; Li et al., 2014; Ahn et al., 2015; Li and Ou, 2016).

A popular alternative to (stochastic) gradient descent in the context of matrix factorization is alternating least squares (ALS) (Hu et al., 2008). This method alternates between fixing one of the low-rank matrices, applying a least squares regression to find an approximation of the other low-rank matrix, and repeating this process for the other low-rank matrix. Here too, extensions have been proposed that are aimed at decreasing the computational burden of the algorithm. He et al. (2016) propose a fast element-wise alternating least squares (eALS) algorithm, an extension to the ALS algorithm and the generic element-wise ALS. This algorithm reduces the time complexity through memoization, i.e. by pre-computing certain large calculations that will be used repeatedly and storing them for later use. This prevents unnecessary recalculation of computationally exhaustive expressions. Hastie et al. (2015) introduce a new algorithm for rank-restricted singular value decomposition (SVD) with fast alternating least squares. They use the sparse plus low-rank structure of the input data to efficiently calculate matrix multiplications and significantly speed up computation time.

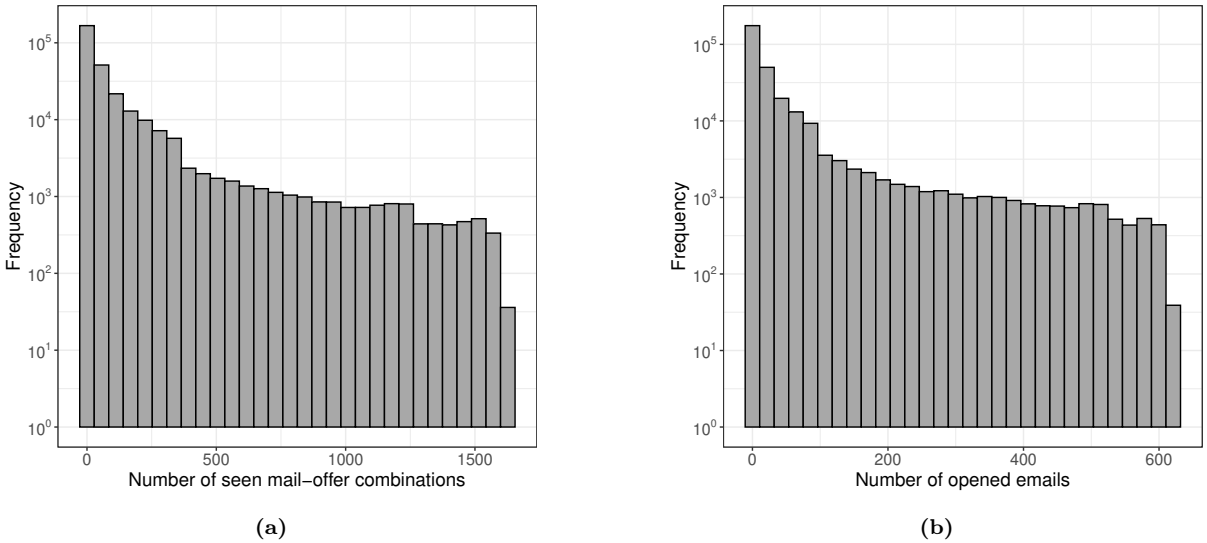
A less common approach to matrix factorization is through the minimization by majorization (MM) approach. Instead of minimizing the loss function directly, which may take on a complex expression, MM minimizes a series of majorizing functions, which are less complex to minimize, yet still guarantee a decrease of the loss function in each iteration (de Leeuw and Lange, 2009). MM has been applied in non-negative matrix factorization (NMF) (see Févotte (2011) and Lin et al. (2017)), where the low-rank matrices resulting from the decomposition have the additional constraint that each element must be non-negative. Furthermore, Groenen et al. (2003) propose a logistic majorization algorithm, used to minimize the negative log likelihood of logistic functions, where the logistic function is parameterized by a bi-additive interaction effect of users and items, similar to Johnson (2014). Using quadratic majorizers, the majorization function for the negative log likelihood can be expressed as a weighted least squares problem.

Finally, different methods can be combined into one model, incorporating more and/or different kinds of information and potentially avoiding some of the limitations of the individual methods. A systematic overview of hybrid systems can be found in Çano and Morisio (2017). A common combination is that of content-based and collaborative filtering methods. Depending on the hybridization approach, different types of hybrid systems can be found. Some of the approaches require building separate recommender systems first, then combining the outputs of these systems by weighting or ranking them, or switching between them. For instance, using a weighted approach, the resulting scores given by the separate

recommender systems can be combined using a linear formula (Miranda et al., 1999; Mobasher et al., 2003). Cascade systems use a hierarchy of recommendation systems, where predictions are based on the primary model, and alternative or secondary systems are used only to break ties. A switching hybrid system switches between recommendation systems using some switching criterion; either a comparable measure of confidence of the individual algorithms or an alternative measure taking into account the strengths of the methods. Ghazanfar and Prugel-Bennett (2010) combine item-based CF with both Naive Bayes and Support Vector Machine (SVM) classifiers by switching between the individual methods depending on the difference between their predictions. A different hybridization approach consists of combining features derived from different knowledge sources. For example, in feature augmentation type hybrids, a recommendation technique is used to compute a feature, which is then part of the input into the next technique (Burke, 2002).

3 Data

In this research, we look at a mailing dataset of large online tour operator Sunweb. The dataset contains information on e-mails that were sent to approximately 300.000 users, spanning a period of 20 months in which users often received e-mails on a daily basis. These e-mails contained one or multiple holiday offers. For each mail-offer combination, (implicit) feedback is available in the form of clicks. Hence, for a sequence of e-mails per user, we know what holiday offers this user did or did not click on. It is important to note that we only have information on e-mails that were opened, and that we formulate predictions for mail-offer combinations. In other words, all mail-offers are considered distinct, even if it is in fact the same offer shown in different emails. Any reference to offers in the following sections refers to the mail-offer combinations.



(a) **(b)**

Figure 1: Frequency histogram on a logarithmic scale of the number of seen offers (left) and the number of opened emails (right).

In total, the dataset provides 12.031.803 opened e-mails, which on average contain 2.67 different offers, with 10 being the maximum number of offers per e-mail. This yields 32.173.022 data points in total, spread over 297.572 users. This means our feedback is highly sparse, as at most 5.07% of the user/offer combinations have been observed. When we look at the total number of seen offers per user (across all e-mails opened by the user), we see large disparities, with a mean of 108, median of 19, standard deviation of 228 and a minimum and maximum of 1 and 1628, respectively. Panel (a) of Figure 1 shows the frequency histogram of the number of seen offers (on a logarithmic scale). In most cases, a relatively small number of offers was evaluated. When we look at the number of opened e-mails, we again see that most individuals opened relatively few e-mails. We find that the mean number of opened e-mails is 40, with a median of 6, standard deviation of 92, and a minimum and maximum of 1 and 622, respectively. Panel (b) of Figure 1 shows the frequency histogram of the number of opened mails (on a logarithmic scale). The spike at the start of the histogram reveals that many users opened few mails. In fact, approximately 25% of individuals only opened one mail.

As noted before, our variable of interest concerns implicit feedback and tracks whether a user clicked on a certain offer or not. The average click rate for the entire sample is 2.19%. The average number of clicks per user is 2.37. However, this is caused by some users that click on many offers (the maximum being 848), whereas the majority (72.34%) of users have not clicked on any of the offers (whilst all of them having opened at least one mail). Of all users, 9.48% clicked on a single offer only and 18.18% clicked on more than one.

Table 1: Overview of offer characteristics.

OFFER_POSITION	Integer denoting the position of the offer in the e-mail.
HALF_WIDTH	Whether the offer visualization in the e-mail takes only half of the e-mail width.
DEPARTURE_MONTH	The month of departure.
MAIL_MONTH	The month of when the e-mail was sent.
COUNTRY_NAME	Destination country.
REVIEW_RATING	Tour operator user rating of the accomodation.
STAR_RATING	Star rating of the accommodation.
ROOM_OCCUPANCY	Number of adults for which the offer holds.
CHILDREN	Whether there are beds for children in the room.
PRICE_ORIGINAL	Original price of the offer.
DISCOUNT	Discount given from the original price.
MEAL_PLAN	Meal plan of the hotel, e.g. all inclusive or breakfast only.

Besides click behavior of each user, the dataset provides characteristics of the holiday offers. For example, specifics regarding the star rating, room setup and location of the offer are known. Table 1 provides a subset of the variables used in this research, including a short description. In total, the dataset contains 2,187 unique offers.

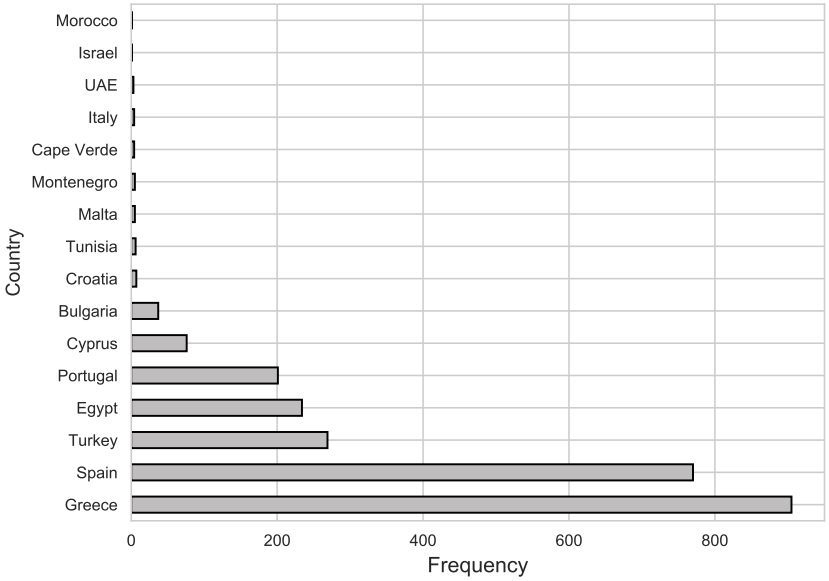


Figure 2: Frequency of countries among the vacation offers.

