Erasmus University Rotterdam

Master research
Business Analytics & Quantitative Marketing

Comparing Recommender Systems for a Large Assortment

Author:
P. VERHEIJEN MSc

Supervisor:
Prof. dr. B. DONKERS

Erasmus School of Economics

November 17, 2018

Abstract

In today's online world where content is abundant and attention is scarce, showing the right products to a consumer is becoming increasingly important. Tools that address this problem have been researched intensively over the years. The release of a large consumer rating data set released by Netflix propelled research on this topic. In this thesis three different models for online recommendations are compared: Matrix Factorization, Non-Negative Matrix Factorization and the Cluster Affiliation Model for Big Networks. Using the BigCLAM algorithm as a recommender system is a novel approach since the algorithm is originally developed to be used as a community detection model. The models are applied to purchasing data from a large online retailer after which the results are compared in terms of recommendations, estimation speed and interpretability. The results show that the BigCLAM model is an interesting and effective alternative that may offer higher predictive performance that requires less estimation time.

1 Introduction

In today's online world where content is abundant, and attention is scarce, showing the right product the the right consumer is becoming increasingly difficult and important. A technique called Matrix Factorization is often used as the basis of algorithms that addresses these problems in the form of recommender systems. Recommender systems attempt to derive the preferences of customers and serve these customers based on their preferences. In this research three different models are compared that can be used as Recommender systems. The performance of Matrix Factorization (MF) and Non-Negative Matrix Factorization (NMF) is compared with a different algorithm called Cluster Affiliation Model for Big Networks (BigCLAM). This algorithm is designed as a method to find clusters in social networks by deriving community affiliations of individuals that best explain the a given network. This model is a variation on NMF only with a different objective function. These models are compared based on their performance in predictions by calculating their precision and a ranked score. Furthermore, the models are compared on their sensitivity to local minima, the estimation speed, the predictive performance and lastly the interpretability of their latent dimensions.

Now the structure of this work is elaborated. First an overview of the literature study is given. Then the models are introduced by giving an intuitive explanation in words of the goals of the techniques. After which the initialization of the models is discussed. The algorithms are initialized with the goal of making the model estimation as fast as possible. Then the estima-

tion procedures are elaborated upon. In traditional MF gradient descent is used to estimate the models. In this paper Stochastic Gradient Descent (SGD) is used for reasons disclosed in the Method section. The learning rate and the batch size and their effect on the estimation procedure are then elaborated upon. Generally overfitting is avoided by means of regularization. How this is done is elaborated upon and the alternative method that is used, is elaborated upon as well. After which mini-batching is introduced which is used to assist the use of SGD by making the gradient less volatile. Lastly in the Methods section of this paper the performance measures of the models are introduced. Then the data set that is used for this thesis is introduced. The data set is split in a train, select and a test data set. The models are estimated and evaluated with this data sets and the results are reported. The findings are reported in the Conclusions section of this paper and the last chapter is the Discussion and Future Research chapterin which shortcomings and suggestions for future research are discussed.

2 Literature Review

Cotter and Smyth (2000) argue that although today's world is offering us with unpredencted access to larger and larger amounts of electronic information, we are faced with significant problems when it comes to finding the right information at the right time. A simple example of this is offered by Jacobs, Donkers and Bas (2016). They claim that product assortments have exploded in size over the past two decades and that this can be largely attributed to the rise of online shopping which has allowed retailers to retain assortments that are virtually unlimited in size. Koren, Bell and Volinksy (2009) claim that helping consumers deal with this overload of information is of paramount importance, since matching consumers with the most appropriate products is key to enhancing user satisfaction and loyalty. But in order to help consumers deal with this overload of information, consumer data is required. Fortunately, customer characterics and customer purchase history data are usually readily available (Jacobs, Donkers and Fok, 2016). This information is further supplemented with search and click behaviour (Jacobs, 2017).

Several decades ago it was already argued by Shardanand and Maes (1995) that we need technology to help us deal with the aforementioned problem, namely, helping consumers deal with this overload of information. These early techniques, however, faced problems such as "popularity bias", "early rater" problem and "sparse ratings" (see, e.g., Burke and Robin, 1999; Carenini and Sharma, 2004; Hsu et al., 2004; Celma and Herrera, 2008) that are still faced today. In order to overcome these problems, research quickly followed on combining different techniques (see,

e.g., Claypool, Gokhale, Miranda, Murnikov, Netes and Sartin, 1999).

These techniques are now known as personalization technologies and recommender systems. Adomavicius and Kwon (2007) claim that most online shopping sites and many other applications nowadays use recommender systems to help online consumers. The rise of personalization technologies and recommender systems started with filtering techniques. In specific, content-based filtering and collaborative filtering. Content-based filtering systems select items based on the correlation between items (van Meteren and van Someren, 2009). Collaborative filtering systems help people make choices based on the opinions of other people. It is based on the idea that people who have agreed with each other in the past, will most likely agree again with each other in the future (see, e.g., Resnick et al., 1994; Hill et al., 1995; Shardanand and Maes, 1995).

Another type of recommender system that was also being developed roughly simultaneously were knowledge-based recommender systems. These knowledge-based recommender systems attempt to use knowledge about users and products to generate recommendations. Trewin and Shari (2000) argue that knowledge-based recommenders are not only valuable systems on their own, but also highly complementary to other types of recommender systems. This type of recommender systems generally do not have the same cold start problem (;early rater problem) as mentioned earlier. Adomavicius and Kwon (2005) claim that it may not be sufficient in many applications to only consider users and items, they claim that is also important to incorporate contextual information of the user's decision scenario into the recommendation process. This knowledge-based approach to generation a recommendation could help improve recommendations. Going through the literature on these different techniques, it becomes apparant that each of these have their own strenghts and their own weaknesses. Studies such as Burk, Bursilovsky, Kobsa and Nejdl (2007) and Bell, Koren and Volinsky (2006) argue that blending multiple techniques into hybrid solutions offers far more predictive accuracy and therefore researchers should focus deriving substantially different approaches rather than refining single techniques.