The average price of the vacations is €684 with a standard deviation of €219.50. The offers are almost always discounted: 97% of the offers in the e-mails feature a price discount, which on average amounts to 27% of the original price. The average star rating of the hotels is 3.84 and the average review rating is 8.11. The `MEAL_PLAN` variable contains information on the catering service of the hotel, ranging from no food included to half-pension to ultra all-inclusive. All-inclusive is the most common plan. From Figure 2, we can see that Greece and Spain are the most frequently appearing destinations. Lastly, Figure 3 shows that while roughly the same number of e-mails are sent out each month, many of the trips presented in the emails depart in January. These trips by far outnumber the trips departing in other months, with the second most common departure month (July) being less than half as common.

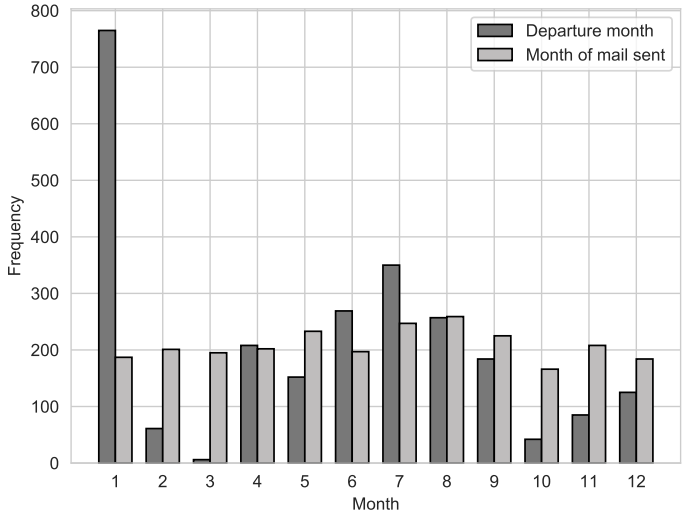


Figure 3: The number of departures and sent mails per month.

4 Methodology

In this section, we describe all methods used to predict the probability that a user will click on a given offer. Two models are introduced: a collaborative filtering model coined logistic matrix factorization and a content-based model using logistic regression. Here, estimates for the former model are acquired through two different optimization methods: a (stochastic) gradient ascent method and a method using majorization of the objective function.

We have data on n_u users that have seen some subset of the n_i offers. We load click data into the $n_u \times n_i$ sparse matrix $\boldsymbol{Y}$ in the following manner: $y_{ui} = 1$ if the user u has clicked on offer i, $y_{ui} = 0$ if the user u has seen offer i but has not clicked on the offer. Since y_{ui} is binary, it is common practice to assume that the probability π_{ui} of a response is modeled by a binomial distribution. The goal of our models is to give the best possible approximation of the mean of this distribution μ_{ui}, using the root mean squared error (RMSE) as the primary performance measure, given by

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{G}|} \sum_{(u,i) \in \mathcal{G}} (y_{ui} - \hat{\mu}_{ui})^2}. \tag{1}$$

Here, $\mathcal{G}$ is generally a set of observations unseen by the models (referred to as the test set from now on). The set of observations used for training the models is referred to as the training set, or $\mathcal{A}$. It may happen that some users or offers that appear in the test set do not appear in the training set. If a user of the test set does not appear in the training set, neither content-based nor collaborative filtering are able to produce probabilities. When an offer does not appear in the training set, it causes a problem with the CF methods only. In these cases, we cannot model the probability μ_{ui} and resort to predicting the global

average click rate (see Section 4.1). Furthermore, if a user has never clicked on any offer in the past, we predict a probability of zero for any new offers they may encounter.

4.1 Baseline models

In order to compare the outcomes of our methods, we specify four baseline methods to predict the probability of a user clicking on a certain offer.

1. **Majority rule**: the mode of the observed clicks, i.e. zero.
2. **Global average click rate**: $\frac{\sum_{(u,i)\in\mathcal{A}} y_{ui}}{|\mathcal{A}|}$, where $\mathcal{A}$ is the set of (u,i) pairs for which y_{ui} is observed.
3. **Average click rate per user**: $\frac{\sum_{i\in\mathcal{A}_u} y_{ui}}{|\mathcal{A}_u|}$, where $\mathcal{A}_u$ is the set of offers seen by user u.
4. **Average click rate per offer**: $\frac{\sum_{u\in\mathcal{A}_i} y_{ui}}{|\mathcal{A}_i|}$, where $\mathcal{A}_i$ is the set of users who have seen offer i.

4.2 Logistic matrix factorization

The model implemented in this paper is largely based on the logistic bi-additive model as proposed in Groenen et al. (2003). Here we model μ_{ui} using a logistic function

$$\mu_{ui} = \frac{e^{\gamma_{ui}}}{1+e^{\gamma_{ui}}}, \tag{2}$$

where the elements γ_{ui} are captured in an $n_u \times n_i$ matrix $\mathbf{\Gamma}$. $\mathbf{\Gamma}$ is described by the bi-additive model and includes the following effects:

- the main effect for the users in the $n_u \times 1$ vector $\boldsymbol{\alpha}$,
- the main effect for the offers in the $n_i \times 1$ vector $\boldsymbol{\beta}$, and
- the bi-additive interaction effect between rows and columns estimated by the rank f decomposition $\boldsymbol{CD}'$ with $\boldsymbol{C} = (\boldsymbol{c}'_1, \boldsymbol{c}'_2, \ldots, \boldsymbol{c}'_{n_u})'$ of size $n_u \times f$ and $\boldsymbol{D} = (\boldsymbol{d}'_1, \boldsymbol{d}'_2, \ldots, \boldsymbol{d}'_{n_i})'$ of size $n_i \times f$.

As such, $\mathbf{\Gamma}$ can be written as

$$\mathbf{\Gamma} = \boldsymbol{\alpha}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}' + \boldsymbol{CD}', \tag{3}$$

where $\boldsymbol{\beta}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are column-centered to maintain unique identification.

As the data used in this paper is very sparse, we introduce the weighting matrix $\boldsymbol{W}$ as an addition to the model proposed in Groenen et al. (2003), in which we define $w_{ui} = 1$ if user u observed offer i and $w_{ui} = 0$ if the offer was not observed. Using this $\boldsymbol{W}$ the likelihood function becomes

$$\mathcal{L}_{prior}(\boldsymbol{Y}|\mathbf{\Gamma}) = \prod_{ui} \mu_{ui}^{y_{ui}w_{ui}} (1-\mu_{ui})^{(1-y_{ui})w_{ui}}. \tag{4}$$

To find a feasible solution we need to restrict γ_{ui}, as otherwise $\gamma_{ui} = \infty$ for $y_{ui} = 1$ and $\gamma_{ui} = -\infty$ for $y_{ui} = 0$ is a trivial and optimal solution. We borrow the zero-mean spherical Gaussian priors on user and offer latent factor vectors used in Johnson (2014). The prior used for $\boldsymbol{C}$ is

$$p(\boldsymbol{C}|\sigma^2) = \prod_u \mathcal{N}(\boldsymbol{c}_u|0, \sigma^2 I) = \prod_u (2\pi\sigma^2)^{-\frac{n_i}{2}} e^{-\frac{||\boldsymbol{c}_u||^2}{2\sigma^2}}, \tag{5}$$

where $\|.\|$ is the Euclidean norm. A similar prior can be set for $\boldsymbol{D}$. Additionally, we set diffuse priors for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, i.e. $p(\boldsymbol{\alpha}) \propto 1$ and $p(\boldsymbol{\beta}) \propto 1$. With these priors in place, the posterior log likelihood reads

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) &= \sum_{ui} \left[y_{ui} w_{ui} \log(\mu_{ui}) + (1 - y_{ui}) w_{ui} \log(1 - \mu_{ui}) \right] - \frac{\lambda}{2} \sum_u \|\boldsymbol{c}_u\|^2 - \frac{\lambda}{2} \sum_i \|\boldsymbol{d}_i\|^2 + \nu \\ &= \sum_{u,i} \left[y_{ui} w_{ui} (\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i) - w_{ui} \log(1 + e^{\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i}) \right] - \frac{\lambda}{2} \|\boldsymbol{C}\|_F^2 - \frac{\lambda}{2} \|\boldsymbol{D}\|_F^2 + \nu, \end{aligned} \tag{6}$$

where $\|.\|_F$ is the Frobenius norm, $\lambda = \frac{1}{\sigma^2}$, ν is a constant term. The posterior log likelihood needs to be maximised over $\boldsymbol{\Gamma}$ and contains a penalty term for both $\boldsymbol{C}$ and $\boldsymbol{D}$. This penalty is equivalent to the nuclear norm penalty, defined as the sum of singular values of $\boldsymbol{C}\boldsymbol{D}'$ (Mazumder et al., 2010).

Optimizing this problem in an iterative manner potentially requires calculations for a large number of parameters in each iteration. The most obvious method, (full) gradient ascent, is often infeasible given the size of many recommendation system problems, including ours. One way to avoid this problem is by only considering a smaller subset of the data in each iteration, and calculating the gradients based on this subset. This is the first approach we take, described in Section 4.2.1. Alternatively, in Section 4.2.2 we present a method that can provide updates using the full dataset, and does so through iterative minimizations of repeated majorizations of the negative log posterior likelihood.