In 2006 Netflix released a large consumer rating data set and announced a competition in which participants were encouraged to develop a recommender system that significantly outperformed their own. Research on recommender systems was hugely intensified and a whole new group of recommender systems were developed. This new group of recommender systems were at their esssence based on singular value decomposition (see e.g., Klema and Laub, 1980). Brandon (2006) claims that if meaningful generalities can help represent data with fewer numbers,

then finding a way to represent your data in fewer numbers can help find meaningful generalities. A technique that helps represent data with fewer numbers is Matrix Factorization. Koren, Bell and Volinksy (2009) claim that Matrix Factorization as a recommender system, in its most basic form, characterizes both items and users by vectors of factors inferred from rating patterns. A variant on Matrix Factorization is Non-Negative Matrix Factorization. Hoyer and Patrik (2004) describe this technique as a method of finding parts-based, linear representations of non-negative data and Lee and Seung (2001) argue that this technique has been shown to be an useful decomposition of multivariate data.

Many existing approaches and techniques to these type of techniques have difficulties dealing with very large datasets and with users who have very few rating, (see e.g., Hegelich and Jannach, 2009; Das, Datar, Garg and Rajaram, 2007), and computation efficiency and scalability are still an ongoing research topic (see, Goldberg, Roeder, Gupta and Perkins, 2001; Nathanson, Bitton and Goldberg, 2007; Jacobs, Donkers and Fok, 2016).

Over the years several other techniques have been proposed and researched in the literature. Some of these techniques are Probabilistic Matrix Factorization, Restricted Boltzmann Machines, Latent Direchtlet Allocation and Boolean Matrix Factorization. Probabilistic Matrix Factorization is a technique that is argued to perform well on large, sparse, and very imbalanced datasets such as the Netflix dataset (Mnih and Ruslan, 2008). A Restricted Boltzmann Machine is a technique that can be used to model tabular data, such as user's ratings of movies (Salakhutdinov, Mnih and Hinton, 2007). Latent Direchlet Allocation is a technique for collections of discrete data that provides full generative probabilistic semantics for documents. Where documents are modeled via a hidden Dirichlet random variable that specifies a probability distribution on a latent, low-dimensional topic space (see, Blei, Ng and Jordan, 2002; Jacobs, Donkers and Fok, 2016). And lastly, Boolean Matrix Factorization is a technique for the factorization of data sets in binary {0,1} alphabet based on Boolean Algebra (see, Snášel and Václav, et al., 2008; Miettinen and Vreeken, 2014).

Lastly, a topic that is unfortunately generally overlooked in the literature, is reviewed, namely user experience of Recommender Systems. Cosley, Lam, Albert, Konstan and Riedl (2003) show that showing predictions when users rate movies changes their ratings, although it is not known how long this change of opinion lasts. Furthermore Knijnenburg, Willemsen and Hirtbach (2010) show that choice satisfaction and system effectiveness increase when a system provides personalized recommendations (compared to the same system that provides random rec-

ommendations). Bollen et al., (2010) however argue on to this that larger sets containing only good items do not necessarily result in higher choice satisfaction compared to smaller sets, as the increased recommendation set attractiveness is counteracted by the increased difficulty of choosing from these sets. Lastly Knijnenburg, Niels, Reijmer and Willemsel (2011) and Knijnenburg et al. (2012) argue that researchers should consider that different types of users prefer different interaction methods and that accuracy only partially constitutes the user experience of a recommender system.

This thesis presents an alternative approach to NMF based on research on clustering algorithms for community detections. The objective function of this algorithm can be seen as a variant on NMF. The approach taken by the algorithm rests upon the fact that communities in networks often overlap as nodes can belong to multiple communities at once. Yang and Leskovec (2012) argue that identifying such overlapping communities is crucial for the understanding the structure as well as the function of real-world network. The model they propose is Cluster Affiliaton Model for Big Networks, which is an overlapping community detection method that scales to large networks of millions of nodes and edges. The model builds on their observation that overlaps between communities are densely connected, which they argue is in sharp contrast with present community detection algorithms. These algorithms implicitly assume that overlaps between communities are sparsely connected and thus cannot properly extract overlapping communities in a network.

In the last thirty years many techniques have been developed and tested. Researchers such as Bell, Koren and Volinsky (2007) argue that research should focus on deriving substantially different approaches, rather than refining single techniques. This thesis aims to contribute to this idea by researching a different approach to Non-negative Matrix Factorization.

3 Methods

Oftentimes in business the preferences of customers are of interest. By knowing their preferences customers can be better served. These preferences can be seen as relations between customers and products. Most of the time we already have an idea of some of these relationships, but not all the relationships are known nor are they exactly clear to us. Matrix Factorization is a technique that can help us discover these relations. In its most basic form it is a technique that tries to find relations between two groups. In this scenario these two groups are the consumers and the products. By applying this technique, we can uncover customer's preferences. Matrix Factorization is a well-known and researched technique. Several

variants have been created and each variant has its advantages and disadvantages. This research compares three different variants of Matrix Factorization, two of which are well-known and a third which is a new approach.

3.1 Matrix Factorization

Traditional Matrix Factorization, in its most basic form, attempts to characterize both items and users by vectors of factors inferred from feedback. This feedback consists of information that indicates (dis)interest either implicitly or explicitly. These vectors attempt to distill the underlying factors that correspond to the feedback. A high correspondence between an item's and user's vector indicate a positive link between the two. The strength of these links are used to determine potential (dis)interest. A high potential interest in a retail setting could mean that the product could be a good recommendation to the user. In an entertainment service setting, a high potential interest could mean that the user will probably enjoy certain content.

In order to determine the preferences of customers we need to derive the latent factor vectors that accurately describe them. These vectors can be based on either explicit or implicit feedback. Examples of explicit feedback are star ratings that Netflix uses or likes as seen on Facebook. Examples of implicit feedback are time spent viewing content, clicks and purchase history, the latter is used in this research. We can construct a matrix of the purchase history of the consumers, where one axis contains the users and the other axis contain the products. The value 1 in this matrix would denote that an individual u has purchased product i, the value 0 indicates that individual u did not purchase product i, these values are coded as $R_{u,i}$. So each value of $R_{u,i}$ indicates whether user u purchased product i. Now these ratings can be modelled by deriving latent factor vectors for both the users and the products, which are described as p_{uk} and q_{ik} respectively. All of the values of $R_{u,i}$ are modelled by taking the inner product of these vectors. This leads to Equation 1:

$$\hat{R}_{ui} = q_i^t * p_u \tag{1}$$

However the correct values for p_u and q_i are not known. Hence, we need to devise a way to derive these values. In the traditional setting we can calculate the error as seen in Equation 2. By saying that if these values should be equal then the difference of these values should be zero. Any non-zero value from differencing the real value and the estimated value are denoted as the error e_{ui}.