4.2.1 Logistic matrix factorization with iterated stochastic gradient ascent

Maximizing the objective function defined in (6) can be achieved by performing an alternating mini-batch gradient ascent procedure as outlined in Johnson (2014). In each iteration, we first fix the user vectors and biases and take a step towards the gradient of the item vectors and biases. Then, we fix the item vectors and biases and perform the same step for user vectors and biases. Partial derivatives for user vectors and biases are given by

$$\frac{\partial}{\partial \boldsymbol{c}_u} = \sum_i \left(y_{ui} - \frac{e^{\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i}}{1 + e^{\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i}} \right) w_{ui} \boldsymbol{d}_i - \lambda \boldsymbol{c}_u, \tag{7}$$

$$\frac{\partial}{\partial \alpha_u} = \sum_i \left(y_{ui} - \frac{e^{\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i}}{1 + e^{\alpha_u + \beta_i + \boldsymbol{c}_u' \boldsymbol{d}_i}} \right) w_{ui}. \tag{8}$$

As an addition to Johnson (2014), to accelerate running times, we approximate the gradient by calculating partial derivatives on a proportion B of the data in each iteration. When we update user vectors and

biases, we fix item vectors and biases and take samples $\widetilde{\boldsymbol{Y}}$ and $\widetilde{\boldsymbol{W}}$, both of size $n_u \times m_i$ of the original matrices $\boldsymbol{Y}$ and $\boldsymbol{W}$, where m_i equals $B \times n_i$, with B set to 10%. Accordingly, we take a subset of item vectors $\widetilde{\boldsymbol{D}}$ of size $m_i \times f$. Using this approximated gradient $\boldsymbol{g}_u^{(t)}$ at each iteration t the parameters are then updated according to AdaGrad (Johnson, 2014):

$$\boldsymbol{c}_u^{(t)} = \boldsymbol{c}_u^{(t-1)} + \frac{\delta \boldsymbol{g}_u^{(t-1)}}{\sqrt{\sum_{s=1}^{t-1} (\boldsymbol{g}_u^{(s)})^2}}. \tag{9}$$

4.2.2 Logistic matrix factorization with majorization

A second way we can obtain estimates for this model is by applying minimization by majorization (MM). MM is a technique that is used to simplify complex optimization problems, by replacing the objective function with a majorizing function for which the minimum can be found more easily (de Leeuw and Lange, 2009). The definition of a majorizing function is as follows. A function $g : \mathcal{D}_g \to \mathcal{R}_g$ is considered to be a majorizing function for an objective function $f : \mathcal{D}_f \to \mathcal{R}_f$ at the point x_0 if

$$g(x_0; x_0) = f(x_0), \tag{10}$$

$$g(x; x_0) \geq f(x), \qquad \forall x \in \mathcal{D}_f. \tag{11}$$

Additionally, if f and g are both twice differentiable, the following properties also apply:

$$g'(x_0; x_0) = f'(x_0), \tag{12}$$

$$g''(x_0; x_0) \geq f''(x_0). \tag{13}$$

Optimization (i.e. minimization) of f using MM is achieved iteratively by finding a majorization function g at the current best solution $x^{(k)}$ and updating the solution with $x^{(k+1)}$, where $x^{(k+1)}$ minimizes g. Then, a new majorization function is created at the updated solution after which this is again minimized. This process is iterated until the solution has converged. Following from the properties of majorizing functions shown in (10) and (11), this process guarantees descent of the objective function, as illustrated by the 'sandwich inequality'

$$f(x^{(k+1)}) \leq g(x^{(k+1)}; x^{(k)}) \leq g(x^{(k)}; x^{(k)}) = f(x^{(k)}), \tag{14}$$

which demonstrates that when g majorizes f at $x^{(k)}$, the x that minimizes g yields an objective value for f that is at least as small as the previous objective value.

In this paper, we implement logistic majorization based on Groenen et al. (2003). Instead of maximizing the log likelihood, this algorithm minimizes the negative log likelihood using convex quadratic majorizers.

Rewriting the log likelihood function in equation (6) gives us the following negative log likelihood

$$-\log \mathcal{L}(\mathbf{\Gamma}|\boldsymbol{Y}) = \sum_{ui} \left[y_{ui} w_{ui} \log(1 + e^{-\gamma_{ui}}) + (1 - y_{ui}) w_{ui} (\gamma_{ui} + \log(1 + e^{-\gamma_{ui}})) \right] + \frac{\lambda}{2} \|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2} \|\boldsymbol{D}\|_F^2 + \nu, \tag{15}$$

where we want to majorize the terms

$$\begin{aligned} f_1(\gamma_{ui}) &= \log(1 + e^{-\gamma_{ui}}) && \text{if} \quad y_{ui} = 1 \ \wedge \ w_{ui} = 1 \\ f_2(\gamma_{ui}) &= \gamma_{ui} + \log(1 + e^{-\gamma_{ui}}) && \text{if} \quad y_{ui} = 0 \ \wedge \ w_{ui} = 1 \\ f_3(\gamma_{ui}) &= 0 && \text{if} \quad w_{ui} = 0, \end{aligned}$$

using quadratic majorizating functions in the format of

$$g(\gamma_{ui}; \gamma_{ui}^{(0)}) = a_{ui} \gamma_{ui}^2 - 2 b_{ui} \gamma_{ui} + c_{ui}, \tag{16}$$

with $\gamma_{ui}^{(0)}$ indicating the current value of γ_{ui}. We use the properties from (10) - (13) to find parameters a_{ui}, b_{ui}, and c_{ui} as follows. First, we find the second derivatives of the separate terms f_k for all $k \in \{1, 2, 3\}$ and $g(\gamma_{ui}; \gamma_{ui}^{(0)})$, given by

$$\begin{aligned} f_1''(\gamma_{ui}) = f_2''(\gamma_{ui}) &= \frac{e^{\gamma_{ui}}}{(1 + e^{\gamma_{ui}})^2}, \\ f_3''(\gamma_{ui}) &= 0, \\ g''(\gamma_{ui}; \gamma_{ui}^{(0)}) &= 2 a_{ui}. \end{aligned} \tag{17}$$

The second derivative of f_1 and f_2 has a maximum of $\frac{1}{4}$. Therefore, we set a_{ui} to $\frac{1}{8}$ such that the second derivative of $g(\gamma_{ui}; \gamma_{ui}^{(0)})$ is always larger or equal to the second derivatives of f_k, satisfying property (13). Next, we use the properties in (10) and (12) to derive the conditions for b_{ui} and c_{ui}, illustrated below for f_1,

$$f_1(\gamma_{ui}^{(0)}) = g(\gamma_{ui}^{(0)}; \gamma_{ui}^{(0)}) = \frac{1}{8} (\gamma_{ui}^{(0)})^2 - 2 b_{ui} \gamma_{ui}^{(0)} + c_{ui} \tag{18}$$

$$f_1'(\gamma_{ui}^{(0)}) = g'(\gamma_{ui}^{(0)}; \gamma_{ui}^{(0)}) = \frac{1}{4} \gamma_{ui}^{(0)} - 2 b_{ui}. \tag{19}$$

Solving these equations for b_{ui} and c_{ui}, we get

$$b_{ui} = \frac{1}{8} \gamma_{ui}^{(0)} - \frac{1}{2} f_1'(\gamma_{ui}^{(0)}), \tag{20}$$

$$c_{ui} = f_1(\gamma_{ui}^{(0)}) + \frac{1}{8} (\gamma_{ui}^{(0)})^2 - f_1'(\gamma_{ui}^{(0)}) \gamma_{ui}^{(0)}. \tag{21}$$

Extending the above for f_2 and f_3, and keeping in mind that $f_3(\gamma_{ui}) = f_3'(\gamma_{ui}) = 0$, the parameters of

$g(\gamma_{ui};\gamma_{ui}^{(0)})$ can be defined as

$$
\begin{aligned}
a_{ui} &= \frac{1}{8}, \\
b_{ui} &= \begin{cases} \frac{1}{8}\gamma_{ui}^{(0)} - \frac{1}{2}f_1'(\gamma_{ui}^{(0)}) & \text{if } y_{ui}=1 \ \wedge \ w_{ui}=1, \\ \frac{1}{8}\gamma_{ui}^{(0)} - \frac{1}{2}f_2'(\gamma_{ui}^{(0)}) & \text{if } y_{ui}=0 \ \wedge \ w_{ui}=1, \\ \frac{1}{8}\gamma_{ui}^{(0)} & \text{if } w_{ui}=0, \end{cases} \\
c_{ui} &= \begin{cases} f_1(\gamma_{ui}^{(0)}) + \frac{1}{8}(\gamma_{ui}^{(0)})^2 - f_1'(\gamma_{ui}^{(0)})\gamma_{ui}^{(0)} & \text{if } y_{ui}=1 \ \wedge \ w_{ui}=1, \\ f_2(\gamma_{ui}^{(0)}) + \frac{1}{8}(\gamma_{ui}^{(0)})^2 - f_2'(\gamma_{ui}^{(0)})\gamma_{ui}^{(0)} & \text{if } y_{ui}=0 \ \wedge \ w_{ui}=1, \\ \frac{1}{8}(\gamma_{ui}^{(0)})^2 & \text{if } w_{ui}=0, \end{cases}
\end{aligned}
\tag{22}
$$

where the derivatives of f_1 and f_2 are defined as

$$f_1'(\gamma_{ui}) = -\frac{1}{1+e^{\gamma_{ui}}}, \tag{23}$$

$$f_2'(\gamma_{ui}) = \frac{1}{1+e^{-\gamma_{ui}}}. \tag{24}$$

With (22) we can define the majorization of (15) as

$$-\log\mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) \leq \sum_{ui}\left(\frac{1}{8}\gamma_{ui}^2 - 2b_{ui}\gamma_{ui} + c_{ui}\right) + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c, \tag{25}$$

which can be rewritten as the weighted least-squares problem

$$
\begin{aligned}
-\log\mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) &\leq \frac{1}{8}\sum_{ui}(h_{ui}-\gamma_{ui})^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c \\
&= \frac{1}{8}\|\boldsymbol{H}-\boldsymbol{\Gamma}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c,
\end{aligned}
\tag{26}
$$

where

$$
\begin{aligned}
h_{ui} &= 8b_{ui}, \\
c &= \sum_{ui}\left(c_{ui} - 8b_{ui}^2\right).
\end{aligned}
\tag{27}
$$

Notice we do not have to calculate c_{ui} as the only occurrence of c_{ui} is in the constant of (26), which need not be optimized. Using the definitions of b_{ui} in (22), we are able to formulate $\boldsymbol{H}$ using a sparse plus low-rank structure and use the corresponding methods discussed in Mazumder et al. (2010) and Hastie et al. (2015). We can write

$$
\begin{aligned}
\boldsymbol{H} &= \boldsymbol{Z}_{\text{sparse}} + \boldsymbol{\Gamma}^{(\mathbf{0})} \\
&= \boldsymbol{Z}_{\text{sparse}} + \boldsymbol{\alpha}^{(0)}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}^{(0)\prime} + \boldsymbol{C}^{(0)}\boldsymbol{D}^{(0)\prime} \\
&= \text{sparse} + \text{low-rank},
\end{aligned}
\tag{28}
$$

where z_{ui}, the elements of $\boldsymbol{Z}_{\text{sparse}}$, equal

$$z_{ui} = \begin{cases} -4f_1'(\gamma_{ui}^{(0)}) & \text{if } y_{ui} = 1 \ \wedge \ w_{ui} = 1, \\ -4f_2'(\gamma_{ui}^{(0)}) & \text{if } y_{ui} = 0 \ \wedge \ w_{ui} = 1, \\ 0 & \text{if } w_{ui} = 0. \end{cases} \tag{29}$$