$$e_{ui} = R_{ui} - \hat{R_{ui}} \tag{2}$$

In the previous Equation $\hat{R_{ui}}$ is introduced which

is the predicted value for R_{ui} which is calculated by taking the inner product of the latent factor vectors q_i and p_u. Since we do not have the correct latent factor vectors to begin with, there are most certainly errors. These errors can be used to derive better latent factor values. We can do this by minimizing the errors. If we first square the errors, we ensure that negative and positive errors are both penalized similarly and that larger values are penalized more strongly than smaller errors. This leads to the objective function where we attempt to find the values for the vectors that achieve the lowest Mean Squared Prediction Error (MSPE). In this Equation the objective is to minimize the total sum of the squared errors. The objective function is denoted in Equation 3. In this Equation K indicates the number of latent dimensions.

$$\min_{q,p} \sum_{(u,i)\in R^K} e_{ui}^2 = \min_{q,p} \sum_{(u,i)\in R^K} (R_{ui} - q_i^t p_u)^2 \tag{3}$$

3.2 (Non-Negative) Matrix Factorization

MF as a technique can be specified in different ways. A simple variant that has been thought of is called NMF. The vectors of factors that attempt to characterize the items and users are constrained to be non-negative. A negative factor value would indicate a disinterest in an item and could potentially lead to predicted values that are negative. From a theoretical point of view MF would allow for a better fit due to the latent factor values being allowed a broader range of values, namely also negative values. In some situations, however, negative factor values do not make any sense since disinterest is not possible. A simple example that can be thought of is matching job applicants with vacancies. Vacancies can have certain skills as requisites and candidates either have or do not have these skills. In the situation where we attempt to predict the purchase probability, we do not wish to predict negative values since this would violate the basic probability axioms.

Our objective function is exactly the same as seen in Equation 3. The only thing that differs in NMF is that there are additional constraints in the form that the vectors for both the users p_u and q_i are now non-negative as seen in Equation 4.

$$\min_{q,p} \sum_{(u,i)\in R^K} (R_{ui} - q_i^t p_u)^2, \quad q_i > 0 \quad \& \quad p_u > 0 \tag{4}$$

3.3 Cluster Affiliation Model for Big Networks

The Cluster Affiliation Model for Big Networks (BigCLAM) is described as a probabilistic generative model for graphs that reliably capture

the organization of social networks based on community affiliations, Yang and Leskovec (2013). This probabilistic generative model attempts to find the community affiliations of individuals that most likely generate the network that is given. These social networks exist out of individuals that can be connected to every single other individual. This can be visually represented by taking the individuals in the network as vertices and the connections as edges. If two individuals have a high affiliation to a certain community then the probability that they are connected to each other is high. This means the higher the affiliation these individuals have to this community, the higher the probability that they are connected. A real-world example of this could be two individuals that work at the same workplace. If they work there full-time, they have a higher probability to know each other (i.e., be part of the same community) compared to if they work there part-time. Given that an individual can be part of multiple communities, each community affiliation that the individual has in common with another individual increases the chance that they are connected to each other. For example, if two individuals go to the same gym, they work at the same company and they went to the same university, the probability of them knowing each other is likely higher than if they only went to the same university. In a retail setting this approach is slightly different since we are dealing with individuals and products. A community in this setting is a group of users and products e.g., parents with babies and products within the Baby category. A small additional note in this scenario; a user can only directly be connected to products and products can only directly be connected to users. Users are indirectly connected through products and products are indirectly connected through people. The constraint that is just described can be phrased more easily; the structure is that of a bipartite graph.

In this research the BigCLAM is used to model the probabilities that an individual is going to purchase a product. Given that we attempt to model this as a probability, we need to adhere to the probability axioms. The probability that an individual u is connected to product v can be specified as seen in Equation 5. Where F_u denotes the users affiliation to a latent dimension and F_v the products affiliation to the same latent dimension.

$$p(u, v) = 1 - exp(-p_u * q_i^t) \tag{5}$$

Due to adhering to the probability axioms, the probability that an individual is not connected to a product can simply be calculated by calculating the complement given by $1 - p(u, v)$ as seen in Equation 6. Adhering to the probability axioms also constraints the values of p_u and q_i to be non-negative as seen in Equation 7.

$$1 - p(u, v) = exp(-p_u * q_i^t) \tag{6}$$

$$F_u, F_v > 0 \tag{7}$$

Given that we train on a bipartite graph in which the relations are known (i.e., the purchase history of the customers), the objective function is maximizing the likelihood of the model. This is done by deriving the latent factor vectors that are most likely to have generated the bipartite graph. The objective function can be seen in Equation 8.

$$\underset{F}{\operatorname{argmax}} \prod_{(u,v)\in E} p(u, v) \prod_{(u,v)\notin E} (1 - p(u, v)) \tag{8}$$

Where E comprises the set of purchases (i.e. the edges) in the bipartite graph and everything that is $\notin E$ comprise the non-existent edges. With MF the objective function, as seen in Equation 3, we attempt to minimize the squared errors by finding the optimal latent factor vectors. With BigCLAM the objective function, as seen in Equation 8 we attempt to maximize the log likelihood by finding the optimal community affiliations. Now it becomes clear that if we consider the community affiliations as latent factor vectors where communities are latent dimensions, then the technique can be seen as a variant on NMF, but with a more complex objective function.