By pre- and post-multiplying $\boldsymbol{H}$ with centering matrices $\boldsymbol{J}_{n_u}$ and $\boldsymbol{J}_{n_i}$, we get

$$\begin{aligned} \boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i} &= (\boldsymbol{I}_{n_u} - n_u^{-1}\mathbf{1}\mathbf{1}')\boldsymbol{H}(\boldsymbol{I}_{n_i} - n_i^{-1}\mathbf{1}\mathbf{1}') \\ &= \boldsymbol{H} - n_u^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H} - n_i^{-1}\boldsymbol{H}\mathbf{1}\mathbf{1}' + (n_u n_i)^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H}\mathbf{1}\mathbf{1}' \\ &= \boldsymbol{H} - n_i^{-1}\boldsymbol{H}\mathbf{1}\mathbf{1}' - n_u^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H}\boldsymbol{J}_{n_i}, \end{aligned} \tag{30}$$

which implies

$$\boldsymbol{H} = \boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i} + n_i^{-1}\boldsymbol{H}\mathbf{1}\mathbf{1}' + n_u^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H}\boldsymbol{J}_{n_i}. \tag{31}$$

By substituting $\boldsymbol{\Gamma}$ with equation (3), $\boldsymbol{H}$ with equation (31), and regrouping the terms in problem (26), we can rewrite it to

$$\begin{aligned} &\text{-log}\,\mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) \\ &\quad\leq \frac{1}{8}\|\boldsymbol{H} - \boldsymbol{\Gamma}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c \\ &\quad= \frac{1}{8}\|\boldsymbol{H} - (\boldsymbol{\alpha}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}' + \boldsymbol{C}\boldsymbol{D}')\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c \\ &\quad= \frac{1}{8}\|(\boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i} + n_i^{-1}\boldsymbol{H}\mathbf{1}\mathbf{1}' + n_u^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H}\boldsymbol{J}_{n_i}) - (\boldsymbol{\alpha}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}' + \mathbf{C}\mathbf{D}')\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c \\ &\quad= \frac{1}{8}\|(n_i^{-1}\boldsymbol{H}\mathbf{1}\mathbf{1}' - \boldsymbol{\alpha}\mathbf{1}') + (n_u^{-1}\mathbf{1}\mathbf{1}'\boldsymbol{H}\boldsymbol{J}_{n_i} - \mathbf{1}\boldsymbol{\beta}') + (\boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i} - \boldsymbol{C}\boldsymbol{D}')\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c. \end{aligned} \tag{32}$$

Now, $\boldsymbol{\beta}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are column-centered for identification purposes similar to the uniqueness restrictions for the bi-additive model discussed in Groenen et al. (2003). We have $\boldsymbol{\beta}' = \boldsymbol{\beta}'\boldsymbol{J}_{n_i}$ and $\boldsymbol{C}\boldsymbol{D}' = \boldsymbol{J}_{n_u}\boldsymbol{C}\boldsymbol{D}'\boldsymbol{J}_{n_i}$, such that

$$\begin{aligned} &\text{-log}\,\mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) \\ &\quad\leq \frac{1}{8}\|(n_i^{-1}\boldsymbol{H}\mathbf{1} - \boldsymbol{\alpha})\mathbf{1}' + \mathbf{1}(n_u^{-1}\mathbf{1}'\boldsymbol{H} - \boldsymbol{\beta}')\boldsymbol{J}_{n_i} + (\boldsymbol{J}_{n_u}(\boldsymbol{H} - \boldsymbol{C}\boldsymbol{D}')\boldsymbol{J}_{n_i})\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c \\ &\quad= \frac{1}{8}\left(\|(n_i^{-1}\boldsymbol{H}\mathbf{1} - \boldsymbol{\alpha})\mathbf{1}'\|_F^2 + \|\mathbf{1}(n_u^{-1}\mathbf{1}'\boldsymbol{H} - \boldsymbol{\beta}')\boldsymbol{J}_{n_i}\|_F^2 + \|\boldsymbol{J}_{n_u}(\boldsymbol{H} - \boldsymbol{C}\boldsymbol{D}')\boldsymbol{J}_{n_i}\|_F^2\right) \\ &\qquad+ \frac{1}{4}\operatorname{tr}\left(\mathbf{1}\left(n_i^{-1}\boldsymbol{H}\mathbf{1} - \boldsymbol{\alpha}\right)'\mathbf{1}\left(n_u^{-1}\mathbf{1}'\boldsymbol{H} - \boldsymbol{\beta}'\right)\boldsymbol{J}_{n_i}\right) + \frac{1}{4}\operatorname{tr}\left(\mathbf{1}\left(n_i^{-1}\boldsymbol{H}\mathbf{1} - \boldsymbol{\alpha}\right)'\boldsymbol{J}_{n_u}(\boldsymbol{H} - \boldsymbol{C}\boldsymbol{D}')\boldsymbol{J}_{n_i}\right) \\ &\qquad+ \frac{1}{4}\operatorname{tr}\left(\boldsymbol{J}_{n_i}\left(n_u^{-1}\mathbf{1}'\boldsymbol{H} - \boldsymbol{\beta}'\right)'\mathbf{1}'\boldsymbol{J}_{n_u}(\boldsymbol{H} - \boldsymbol{C}\boldsymbol{D}')\boldsymbol{J}_{n_i}\right) \\ &\qquad+ \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c. \end{aligned} \tag{33}$$

Using the cyclic property of the trace and the property of centering matrices that $\mathbf{1}'\boldsymbol{J}_{n_u} = \mathbf{0}$ and $\boldsymbol{J}_{n_i}\mathbf{1} = \mathbf{0}$,

we can determine that all crossproducts are equal to zero. This leaves us with

$$-\log \mathcal{L}(\boldsymbol{\Gamma}|\boldsymbol{Y}) \leq \frac{1}{8}\left(\|\left(n_i^{-1}\boldsymbol{H}\mathbf{1}-\boldsymbol{\alpha}\right)\mathbf{1}'\|_F^2 + \|\mathbf{1}\left(n_u^{-1}\mathbf{1}'\boldsymbol{H}-\boldsymbol{\beta}'\right)\boldsymbol{J}_{n_i}\|_F^2 + \|\boldsymbol{J}_{n_u}\left(\boldsymbol{H}-\boldsymbol{C}\boldsymbol{D}'\right)\boldsymbol{J}_{n_i}\|_F^2\right) + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 + c. \tag{34}$$

This allows us to split the problems into three seperate minimization problems. The majorization function in (34) is minimized with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ when

$$\boldsymbol{\alpha} = n_i^{-1}\boldsymbol{H}\mathbf{1}, \tag{35}$$

$$\boldsymbol{\beta}' = n_u^{-1}\mathbf{1}'\boldsymbol{H}\boldsymbol{J}_{n_i}, \tag{36}$$

as both $(n_i^{-1}\boldsymbol{H}\mathbf{1}-\boldsymbol{\alpha})$ and $(n_u^{-1}\mathbf{1}'\boldsymbol{H}-\boldsymbol{\beta}')$ will then be zero. The next step is to obtain estimates for $\boldsymbol{C}$ and $\boldsymbol{D}$ to minimize the remainder of the majorization function. In other words, we want to find the solution to

$$\min_{\boldsymbol{C},\boldsymbol{D}} \frac{1}{8}\|\tilde{\boldsymbol{H}}-\boldsymbol{C}\boldsymbol{D}'\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2, \tag{37}$$

where $\tilde{\boldsymbol{H}} = \boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i}$. To do this, we use the Rank-Restricted Soft SVD algorithm from Hastie et al. (2015). This algorithm alternates between fixing $\boldsymbol{C}$ and minimizing (37) with respect to $\boldsymbol{D}$ using a ridge regression, and vice versa. After each ridge regression, singular value decomposition (SVD) is applied on the solution in order to obtain a low-rank approximation. This process is iterated until the convergence of $\boldsymbol{C}\boldsymbol{D}'$. Since the ridge regressions often produce singular values that are very close but not exactly zero, a final, *soft-thresholding* SVD is computed after convergence, which sets the singular values to zero if they are below λ. This soft-thresholding 'cleans up' the singular values and enables us to determine the rank of the solution. More formally, the algorithm is given in Algorithm 1.

Algorithm 1 Rank-Restricted Soft SVD for finding $\boldsymbol{C}$ and $\boldsymbol{D}$

Initialize

$\tilde{\boldsymbol{H}} \leftarrow \boldsymbol{J}_{n_u}\boldsymbol{H}\boldsymbol{J}_{n_i}$

$\boldsymbol{U} \leftarrow$ an $n_u \times f$ random matrix with orthonormal columns

$\boldsymbol{Z} \leftarrow \boldsymbol{I}_f$

$\boldsymbol{C} \leftarrow \boldsymbol{U}\boldsymbol{Z}$

repeat

Given $\boldsymbol{C}$, solve for $\boldsymbol{D}$:

$$\hat{\boldsymbol{D}} = \arg\min_{\boldsymbol{D}} \frac{1}{8}\|\tilde{\boldsymbol{H}} - \boldsymbol{C}\boldsymbol{D}'\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{D}\|_F^2 = \tilde{\boldsymbol{H}}'\boldsymbol{U}\boldsymbol{Z}(\boldsymbol{Z}^2 + 4\lambda\boldsymbol{I})^{-1}$$

Compute the SVD of $\hat{\boldsymbol{D}}\boldsymbol{Z} = \hat{\boldsymbol{V}}\hat{\boldsymbol{Z}}^2\boldsymbol{R}'$, and let $\boldsymbol{V} \leftarrow \hat{\boldsymbol{V}}$, $\boldsymbol{Z} \leftarrow \hat{\boldsymbol{Z}}$, and $\boldsymbol{D} \leftarrow \boldsymbol{V}\boldsymbol{Z}$

Given $\boldsymbol{D}$, solve for $\boldsymbol{C}$:

$$\hat{\boldsymbol{C}} = \arg\min_{\boldsymbol{C}} \frac{1}{8}\|\tilde{\boldsymbol{H}} - \boldsymbol{C}\boldsymbol{D}'\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{C}\|_F^2 = \tilde{\boldsymbol{H}}\boldsymbol{V}\boldsymbol{Z}(\boldsymbol{Z}^2 + 4\lambda\boldsymbol{I})^{-1}$$

Compute the SVD of $\hat{\boldsymbol{C}}\boldsymbol{Z} = \hat{\boldsymbol{U}}\hat{\boldsymbol{Z}}^2\boldsymbol{R}'$, and let $\boldsymbol{U} \leftarrow \hat{\boldsymbol{U}}$, $\boldsymbol{Z} \leftarrow \hat{\boldsymbol{Z}}$, and $\boldsymbol{C} \leftarrow \boldsymbol{U}\boldsymbol{Z}$

until $\boldsymbol{C}\boldsymbol{D}'$ has converged

Compute the SVD of $\tilde{\boldsymbol{H}}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{Z}_\sigma\boldsymbol{R}'$, and let $\boldsymbol{S}_\lambda(\boldsymbol{Z}_\sigma) \leftarrow \text{diag}[\max(\sigma_1 - \lambda, 0), \ldots, \max(\sigma_f - \lambda, 0)]$, and $\boldsymbol{V} \leftarrow \boldsymbol{V}\boldsymbol{R}$

return $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{S}_\lambda(\boldsymbol{Z}_\sigma)^{1/2}$ and $\boldsymbol{D} = \boldsymbol{V}\boldsymbol{S}_\lambda(\boldsymbol{Z}_\sigma)^{1/2}$

Using the estimates of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ from equations (35) and (36) and $\boldsymbol{C}$ and $\boldsymbol{D}$ from Algorithm 1, we can re-estimate $\boldsymbol{H}$, which can then be used to find new estimates of the parameters of $\boldsymbol{\Gamma}$. We iterate these steps until the negative log likelihood in equation (15) has converged. The complete majorization algorithm is given below.

Algorithm 2 Complete majorization algorithm

initialize $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\beta}^{(0)}$, $\boldsymbol{C}^{(0)}$, $\boldsymbol{D}^{(0)}$

Subtract the column means from $\boldsymbol{\beta}^{(0)}$, $\boldsymbol{C}^{(0)}$, and $\boldsymbol{D}^{(0)}$

while $t = 0$ or $(\log L(\boldsymbol{\Gamma}^{(t)}|\boldsymbol{Y}) - \log L(\boldsymbol{\Gamma}^{(t-1)}|\boldsymbol{Y}))/\log L(\boldsymbol{\Gamma}^{(t-1)}|\boldsymbol{Y}) \geq \epsilon$ **do**

$t \leftarrow t + 1$

Update $\boldsymbol{H}$: $h_{ui}^{(t)} \leftarrow 8b_{ui}^{(t-1)}$

Update $\boldsymbol{\alpha}$: $\boldsymbol{\alpha}^{(t)} \leftarrow n_i^{-1}\boldsymbol{H}^{(t)}\mathbf{1}$

Update $\boldsymbol{\beta}$: $\boldsymbol{\beta}^{(t)\prime} \leftarrow n_u^{-1}\mathbf{1}'\boldsymbol{H}^{(t)}\boldsymbol{J}_{n_i}$

Update $\boldsymbol{C}^{(t)}$ and $\boldsymbol{D}^{(t)}$ using Algorithm 1

Update $\boldsymbol{\Gamma}$: $\boldsymbol{\Gamma}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t)}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}^{(t)\prime} + \boldsymbol{C}^{(t)}\boldsymbol{D}^{(t)\prime}$

Compute $-\log L(\boldsymbol{\Gamma}^{(t)}|\boldsymbol{Y})$

end while

return $\boldsymbol{\alpha}^{(t)}$, $\boldsymbol{\beta}^{(t)}$, $\boldsymbol{C}^{(t)}$, $\boldsymbol{D}^{(t)}$

4.3 Content-based filtering

Besides click data, we also have data on the offers that have been sent out. The characteristics of offer i, also presented in Table 1, can be represented by the vector $\boldsymbol{x}_i$. `MEAL_PLAN` is treated as an ordinal variable and dummies are created from `MAIL_MONTH`, `DEPARTURE_MONTH` and `COUNTRY_NAME`. As discussed in Section 2, content-based methods can only be applied in the case where we have at least one click from a user, indicating an offer that the user liked, which then can be used to find similar offers. On the subset of users that have seen at least a certain predetermined number of offers and have at least a certain predetermined number of clicks, we can model the probability of user u interacting with offer i as a standard logistic regression of y_{ui} on the item characteristics

$$p(y_{ui} = 1|\boldsymbol{x}_i, \boldsymbol{\theta}_u) = \pi_{ui} = \frac{e^{\boldsymbol{x}_i'\boldsymbol{\theta}_u}}{1 + e^{\boldsymbol{x}_i'\boldsymbol{\theta}_u}} \quad \forall u. \tag{38}$$

In some cases the number of predictors may exceed the number of observations, which may lead to overfitting. Therefore, we need to apply penalized regressions. The user-specific coefficient estimates $\hat{\boldsymbol{\theta}}_u$ are given by maximizing the following objective function, where a L_2 ridge penalty is added to the log-likelihood $l(\boldsymbol{\theta}_u)$ (Le Cessie and Van Houwelingen, 1992):

$$l^{\lambda}(\boldsymbol{\theta}_u) = \sum_{i \in \mathcal{V}_u} \left[y_{ui} \log\left(\pi_{ui}\right) + \left(1 - y_{ui}\right) \log(1 - \pi_{ui}) \right] - \lambda \sum_{j=1}^{p} \theta_{uj}^2, \tag{39}$$

where $\mathcal{V}_u$ is the set of vacation offers that user u has seen, λ controls the amount of shrinkage and p is the number of predictors. The resulting estimated coefficients $\hat{\boldsymbol{\theta}}_u$ can be used to predict the click probability of a given user u that sees offer i. For the users for which $\boldsymbol{\theta}_u$ is not estimated, we predict the global average clickrate. Finally, for the rare case where the user clicked on each of the seen offers, it is also impossible to estimate $\boldsymbol{\theta}_u$ and thus the prediction of the click probability will be one.

4.4 Model selection

In the next two subsections, we describe the process of hyperparameter optimization, which refers to selecting the hyperparameters that provide the best performance for each of the above models. Furthermore, we address the task of hybridization, the act of combining the predictions of the collaborative filtering and content-based models in such a way as to attain the best results.

To find the optimal hyperparameters and hybridization method, we first split the data. Figure 4 gives a representation of how the dataset is split. First, in order to provide unbiased results we reserve a randomly selected 20% of the data to present our final model performance. This set will only be used to present our findings in Section 5. Next, of the remaining 80%, another random 20% (16% of the total) is reserved for hybridization. This means that, once we select our ideal set of models with the best hyperparameters, this set is used to determine what combination of the models provides the best

results. The remaining 64% is therefore used for hyperparameter testing. The resulting subsets will be respectively referred to as the hyperparameters, combination and results set.

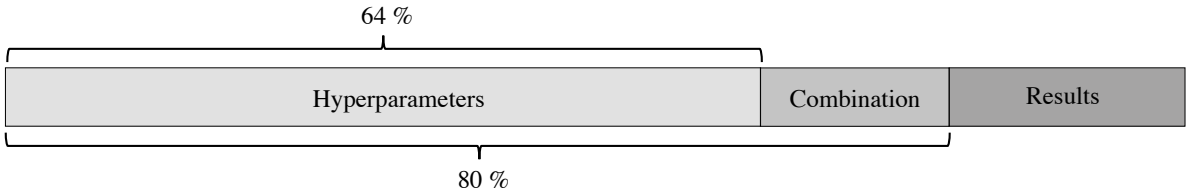


Figure 4: Illustration of the way the data is split.

4.4.1 Hyperparameter optimization

In order to optimize the hyperparameters for the three specified methods, we use 5-fold cross-validation. However, to reduce computation time, we first select a smaller subset of the hyperparameters dataset specified above. This subset consists of 10,000 randomly selected users, yielding a sample size of approximately 1.1 million observations.

In logistic matrix factorization with gradient ascent, the hyperparameters to optimize include step size δ, regularization parameter λ and the number of factors f. As excessive factors will be dropped by our MM algorithm we only optimize λ, and simply set the number of factors to a large enough value to avoid underfitting. In addition, in both cases, we compare the results of excluding the users with no clicks in the training set with the results where these users are included.

In the content-based model, the subset of users that we create a model for has an effect on the predictive accuracy of the model and the overall running time. The two constants controlling the size of this subset are the minimum number of observations τ_k and the minimum number of clicks τ_l necessary for users to be included in the subset. The best combination is determined using 5-fold cross-validation as well.

4.4.2 Hybridization

Hybrid systems are often used to improve accuracy (Çano and Morisio, 2017). To achieve this, we consider a weighted and a switching hybridization approach. Let $\hat{\mu}^{CBF}$ and $\hat{\mu}^{LMF}$ be the predicted probabilities from our content-based filtering method and the better one (as measured by the RMSE of the combination set) of the two logistic matrix factorization methods, respectively.

For some users, for example for those that have strong preferences about their vacations (i.e. they only like all-inclusive vacations in Egypt) and have sufficiently demonstrated their preferences through their clicks in the past, the content-based method is expected to work well. However, for users with fewer observations, only our collaborative filtering model can be used. Additional benefits of collaborative

filtering are that it can discover users that have similar click patterns and that it can internalize the general popularity of an offer. We propose the following hybrids, which will be trained on the combination set:

1. **Weighted average based on linear regression**: we regress the click behavior y on the predictions $\hat{\mu}^{CBF}$ and $\hat{\mu}^{LMF}$ without an intercept. The estimated coefficients of this regression are transformed to sum up to 1. These can be used as the weights ω_1 and ω_2 for the weighted averages.
2. **Switching based on the number of observations and number of clicks**: We trust the content based model more if it is based on more observations and/or more clicks. Therefore the number of observations and clicks per user k_u and l_u determine which method to use. If k_u and l_u both exceed certain thresholds, we use $\hat{\mu}_u^{CBF}$ for prediction for user u. Otherwise we use $\hat{\mu}_u^{LMF}$. The thresholds can be tuned to find the best switching criteria.