3.4 Algorithm Initialization

In order to estimate the models, the algorithms have to be initialized with starting values for the latent factors. Picking bad starting values that are very far from the optimal solution results in a longer time till convergence and in the worst case it might even cause the algorithm to fail. Picking good starting values ensures that the estimation procedure is as fast as possible and converges to the global minimum. We can pick good starting values by making an educated guess. A possible strategy is to pick starting values in such a way that we have the right values on average. The right values on average can be chosen by ensuring that the expected number of purchases correspond to the real amount of purchases in the training data set. With this strategy we avoid the scenario where the gradient is unidirectional which would mean that we could easily have better starting values by simple starting either larger or smaller values. Calculating the expected latent factor values for (N)MF and BigCLAM slightly differ. In both cases we can pick the starting values in such a way that the number of predicted purchases corresponds with the actual amount of purchases in the training data set. This total amount of purchases should be the sum of all predicted values when using (N)MF and the sum of all predicted purchase probabilities when using BigCLAM. This implies that each value that we are

trying to predict should be equal to the average purchase probability p. For (N)MF we end up with the Equation 9 to determine the starting values.

$$p = p_u * q_i^t \tag{9}$$

For BigCLAM we end up with Equation 10 and 11 to determine the starting values. Whenever p is very small, the values can be approximated with Equation 9. A more exact approach however is preferred and appropriate for this training data set.

$$p = 1 - exp^{-p_{uk} * q_{ik}^t} \tag{10}$$

$$ln(1-p) = p_{uk} * q_{ik}^t \tag{11}$$

These equations give us an indication on what the starting values should be on average. We can however not use the same starting values everywhere; the starting values should be these values on average. We can add some random variation by letting the starting values be set in a given range that is centred around these averages. We can also differ the average starting values for the users and for the products. As long as the inner product of the latent factor values lead to the correct expected purchase probability or score. In this research the range for NMF is initially set from zero to two times the average latent factor value. For MF the range is set from to minus two to four times the average latent factor value and for BigCLAM the range is set to a half to one-and-a-half times the average latent factor value. All of these values should take the into account the number of dimensions of the latent factor values. For ((N)MF this the average starting value (asv_{mf}) is calculated by dividing the average purchase probability p by the number of latent dimensions K and taking the square root of this value. The idea behind this approach is that the total value consists out of equal parts for each multiplication in the inner product. Since each multiplication can be seen as taking the square value if the terms in these elements are equal, we can take the square root to find the initial average starting value.

$$asv_{mf} = sqrt(p/K) \tag{12}$$

For BigCLAM we have to take the exponential transformation into consideration. Therefore, the average starting value of BigCLAM $asv_{bigclam}$ is calculated by taking the natural logarithm of 1 minus the average purchasing probability p, dividing this value by the number of latent dimensions K and taking the square root.

$$asv_{bigclam} = sqrt((1 - ln(p))/K) \tag{13}$$

The original approach of using the starting values that would be correct on average was abandoned after the model estimation procedure led to poor local minima being found. This

was easily deducted from the fact that the NMF models had a lower objective function value than the MF models. Since the latter models are nested this should not be the case. Furthermore, starting the BigCLAM models with smaller values led to higher log-likelihoods, but the model estimation procedure increased the log-likelihood instead of lowering it. In order to address these issues, multiple starting ranges have been tested and the following heuristics seemed to work. The starting range for MF was kept positive, the starting range for the products and for the users should differ with the (N)MF models. For the BigCLAM models the starting range for the products and users should be similar. The BigCLAM models should not be started with starting values that are too low. The starting ranges that are used are shown in table 1. The starting values are drawn at random from an uniform distribution for which the lower bound and upper bound are also given in Table 1.

This was detected in the estimated models since the MF models were outperformed by the NMF models.

3.5 (Stochastic) Gradient Descent

In this research SGD is used as the optimization strategy to estimate the models. Gradient Descent is an iterative optimization algorithm that can be used to find the minimum or maximum of a function. This is achieved by making use of the first-order gradient of the function. The optimization algorithm does this by taking (small) steps proportional to the gradient in order to find the minimum. By taking positive steps in the direction of the gradient, which is also known as Gradient Ascent (GA), we attempt to find the maximum of the function. If we take negative steps in the direction of the gradient we attempt to find the minimum of the function. GD typically does this for all values one after another in order to find the minimum or maximum of a function. SGD is similar in the methodology. The thing that differs with SGD is that gradient is calculated based on a randomly selected sample from the rows. An advantage of SGD over GD is that the algorithm usually converges faster, the results in term of optimizing the objective function are however noisier.

The objective function in NMF is to minimize the Mean Squared Prediction Error (MSPE) as seen in Equation 4. Since it is the objective to find the minimum of the function we take negative steps towards the gradient. First, we calculate the first order partial derivatives of the objective function. The gradient to p_u and q_i^t are given by Equations 14 and 15.

$$\frac{\partial e_{u.i}^2}{\partial p_u} = -2 * e_{ui} * q_i \tag{14}$$

$$\frac{\partial e_{u.i}^2}{\partial q_i} = -2 * e_{ui} * p_u \tag{15}$$

Since we are trying to find the minimum of the

Table 1: Algorithm initialization: starting values. Where p indicates the average starting value, U indicates that it is for the starting values of the users' latent dimensions, P indicates that it is for the starting values of the products' latent dimensions, LB indicates the lower bound and UB indicates the upper bound.

Model	# Dimensions	Batch size	U LB	U UB	P LB	P UB	LR
MF	K = 3	10	0	asv_{mf}*1.5	0	p*0.25	$1 * 10^{-4}$
MF	K = 5	10	0	asv_{mf}*1	0	p*0.25	$1 * 10^{-4}$
MF	K = 10	10	0	asv_{mf}*0.25	0	p*0.1	$1 * 10^{-4}$
NMF	K = 3	10	0	asv_{mf}*1.5	0	p*0.25	$1 * 10^{-4}$
NMF	K = 5	10	0	asv_{mf}*1	0	p*0.25	$1 * 10^{-4}$
NMF	K = 10	10	0	asv_{mf}*0.25	0	p*0.1	$1 * 10^{-4}$
BigCLAM	K = 3	10	0	$asv_{bigclam}$*1.5	0	p*1.5	$2 * 10^{-4}$
BigCLAM	K = 5	10	0	$asv_{bigclam}$*1.5	0	p*1.5	$5 * 10^{-4}$
BigCLAM	K = 10	10	0	$asv_{bigclam}$*1.5	0	p*1.5	$5 * 10^{-4}$

objective function we take a negative step in the direction of the gradient. We take the original latent factor vector, we calculate its gradient and update the vector with a (small) step by subtracting the gradient proportional to the step size as seen in Equations 16 and 17.

$$p_{u_{new}} = p_{u_{old}} + \gamma * 2 * e_{ui} * q_i \tag{16}$$

$$q_{i_{new}} = q_{i_{new}} + \gamma * 2 * e_{ui} * p_u \tag{17}$$

Now the parameter γ is introduced as the step size towards the gradient. This step size is formally known as the learning rate and the next section elaborates further upon this. The objective function for NMF has the constraints that the vectors should be non-negative. Whenever elements of the latent factor vectors become negative after updating, we set the value of these negative elements to zero as seen in Equations 18 and 19, where k indicates the element in the vector.