4.5 Implementation details

In this section, we shortly elaborate on the efficiency improvements made during the implementation of the previously discussed methods.

4.5.1 Logistic matrix factorization with iterated stochastic gradient ascent

The logistic matrix factorization with iterated stochastic gradient ascent method is implemented in `Python`. To further accelerate running times, we use a just-in-time (JIT) compiler, that translates `Python` code from the package `numpy` into fast machine code. Sparse matrices are handled with the `scipy.sparse` package.

4.5.2 Logistic matrix factorization with majorization

The bulk of the implementation of this method is written in `R`. Besides the use of the spare plus low-rank data structure as discussed in Section 4.2.2 we implemented `C++` code using the `Rccp` and `RccpArmadillo` packages in `R` for various parts of the algorithm. While `R` is already rather efficient when using vectorized functions such as calculating the logarithm of all elements in a matrix, we find that lower level language implementation yields notable efficiency improvements for the calculation of specific elements of $\Gamma^{(0)}$ as these calculations are based on loops, which are extremely efficient in `C++`. These values are later used for the purpose of calculating first derivatives.

Initialization of the parameters by means of a warm start is also found to decrease computation time. We initialize the main effect vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ using the average clickrate per user and average clickrate per item respectively. For $\boldsymbol{\alpha}$ we take the γ values that produce the corresponding average click rates per user.

For $\boldsymbol{\beta}$ we do the same, but for the average click rates per order, and demean this result. Both averages are derived from the training set. Furthermore, to decrease the cross-validation times we implement an additional warm start mechanism. This warm start consists of using the converged parameter estimates from the previous set of hyperparameters as the initial parameters for the next set of hyperparameters. The effect of this warm start is that if the method cannot decrease the objective function of equation (15) further, the method will only perform a single iteration. We also made sure to cross-validate in a descending order of λ. Starting with a very large lambda, all factors in $\boldsymbol{C}$ and $\boldsymbol{D}$ are shrunk towards zero and we simply only have nonzero $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in $\boldsymbol{\Gamma} = \boldsymbol{\alpha}\mathbf{1}' + \mathbf{1}\boldsymbol{\beta}' + \boldsymbol{C}\boldsymbol{D}'$. When we slowly decrease λ, either a factor in $\boldsymbol{C}$ and $\boldsymbol{D}$ is released and the objective function decreases or nothing changes and the previous best estimate is returned.

5 Results

In this section, we present the main results of our methods. First, the logistic matrix factorization and content-based models are evaluated separately. For logistic matrix factorization, the ISGA and MM optimization methods are contrasted on their solutions at convergence and RMSE for the combination set. Furthermore, for the MM method we evaluate the speed-up due to using the sparse plus low-rank framework. Lastly, the performance of the final (hybrid) models is evaluated on the results dataset.

5.1 Logistic matrix factorization

For ISGA, 5-fold cross-validation yields final parameter values of $\delta = 1$, $\lambda = 5$ and $f = 10$. Furthermore, we noticed that while the exclusion of users with no clicks prior to estimation makes for faster running times, the resulting predictions yield slightly higher RMSE values. Therefore, we keep these users for estimation, but overwrite their predictions with zeroes later on. Another interesting finding is that RMSE for the test set does not seem to improve significantly after a certain number of iterations (approximately 30), although at this point the gradients have not yet reached zero.

For logistic matrix factorization using majorization, we find that the optimal regularization rate λ is 10, for which, in the final model for the results predictions, 10 factors were retained. Furthermore, during cross-validation we find that, similar to ISGA, removing the users with no observed clicks from the training set yields a higher RMSE. We therefore take the same approach as in ISGA by including them in the estimation, but predicting only zeroes for these users.

5.1.1 Comparison of logistic matrix factorization methods

As noted before, MM and ISGA both serve to optimize the same objective function. A big difference in these methods however is that in each iteration MM uses all data to update the parameters. ISGA, on the other hand, only uses a subset, and could therefore potentially provide less accurate or less beneficial

parameter updates. First, to illustrate why it is necessary to use stochastic as opposed to full gradient ascent, we compare one iteration in the MM method to one iteration using full gradient ascent. Using the largest available dataset, that is the full set of observations, and 10 factors, we find that the average duration of an iteration, taken over 5 iterations, is 681 seconds for full gradient descent and 43 seconds for our MM method.[2] It should be noted however that these iterations are not directly comparable, as one method may be 'more effective' per iteration than the other. Regardless, the fact that the iterations for full gradient descent would take about 16 times longer than the MM iterations seems to indicate that the MM method is likely superior when one wants a method that uses the full dataset.

Before we move to comparing the performance on a test set, we want to provide some insight into the convergence behavior of the two methods. Convergence is most easily illustrated by plotting the objective function over the iterations. Figure 5 illustrates values for the objective function[3], using either the MM or ISGA method for a model containing 10 factors, and with λ equal to 5. We show different data sizes, with 10^4, $10^{4.5}$, and 10^5 users. It is apparent that the ISGA method finds a different local optimum, given that its log likelihood converges to a different point, compared to MM. Moreover, it appears that the ISGA method also finds a less optimal objective value in all three cases. Furthermore, we see that the objective function improves quickly for both methods, after which the speed of convergence greatly decreases.

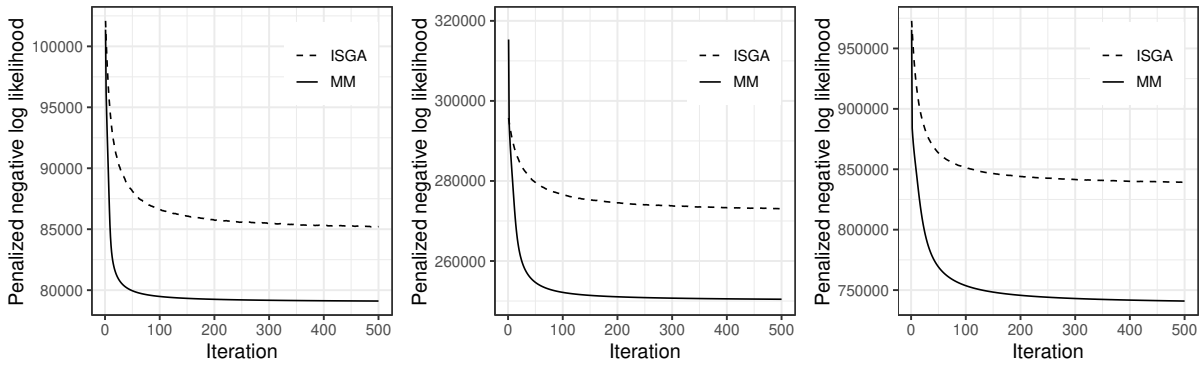


Figure 5: Comparison of the negative log likelihood of the ISGA and MM methods for 10^4, $10^{4.5}$, and 10^5 users.

Whilst the figures show quite big differences in the values reached for the objective functions, this gap does not necessarily show in the achieved test RMSE values. Using the following convergence criteria: a percentage change of the objective function below 10^{-6} for MM and norms of the gradients of $\boldsymbol{C}$ and $\boldsymbol{D}$ being below 10^{-6} for ISGA, we find RMSE values for the combination set of 0.1377 and 0.1389 for MM and ISGA, respectively. Again, relative to the difference in objective function, these differences in RMSE are not very large, and when inspecting the actual development of the RMSE for the test set over the iterations, we observe that the curve flattens out well before the convergence criteria are reached.

[2] As run on a MacBook Pro, 2.3 GHz Dual-Core Intel Core i5, 8 GB 2133 MHz LPDDR3, Intel Iris Plus Graphics 640 1536 MB.

[3] Note that the objective function value of the ISGA method is multiplied by minus one to be able to perform this comparison.

5.1.2 Speed-up due to sparse plus low-rank approach

To analyse the effectiveness of the sparse plus low-rank (S+LR) approach that is taken in our majorization method, we present the results of a simulation study. We create datasets with varying number of users and varying degrees of sparsity. Furthermore, we evaluate if, besides the size of the data, the number of factors in the fitted model also affects the comparative iteration time.

Figure 6 illustrates the computational benefits when using a sparse plus low-rank approach for a model containing 5 factors.[4] The (logarithmic) x-axis illustrates the size of the problem; keeping the number of orders fixed at 100, the number of users is increased from 100 to 1,000,000. Naturally, this has a significant effect on the size of any matrix multiplications that are done with the dimensions of the full data matrix. As for the iteration times on the y-axis, these values are averaged over 5 iterations for each data point respectively. Each figure contains results for the S+LR approach (black lines) and the 'full matrix' approach (grey lines). Furthermore, results are given for different degrees of sparsity (the percentage of missing values in the data matrix). The left hand panel of the figure shows the large increase in iteration time with the increase in users. Besides this clear increase in overall iteration times, we observe that iteration times for higher degrees of sparsity tend to be lower, which is to be expected given the fact that some calculations in the algorithm only apply to the actual observations. Looking at the differences between the S+LR and full matrix methods, it is clear that the S+LR method is considerably quicker. The center panel is similar to the left panel, but uses a logarithmic scale for the iteration time. This helps to make the iteration times for the smaller user numbers readable. Overall we observe that up until 100,000 users (hence, $100,000 \times 100 \times 0.05 = 0.5$ million observations for a sparsity of 95%, and 2.5 million observations for a sparsity of 75%), iteration times are about one to five seconds. However, past this point we find that at 1,000,000 users the iterations are considerably longer, with the full matrix method reaching iteration times close to three minutes on the dataset with low sparsity.

The right hand panel illustrates the speed-up the best. This figure plots the ratio of iteration times, with the full matrix method over the S+LR method, again with varying numbers of users. Here we see that whilst differences are negligible for smaller problems, the speed gap quickly increases when the number of users rises. This is promising given the large dimensionality of many recommender system problems. Using the largest simulated dataset, where the data matrix is of size 1,000,000 times 100, the full matrix method is over four times as slow as the S+LR approach. Interestingly, we see that for lower levels of sparsity the relative gap is smaller. This could be explained by the 'sparse' part in the sparse plus low-rank data structure becoming increasingly big. Regardless, we still observe a widening of the gap with an increasing data size.

[4] As run on a MacBook Pro, 2.3 GHz Dual-Core Intel Core i5, 8 GB 2133 MHz LPDDR3, Intel Iris Plus Graphics 640 1536 MB.

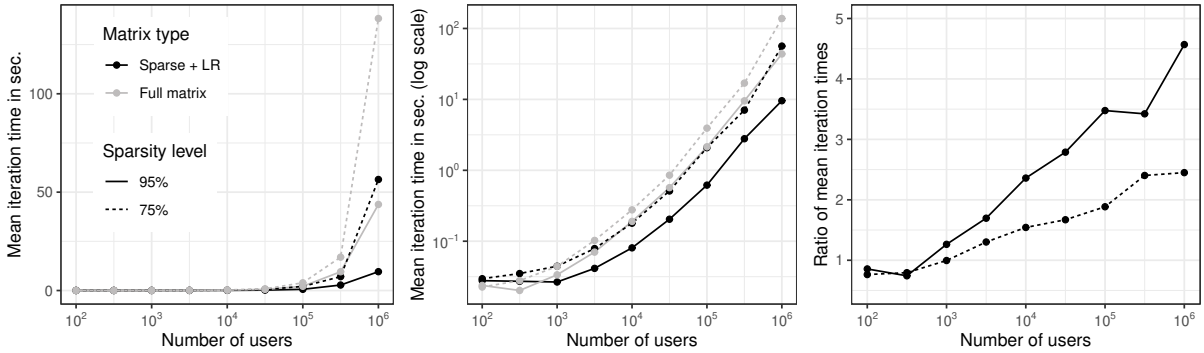


Figure 6: Speed-up due to S+LR for a model with 5 factors and data containing 100 offers and different levels of sparsity. The left panel illustrates the development of the mean iteration time in seconds when the number of users is increased. The middle panel is identical to the left panel, but uses a logarithmic y-axis. The right panel illustrates how the ratio of the iteration time for the full matrix method, over the time for the S+LR method develops with an increase in users.

Figure 7 presents the same graphs as Figure 6, but now for a model containing 20 factors. Overall, we see that the iteration times increase, which makes sense as some parts of the algorithm, such as the size of the updating problem for $\boldsymbol{C}$ and $\boldsymbol{D}$ directly depend on f. Whilst we again see that the S+LR method achieves lower iteration times, the relative differences appear (much) smaller as compared to the panels in Figure 6. This is may be due to the fact that, whilst $\boldsymbol{H}$ does not increase in size, the low-rank part does increase in size when the number of factors is increased. Looking at the panel on the right, we see that for the largest problem specification the full matrix method is only about 75% slower, for both sparsity levels. Regardless we still observe an increase (albeit smaller) in the relative difference with an increasing problem size, again emphasizing the potential power of our method in the context of large recommender system problems.

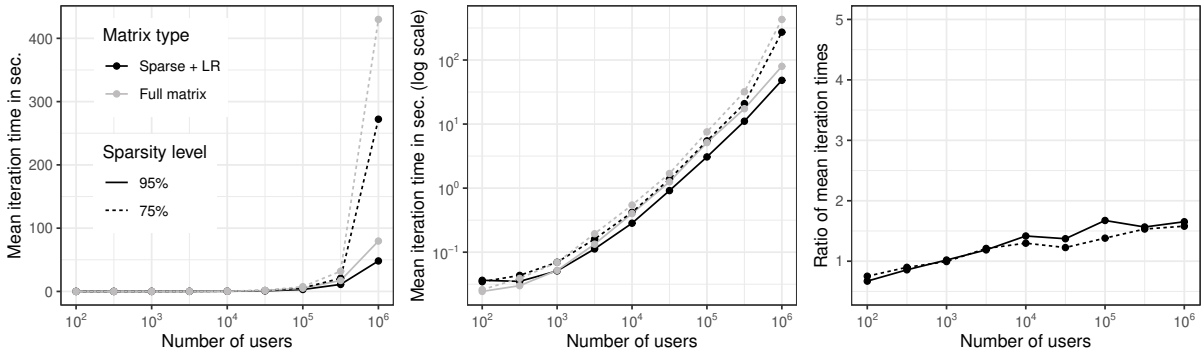


Figure 7: Speed-up due to S+LR for a model with 20 factors and data containing 100 offers and different levels of sparsity. The left panel illustrates the development of the mean iteration time in seconds when the number of users is increased. The middle panel is identical to the left panel, but uses a logarithmic y-axis. The right panel illustrates how the ratio of the iteration time for the full matrix method, over the time for the S+LR method develops with an increase in users.

5.2 Content-based filtering

After performing cross-validation, we conclude that having a minimum of 10 observations and 5 clicks is a reasonable criteria to filter the users for the inclusion in individual logistic regressions. This results in a total of 29,099 users for which we can effectively fit logistic regression models. This is computationally challenging, but not infeasible.

Naturally, the importance of predictors differs per user. To compare the importance, we run the model on normalized data to obtain standardized coefficients. Figure 8 displays a boxplot of the nine most important predictors (measured by their absolute value) along the size of the intercept. The coefficients are shrunken toward zero due to the ridge penalty, and most often the coefficients take the value zero. Therefore, only non-zero values of the coefficients are displayed. From the high variance of the intercept, it appears that the propensity to click varies greatly among individuals. The fact that all of the displayed coefficients take both positive and negative values, illustrates the heterogeneity of the user preferences. However, it is important to note that the estimates likely suffer from omitted variable bias and therefore no causal statements can be made. One likely indication of omitted variable bias is the fact that the `DISCOUNT` variable takes on negative coefficient values, which is generally unexpected.

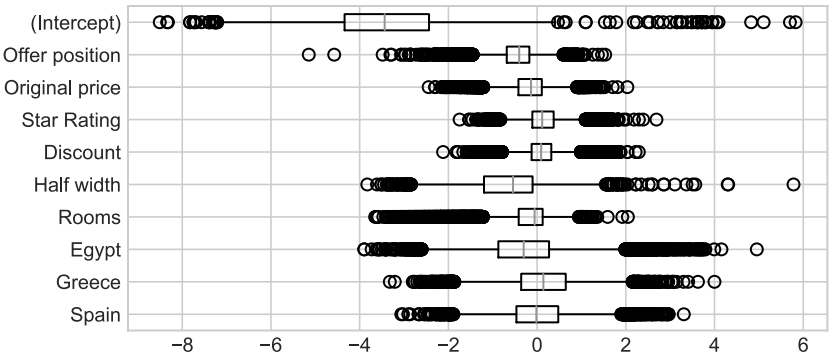


Figure 8: Distribution of a selection of standardized coefficients in the logit models.

5.3 Hybrid model performance

In this section, we compare and contrast the performance of the standalone and hybrid models. It is important to note that both the ISGA and the MM work on the the same model and objective function. Results can differ for various reasons, such as the previously discussed difference in convergence criteria or reaching a different local optimum. Therefore, the results obtained on this specific dataset are not meant to conclude on which method is better in general, but simply highlights which method converged to the better optimum in the context of our problem and furthermore illustrates the performance gain of the hybridization methods. We commence with a comparison based on the RMSE, after which we

evaluate a set of popular performance measures for binary classification problems, namely ROC curves and precision-recall curves.

5.3.1 RMSE

We tune the two different hybrid models based on results for the combination set. Since MM outperforms ISGA on the combination set, we combine MM with the content-based model for the hybrids. For the weighted hybrid, where the weights are determined via linear regression of the actual clicks on the predictions, we find that approximately 41% of the weight is placed on the content-based model. The second hybrid model, which switches based on the number of observations and number of clicks, yields optimal thresholds for k_u and l_u of 200 and 47, respectively. The resulting RMSE of the models of the three separate methods and the hybrid methods, calculated from the predictions on the results set, can be found in Table 2. B1-B4 is used to abbreviate the four baseline methods introduced in Section 4.1, ISGA and MM refer to the logistic matrix factorization with iterated stochastic gradient ascent and majorization respectively, CBF is used for the content-based model, and H1-H2 refers to the hybrid methods from Section 4.4.2.

Table 2: Test RMSE of the methods used.

	B1	B2	B3	B4	ISGA	MM	CBF	H1	H2
RMSE	0.1483	0.1466	0.1414	0.1456	0.1393	0.1366	0.1385	0.1357	0.1369

The best baseline is the average click rate per user (B3). In terms of RMSE, this baseline method is 4.6% better than simply predicting zeroes everywhere (B1). All our methods outperform the best baseline, with logistic majorization boasting the lowest RSME among the separate methods. The weighted average of the predictions by the content-based model and logistic majorization (H1) has the best performance among all our methods. It outperforms the best baseline by 4% and the majority rule by 8.5%. The switching hybrid uses predictions from the content-based model in 5% of the observations, and majorization for the remaining observations. The fact that this model did not yield a lower RMSE than majorization itself leads us to conclude that the content-based model's accuracy is not simply a function of the number of observations and the number of clicks per user - the assumption on which this switching hybrid was built. As mentioned earlier, the content-based method can be very useful for some users that have clear and observable preferences with regards to vacation offers, but has the disadvantage that it will only recommend (predict a high click probability in this case) offers that are similar to some other offers from the past.

5.3.2 ROC curves

Besides the RMSE, we can also look at another popular performance measure for (binary) classification problems: the ROC curve, as introduced by Hanley and McNeil (1982). To plot the ROC curve, the predicted click probabilities are converted to discrete class values using varying cut-off values. We classify a data point from the test set as a positive case if the predicted click probability is above a certain cut-off value, while those below the cut-off value are classified as negative cases. The ROC curve then plots the true positive rate $= \frac{\text{true positives}}{\text{true positives+false negatives}}$, which is also referred to as 'sensitivity', on the y-axis. A rate close to one implies that most positive cases (in our case a click on an offer) are classified correctly. On the x-axis, the false positive rate $= \frac{\text{false positives}}{\text{false positives+true negatives}}$, also known as the 'inverted specificity', is plotted. A low false positive rate implies that negative cases are rarely classified as positive. The curve in turn 'progresses' through different cut-off values, moving from left to right. The ROC curves are plotted in Figure 9. Generally one wants an ROC curve that hugs the upper left corner, such that a cut-off value can be selected for which most of the positive cases are caught, while at the same time not many negative cases are misclassified as positive cases. A good way of quantifying this is by looking at the area under the curve (AUC). The AUCs are given in Table 3 which shows that the first hybrid model gives the highest AUC value.