$$p_{uk_{new}} < 0, \quad \rightarrow \; p_{uk_{new}} = 0 \tag{18}$$

$$q_{ik_{new}} < 0, \quad \rightarrow \; q_{ik_{new}} = 0 \tag{19}$$

The objective function of BigCLAM is given by Equation 8. For mathematical and computational convenience, we choose to optimize the

log-likelihood function instead of the likelihood function. Given that the natural logarithm is a monotonically increasing function, the log-likelihoods have the same relations of order as the likelihoods. The log-likelihood is given in Equation 20.

$$\hat{F} = \underset{F \geq 0}{\operatorname{argmax}} \sum_{(u,v) \in E} log(1 - exp(-p_u q_i^t)) - \sum_{(u,i) \notin E} p_u q_i^t \tag{20}$$

Where E indicates the nodes that are connected and $\notin E$ indicates the nodes that are not connected. To find the maximum of the log-likelihood we need to calculate the gradient of the function by taking the partial derivative(s). The gradient in the calculation is given by Equation 21.

$$\Delta l(p_u) = \sum_{i \in N(u)} q_i \frac{exp(-p_u q_i^t)}{1 - exp(-p_u q_i^t)} - \sum_{i \notin N(u)} q_i \tag{21}$$

The second term $\sum_{v \notin N(u)} q_i$ comprises all of the pairs of users and products that are not connected. Since a sparse matrix is involved, this term contains most of the computational burden. In order to lower the computation burden of the algorithm we will calculate the second term $\sum_{v \notin N(u)} q_i$ by Equation 22:

$$\sum_{i \notin N(u)} q_i = (\sum_{i} q_i - p_u - \sum_{i \in N(u)} q_i) \tag{22}$$

Where $i \notin N(u)$ are all the products that a consumer did not purchase and $i \in N(u)$ are all the products that a consumer did purchase. In this research the term p_u is not present since a node does not have a relationship with itself. Therefore, it can be ignored and the equation can be rewritten as seen in Equation 23:

$$\sum_{i \notin N(u)} q_i = (\sum_{v} q_i - \sum_{i \in N(u)} q_i) \tag{23}$$

If we then insert Equation 23 in 21, we end up with the following Equation 24. We can take out the common term $\sum_{v \in N(u)} q_i$ and include the 1 in the fraction. This allows us to decrease the amount of computations that have to take place, this derivation can be seen in Equations 25 and 26.

$$\Delta l(p_u) = \sum_{i \in N(u)} q_i \frac{exp(-p_u q_i^t)}{1 - exp(-p_u q_i^t)} + \sum_{i \in N(u)} q_i - \sum_{v} q_i \tag{24}$$

$$\Delta l(p_u) = \sum_{i \in N(u)} q_i \left(\frac{exp(-p_u q_i^t)}{1 - exp(-p_u q_v^t)} + 1 \right) - \sum_{i} q_i \tag{25}$$

$$\Delta l(p_u) = \sum_{i \in N(u)} \frac{q_i}{1 - exp(-p_u q_i^t)} - \sum_{i} q_i \tag{26}$$

This leads to the partial derivative to the users in the network that we use to estimate the model. Calculating the partial derivative of the products requires exactly the same steps and leads to Equation 27.

$$\Delta l(q_i) = \sum_{u \in N(i)} \frac{p_u}{1 - exp(-p_u q_i^t)} - \sum_u p_u \quad (27)$$

A small detail to add here is that if we look at the gradient we see that the possibility exists that we attempt to divide by zero. This happens when the term $p_u q_i^t$ becomes equal to zero. In order to prevent this a very small value ϵ is added, this addition is shown in Equation 28.

$$\Delta l(q_i) = \sum_{u \in N(i)} \frac{p_u}{1 - exp(-p_u q_i^t) + \epsilon} - \sum_u p_u \quad (28)$$

Now SGD can be used to find the maximum of the function. Compute the gradient $\Delta l(p_u)$ of row u (while keeping others fixed). In other words, we calculate the gradient based on all of the products individual u did or did not purchase. If we want to calculate the gradient of a product $\Delta l(q_i)$ of column i we look at all the users that did or did not purchase this product. In both scenario's, since we are looking for the maximum of the function, we take a positive step in the direction of the gradient.

$$p_{u_{new}} = p_{u_{old}} + \eta * \Delta l(p_u) \quad (29)$$

$$q_{i_{new}} = q_{i_{old}} + \eta * \Delta l(q_i) \quad (30)$$

The η parameter that is added here is again the step size which is the learning rate. Now the learning rate is elaborated upon and it's influence on GD. Furthermore, BigCLAM has the same constraint as NMF that the latent factor vectors are required to be non-negative. Every time after we update the latent factor vectors we check whether the vectors have been negative. If this is the case set the values that are negative to zero, see Equations 31 and 32.

$$p_{u_{new}} < 0, \quad \rightarrow \; p_{u_{new}} = 0 \quad (31)$$

$$q_{i_{new}} < 0, \quad \rightarrow \; q_{i_{new}} = 0 \quad (32)$$

Furthermore, the calculation of the sum in each gradient $\sum_u p_u$ and $\sum_i q_i$requires us to go over all of the purchases which might be computationally intensive depending on the size of your data set. Another trick that can be applied is storing this sum and after updating only add the difference instead of calculating the sum again as shown in Equations 33 and 34.

$$\sum_u p_{u_{new}} = \sum_u (p_{u_{old}}) + \delta p_u \quad (33)$$

$$\sum_i q_{i_{new}} = \sum_i (q_{i_{old}}) + \delta q_i \quad (34)$$

3.6 Learning Rate

The parameters γ and η as seen in Equations 16, 17 and 29, 30 are the learning rate parameters that are added to control the speed of the GD. These parameters ensure that we move to the gradient proportional to the step size. By doing so we can tweak the model estimation procedure. A small learning rate might take a long

time to converge, or end up in a local minimum. A large learning rate might never reach a minimum because of overshooting, or the algorithm might end up in a tail where the MSPE only increases (or log likelihood decreases). In order to ensure that the model does not end in a local minimum 20 different random starts for each model are taken. From these estimated models the model that has the best predictive power is taken and is estimated 80 more times. The estimation procedure of the models is seeded in order to guarantee reproducibility. If the random starts vary strongly in the achieved objective function then this is a positive indication that there are random starts have ended up in a local minimum. Either increase the learning rate to try to avoid these situations or to spend more time in initializing the algorithms with better starting values and to take as many random starts as possible to find the best fit, without forgetting to take overfitting into consideration. Taking this information into consideration and looking at Table 1, these values were chosen for these reasons. The starting values were chosen as small as possible, without negatively effecting the found optimum, to lower the estimation speed. The learning rate was chosen in such a way that the model converged in a reasonable time and simultaneously did not end the estimation procedure prematurely. The learning rate for the BigCLAM models was increased for this exact reason.