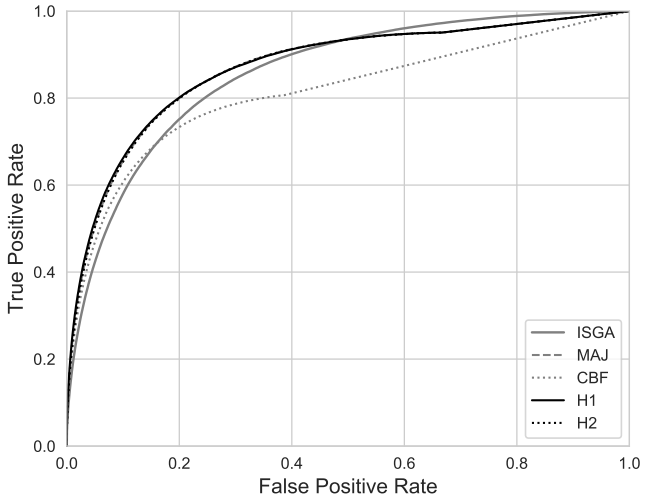


Figure 9: ROC curve of the methods used.

Table 3: AUC of the methods used.

	ISGA	MM	CBF	H1	H2
AUC	0.8558	0.8692	0.8109	0.8703	0.8447

5.3.3 Precision-recall curves

A problem with the ROC curve and AUC is that it can be deceiving for imbalanced datasets (Saito and Rehmsmeier, 2015). For example, let us consider a dataset which has many negatives, a model that simply predicts many negatives will have both a high true negative number and a small false negative number. The former will cause the false positive rate to be low, regardless of the number of false positives as the true negatives 'overpower' the false positives in the denominator. As for the true positive rate, often predicting negatives would cause the false negatives to be very low, making the denominator in the true positive rate small and therefore making the rate itself close to one. In our application, the average click rate is 2.19%, implying we have a very unbalanced dataset. Furthermore, for our data, it would make sense for the positive cases to be more valuable than the non positive cases. Given that customers rarely click, one really wants to extract the cases where they do.

Precision-recall (PRC) plots might be better suited to compare the respective methods as this plot is more informative than the ROC curve when evaluating binary classifiers (Saito and Rehmsmeier, 2015). The PRC plot displays for different cut-off values the precision $= \frac{\text{True Positives}}{\text{True Positives + False Positives}}$ on the y-axis and the recall $= \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$ (which is the same as sensitivity) on the x-axis. Notice that, in calculating precision and recall, true negatives are not used. Thus, the PRC plot is only concerned with the correct prediction of the minority class, the positive cases. The PRC plot is shown in Figure 10. Generally one would like a PRC plot which hugs the upper right corner. We see however, that this may not be the case here, indicating that the classification performance of the models may not be as good as the ROC curve suggests due to imbalanced data. Similar to the ROC curve we can quantify the performance of the methods according to the PRC plot by calculating the area under their curves. These results are displayed in Table 4. We can see that again the first hybrid model performs best.

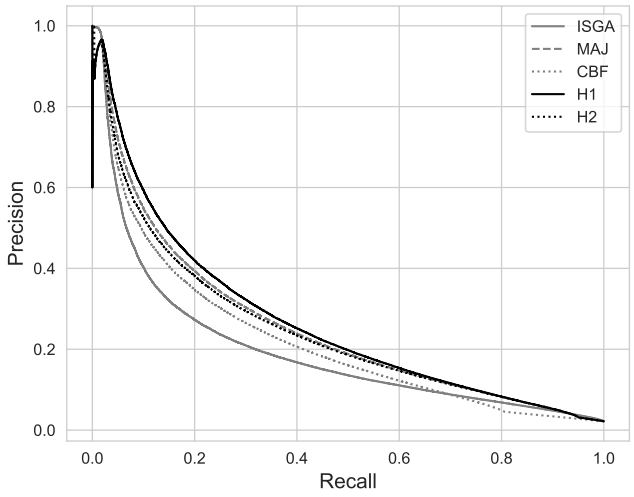


Figure 10: PRC plot of the methods used.

Table 4: Area under the curve of the PRC plot for the methods used.

	ISGA	MM	CBF	H1	H2
AUC	0.1954	0.2534	0.2212	0.2664	0.2472

6 Conclusion and discussion

In this paper we try to find an answer to the research question *'To what extent can we accurately and efficiently predict the clicking behavior of customers?'* First, we discuss the accuracy of our models. We find that in the context of our problem, beating the various specified baseline predictions is a challenging task. Independently, in terms of RMSE, the best cross-validated models beat the best baseline by 0.0021 (1.5%), 0.0048 (3.4%) and 0.0029 (2.1%) for the logistic matrix factorization with iterated stochastic gradient ascent (ISGA), logistic matrix factorization with majorization (MM), and content-based filtering models, respectively. This is in accordance with our first hypothesis, as the logistic matrix factorization that we specify has the ability to encapsulate the baseline cases using $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, but also adds additional components like the bi-additive effect $\boldsymbol{CD}'$. Whilst the RMSE improvements may appear small, even a small percentage increase in the accuracy of the models might be valuable for companies implementing the methods for millions of customers and thousands of items. We find the best hybridized model by combining the results of the MM algorithm and the content-based method using a weighted average based on linear regression. This model improves the RMSE relative to the baseline by 0.0057 (4.0%). It should be noted that this seemingly small performance increase might be due to the relatively small number of users with enough clicks to make a content-based model feasible. Furthermore, to compare the classification performance of the various methods, we plot the ROC curve and the PRC curve. Based on these plots, we also find that the aforementioned hybrid model performs best.

As for efficiency, we demonstrate that in accordance with the second hypothesis, compared to full gradient ascent, iteration times for MM are notably shorter, whilst still incorporating all data. Furthermore, the sparse plus low-rank addition to the MM method generally lowers computation time significantly compared to the same method using dense matrices, which is in agreement with our third hypothesis. This speed-up seems to increase with the size of the problem, and seems especially large for highly sparse data. This is promising, as this closely describes many recommendation problems in practice.

One of the main theoretical advantages of the majorization method is that it guarantees descent of the objective function, a characteristic which does not apply to the ISGA method. Regardless, we hypothesized that the two methods would converge to roughly the same optimum, yet this appeared not to be the case in this application, as the MM algorithm consistently reaches an objective value that is better than the log likelihood produced by the ISGA method.

We identify some areas that may be interesting for future research. Firstly, since the click data used

for the collaborative filtering methods is substantially sparse and only 2.19% percent of the non-sparse elements corresponds to a click, each click contains a lot of information. Therefore, it would be interesting to investigate methods which weigh clicks differently from non-clicks. Furthermore, giving more weight to mails that have at least one click may also provide better results. The reason behind this is that a user may open an email without really looking at its content, meaning that their non-clicks do not necessarily correspond to disinterest. However, the presence of at least one click in the mail implies that the user actually looked at the offers in the e-mail and chose the ones that they found interesting.

Another point that could be evaluated more closely is the fact that we fix a_{ui} to $\frac{1}{8}$ in (22). Whilst this provides computational benefits, it likely also affects our step length. One could investigate the trade-off between additional computations and more effective iterations further. Lastly, the convergence of the ISGA method could be investigated more closely. For example, to prevent the method from converging to a local optimum, one could possibly look at multiple initializations of parameters, adaptive step sizes or adjusting the size of the subsets used to calculate the gradient.

References

Ahn, S., Korattikara, A., Liu, N., Rajan, S., and Welling, M. (2015). Large-scale distributed bayesian matrix factorization using stochastic gradient mcmc. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 9–18.

Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

Çano, E. and Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524.

de Leeuw, J. and Lange, K. (2009). Sharp quadratic majorization in one dimension. *Computational statistics & data analysis*, 53(7):2471–2484.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

Févotte, C. (2011). Majorization-minimization algorithm for smooth itakura-saito nonnegative matrix factorization. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1980–1983. IEEE.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77.

Ghazanfar, M. and Prugel-Bennett, A. (2010). Building switching hybrid recommender system using machine learning classifiers and collaborative filtering. *IAENG International Journal of Computer Science*, 37(3).

Groenen, P., Giaquinto, P., and Kiers, H. (2003). Weighted majorization algorithms for weighted least squares decomposition models. Technical report.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Hastie, T., Mazumder, R., Lee, J. D., and Zadeh, R. (2015). Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402.

He, X., Zhang, H., Kan, M.-Y., and Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee.

Johnson, C. C. (2014). Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems*, 27.

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

Le Cessie, S. and Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(1):191–201.

Li, F., Wu, B., Xu, L., Shi, C., and Shi, J. (2014). A fast distributed stochastic gradient descent algorithm for matrix factorization. In *Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 77–87.

Li, G. and Ou, W. (2016). Pairwise probabilistic matrix factorization for implicit feedback collaborative filtering. *Neurocomputing*, 204:17–25.

Lin, Z., Xu, C., and Zha, H. (2017). Robust matrix factorization by majorization minimization. *IEEE transactions on pattern analysis and machine intelligence*, 40(1):208–220.

Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1):76–80.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322.

Miranda, T., Claypool, M., Gokhale, A., Mir, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*. Citeseer.

Mnih, A. and Teh, Y. W. (2012). Learning label trees for probabilistic modelling of implicit feedback. In *Advances in Neural Information Processing Systems*, pages 2816–2824.

Mobasher, B., Jin, X., and Zhou, Y. (2003). Semantically enhanced collaborative filtering on the web. In *European Web Mining Forum*, pages 57–76. Springer.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE.

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–59.

Ricci, F., Rokach, L., and Shapira, B. (2015). *Recommender systems handbook*. Springer.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3).

Zheng, Y., Liu, C., Tang, B., and Zhou, H. (2016). Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 2–6. ACM.