3.7 Over-fitting and regularization

There might be a possible problem here in terms of over-fitting the solution to the training data set. This leads to a small bias and a high variance. One can reduce over-fitting by adding regularization terms. These terms introduce some bias by adding a penalty dependent on the size of the parameter values. The larger the parameter values, the larger the penalty. By doing so some of the variance of the solution is reduced by adding a (small) bias and to make sure that the solution does not overfit and achieve also high out-of-sample performance. For MF this is usually achieved by adding the penalty in the objective function as seen in Equation 35.

$$\min_{q^i,p^u} \sum_{(u,i)\in K} (R_{ui} - q_i^t p_u)^2 + \lambda(\|q_i\|^2 + \|p_i\|^2) \quad (35)$$

The regularization term added here with the hyperparameter λ introduces a penalty that is proportional to the squared parameter values. For NMF the same approach can be followed but with the small addition that the vectors should be non-negative. This can be seen in Equation 36.

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (R_{ui} - q_i^t p_u)^2 + \lambda(\|q_i\|^2 + \|p_i\|^2), \quad q_i > 0 \quad \& \quad p_u > 0 \tag{36}$$

Now that the objective function is defined, we have a way to find a solution the problem, namely the optimal values for the vectors.

Gradient to p_u is now given by Equation 37.

$$(2 * e_{ui} * q_i - \lambda * p_u) \tag{37}$$

Gradient to q_i^t is now given by Equation 38.

$$(2 * e_{ui} * p_u - \lambda * q_i) \tag{38}$$

In this research however, a different approach is taken to avoid overfitting. This is done by splitting the data set in several sets. Instead of using the traditional split of a training and a test set also a select set is formed.

3.8 Train, Select and Test

For our model estimation procedure, we want to simultaneously achieve convergence and avoid overfitting. To this end we split our data set in three different data sets. These three data sets are the train, select and test data set. Whilst estimating the models we check the performance on the out-of-sample select data set to avoid overfitting and we stop the estimation process after our performance does not increase anymore on this data set. We then assume that we have achieved convergence. For (N)MF the performance of the model in terms of the objective function, the Mean Squared Prediction Error (MSPE), have to be examined. For BigCLAM the performance of the model in terms of the log-likelihood (LL) is examined. The following graphs show the MSPE and LL respectively during the training. For each iteration the MSPE and LL are plotted and after these do not decrease or increase respectively the training process is halted and the models have converged to an optimal solution without. During the estimation procedure value of the objective functions are plotted. This allows use to visually check if the model did converge, if the learning parameter has to be adjusted and lastly if the code is actually working as intended. Whenever the learning parameter was set to high and the model estimation procedure was failing, the value calculated for the objective function would not decrease ((N)MF) or increase (BigCLAM). Two example plots of the MSPE and log-likelihood are shown in Figures 1 and 2.

3.9 Mini-batching

In the traditional approach of MF the algorithm predicts the value for each user and product combination one by one. The algorithms do this in order to calculate gradient and to then step into the direction that depends on the objec-

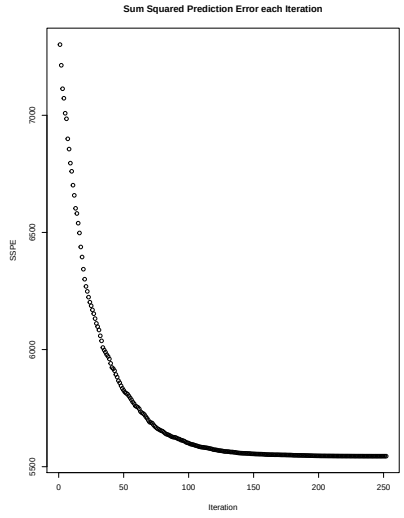

Figure 1: The Sum Squared Prediction Error being calculated each iteration until convergence.

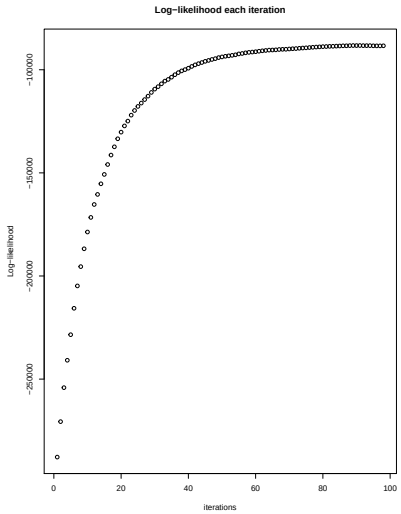

Figure 2: The Log Likelihood being calculated each iteration until convergence.

tive function. If we wish to maximize the function a positive step in the direction of the gradient needs to be taken, to minimize the function a negative step in the direction of the gradient is taken. In this research however for all three methods this method is applied row-wise and column-wise. If the users are denoted as the rows and the products are denoted by the columns then firstly the gradient is calculated based on the purchases and the respective mean gradient. Updates are done with this gradient proportional to the learning rate. Then this is applied for all the non-purchases and the respective mean gradient proportional to the learning rate and the non-purchases. This is repeated for each row and each column. The average derivative of a consumer given all the products and the average derivate of a product given all the users are calculated as seen in Equations 39 and 40 respectively. In BigCLAM this is already done in the original algorithm. Furthermore, the algorithm only goes over the purchases for each iteration since the non-purchases are considered simultaneously in the gradient. Where I is the total number of products and U is the total number of users.

$$\frac{1}{I}\sum_{i=1}^{I}\frac{\partial e_{u.i}^2}{\partial p_u} = \frac{1}{I}\sum_{i=1}^{I} -2 * e_{ui} * q_i \tag{39}$$

$$\frac{1}{U}\sum_{u=1}^{U}\frac{\partial e_{u.i}^2}{\partial q_i} = \frac{1}{U}\sum_{u=1}^{U} -2 * e_{ui} * p_u \tag{40}$$

Stochastic Gradient Descent is then used to estimate the models. A sample of randomly selected consumers or products are selected for which the latent factor vector is updated using the mini-batch method as described before. The optimization algorithm alternates between rows and columns (i.e. consumers and products) until it meets the requirements of the stopping criteria. The reason why Stochastic Gradient Descent is used, is to increase the model estimation speed. With this approach however, the risk exists of updating the latent factor vectors in such a way that it is possible that the update does not move into the right direction of the objective function. By mini-batching the average gradient is calculated for a consumer or a product. By averaging the gradient, the descent is smoothed out to avoid the possibility of stopping the model estimation procedure to quickly. This is also further checked by using multiple random starts for each model that is estimated. The batch size used in this research is 10. By choosing a smaller batch size the gradient becomes nosier. This is done as an additional measure to avoid local minima.

3.10 Performance Measures

The performance of the models is investigated by checking whether the top N predicted products in terms of score or probability were actually purchased in the test data set (the True Positives). There the precision of the algorithms is calculated. The precision for values of N being 1, 3, 5 and 10 are calculated. In order to benchmark the quality of the predictions the top 10 products sold are calculated and are used for the benchmark predictions. If the algorithms in this research do not beat the benchmark this would imply that showing the products that are sold the most would beat the algorithms. This measure is used with two different weighting schemes. The first scheme values each correctly predicted product with an equal weight. To this end the precision is calculated as seen in Equation 41.

$$Precision = \frac{TP}{TP + FP} \tag{41}$$

The second scheme values the products in an ordered fashion by scoring the correct predictions with a higher probability or score higher (within the subset). The product with the highest probability or score, if bought, is scored with N points. The last predicted product (in the consideration set of N) is scored with 1 point.

$$Scored\ \ Precision = \sum_{i=0}^{N-1} (N-i)^{I[TP]} \tag{42}$$

By using this measure, as seen in Equation 42 some insight is generated into the quality of the predictions that the algorithms provide.

3.11 Interpreting Latent Factor Dimensions

After estimating the models and therefore having calculated the optimal latent factor vectors in terms of our objective function, we want to understand what the latent factor vectors could entail. The strategy we use here is to examine the top N products with the highest values within a latent dimension for our NMF and BigCLAM models. For our MF models we can also examine the bottom N products with the lowest values for the same latent dimension.

In order to analyse these top N products we look at the product names of these products at use these to create a corpus. We then aggregate this corpus by keeping track of the frequency of each word that we find within. This is then used as our data set to generate a wordcloud. A worldcloud is a visual representation that can visualize text data. The technique achieves this by centering the words that are most frequent and the adding less frequent words, based on their frequency, outwards in a circular fashion. The size of the font used for each word is positively correlated with the frequency and the frequency of a word has its own colour.

We can then use this wordcloud to interpret the latent dimension. If the words inside the wordcloud contain terms such as fruit, vegetables, organic and names of fruit and vegetables the latent dimension could for example represent produce. If this is the wordcloud formed for the top N products of a latent factor dimension of a MF model we could also form the "opposite" wordcloud for the bottom N products (with negative latent dimension values). If this worldcloud would contain words such as fat, grease, snack, hamburger and other sorts of fast food, then the interpretation of this latent dimension might be something as healthiness.

4 Data

The models are applied to a sample of a data set that is made available anonymously by a company that operates as a grocery delivery service. The data set provided contains detailed information on the customers' orders over time. It contains a sample of over 3 million grocery orders from more than 200.000 customers. For each customer the data set contains between 4 and 100 of their orders. The products are split in 21 departments and 134 aisles. The departments range from Frozen, Bakery, Alcohol to Babies, Snacks and Missing. The aisles range from Prepared Soups Salads and Specialty Cheeses to Energy Granola Bars, Packages Meat and Pasta Sauce.

The data set is chronologically split into two separate parts. The first part is used to estimate models with, the second part is used to test models with. A sample of the complete data set is taken in order to ease the workload of es-

timating the models, roughly top 1% of users are taken in terms of number of different products purchased. The products that these users purchased are also the only products that are considered. Our sampling process results in an uneven distribution of products over categories. The top 5 departments in terms of most number of products are Produce, Dairy Eggs, Pantry, Snacks and Frozen. The bottom 5 departments are Missing, Personal Care, Household, International and Babies. Having an uneven distribution of products over departments makes interpretation of the latent factor dimensions more difficult. The remaining departments and the number of product within a department are reported in Table 2.

The users and products that remain are also the only users that are kept in the test data set. The rest of the users and product are excluded. The test data set is split in two equal parts. The first half is randomly selected from the test set as the select set and the rest of the data is used as the test set. In Table 3 the data sets that are created are summarized in terms of number of customers, unique products and number of product purchases.

For the performance measure(s) the purchases probabilities of the top 10 sold products are required. These purchase probabilities are used as a benchmark for the algorithms. Table 4 shows the top 10 products sold and their purchase probability in the training data set.
In this research the top 1% customers are taken that had the most number of different products purchased. The average purchase probability of the products in this set are very high compared to the complete data set. Due to not randomly picking 1% of the consumers, the consumers in the training data set or not a fair representation of the complete data set. Therefore, it is opted to compare the performance of these products as predictions instead of comparing with the baseline purchase probability.

A small note on the structuring of the website that might give some additional insight in our data set. On the website of the retailer that provided the data set, the products are grouped by department. Whenever a customer visits a department page, other products that are listed in the same department are shown. This might cause people to add these products to their consideration set.

Lastly, the test set is analysed and reported in terms of the top 30 products with the highest purchase probability. The probabilities shown in Table 5 indicate how many percent of the consumers in the test set purchased a given item. This will be further discussed in the next chapter.

Table 2: The departments and the number of products within the department of the sample of the complete data set.

	ID	Department	# Products	ID	Department	# Products
1	1	Frozen	95	14	Breakfast	39
2	3	Bakery	49	15	Canned Goods	61
3	4	Produce	292	16	Dairy Eggs	182
4	6	International	22	17	Household	21
5	7	Beverages	62	18	Babies	24
6	9	Dry Goods Pasta	52	19	Snacks	131
7	11	Personal Care	5	20	Deli	26
8	12	Meat Seafood	39	21	Missing	1
9	13	Pantry	144			

Table 3: Characteristics of the subsets of the original and sample data sets.

Subset	Customers	Unique Products	Purchases	Avg # Purchases/Customer
Full training data	206209	49677	13307953	≈ 65
Full test data	131209	39123	1384617	≈ 11
Sample Training data	2062	1239	163488	≈ 79
Sample Select data	632	1109	6389	≈ 10
Sample Test data	632	1123	6666	≈ 6

5 Results

This section shows the results of this research. First, the model selection procedure is elaborated, and the possible presence of local minima is investigated. Next the top performing models, based on this selection procedure, are further examined in terms of their predictive performance and their capability in finding the optimal latent factor vectors. Lastly, the latent factor dimensions are analysed and interpreted by forming wordclouds based on the product names of the products that score the high within a latent dimension.

Table 4: The top 10 products sold in the training data set.

Rank	Product ID	Product Name	Purchase Prob.
1	23516	Organic Unbleached All-Purpose Flour	0.2047
2	19706	Organic Nectarine	0.1993
3	1025	Lasagna Pans	0.1978
4	12845	Chopped Walnuts	0.1954
5	3464	Curried Lentils	0.1916
6	9092	Girl Scouts Thin Mint Cereal	0.1751
7	4421	Baby Bee Nourishing Baby Oil	0.1712
8	5134	Cilantro Bunch	0.1712
9	11408	Crushed Tomatoes	0.1712
10	12409	Organic Distilled White Vinegar	0.1702

5.1 Local Minima & Model Selection

The model selection is done by investigating the objective function value of the estimated model on the training data set. For the (N)MF models this is the sum of squared predictions errors (SSPE) and for the BigCLAM models this is the log-likelihood. These values are calculated on the training data set during the estimation period and the lowest SSPE and the highest LL indicate the best fit for the model. Since the models can be fit virtually endlessly on the training data set, which would lead to overfitting, the model estimation procedure was stopped whenever the objective function value worsened on the select set. This is the case when the performance on the in-sample error would decrease whilst the out-of-sample error would increase. After having estimated the models, the model with the best objective function value on the train data set is taken of each model type and the results of these models in terms of precision and scoring is reported. For (N)MF this is the model with the lowest SSPE and for BigCLAM this is the model with the highest log-likelihood value.

In order to examine the sensitivity of the estimated models in terms of local minima, the objective function value on the train data set after estimation are plotted for each model type and number of latent dimensions. These plots can be seen in Figure 3. For the BigCLAM, models the goal of the objective functions is to find the maximum value. This implies that the

%	Product Name	%	Product Name	%	Product Name
20.57%	Banana	7.75%	Organic Whole Milk	5.38%	Organic Whole String Cheese
18.99%	Bag of Organic Bananas	7.28%	Organic Garlic	5.38%	Seedless Red Grapes
17.09%	Organic Strawberries	6.80%	Organic Cilantro	5.22%	Organic Baby Arugula
13.29%	Organic Baby Spinach	6.65%	Strawberries	4.91%	Organic Romaine
11.23%	Organic Hass Avocado	6.33%	Organic Yellow Onion	4.91%	Honeycrisp Apple
9.81%	Organic Raspberries	6.33%	Organic Blackberries	4.75%	Organic Garnet Sweet Potato (Yam)
9.02%	Large Lemon	6.17%	Organic Zucchini	4.43%	Organic Large Extra Fancy Fuji Apple
8.70%	Organic Avocado	6.01%	Organic Grape Tomatoes	4.43%	Organic Blueberries
8.39%	Limes	6.01%	Organic Lemon	4.43%	Organic Granny Smith Apple
8.39%	Organic Cucumber	5.70%	Asparagus	4.27%	Organic Red Bell Pepper

Table 5: Purchase probabilities of the top 30 products in the test set. The column denoted as "%" shows the purchase probability of the product to the right of it.

models with a higher log-likelihood have a better fit on the training data set. The objective function of the (N)MF models are to minimize the SSPE on the training data set, so the lower the value the better. Analysing the plots we are looking for large vertical jumps and/or spread in order to see if there are clear local optimal solutions for the algorithms.

First, the plots of the objective function values on the train data set of the BigCLAM models are investigated. The log-likelihood values of the estimated BigCLAM models with 3 latent dimensions range from roughly -615,000 to roughly 650,000. The highest log-likelihood is around -615,000 followed by values ranging from 640,000 to 650,000. The log-likelihood values of the estimated BigCLAM models with 5 latent dimensions range from -640,000 to -650,000. These values of the estimated BigCLAM models with 10 latent dimensions range from -670,000 to 645,000. The lowest values being around roughly -670,000, -658,000, 655,000, which are then followed by values ranging from roughly 649,000 to 646,000.

Then the objective function values of the NMF models are investigated. The SSPE of the estimated NMF models with 3 latent dimensions range from roughly 142,100 to 142600. For the NMF models with 5 latent dimensions these values range from roughly 139,000 to 140,000. And for the NMF models with 10 latent dimensions range from roughly 141,200 to roughly 141,500.

Lastly, the objective function values of the MF models are investigated. The SSPE of the estimated MF models with 3 latent dimensions range from roughly 142,100 to 142,600. These values for the MF models with 5 latent dimensions range from roughly 139,200 to 140,000. And for the MF models with 10 latent dimensions these values range from roughly 137,300 to 138,900.

Taking the objective function values in consideration, the number of latent dimensions that are selected are 3 latent dimensions for the BigCLAM models, 5 latent dimensions for the NMF models and 10 latent dimensions for the MF models. Having decided on the number of latent dimensions for each model type in the previous section, the model selection is continued by taking the estimated model with the lowest objective function value on the train data set, given the random start with which it was initialized.

5.2 Evaluating Predictions - Precision

The models are evaluated based on how many of the products that were predicted by the model were in fact purchased by the consumer. These results are reported for the benchmark and for each model type and the number of latent dimensions the models were estimated with. The top 1, top 3, top 5 and top 10 predictions are generated for the test data set and the percentage of correctly predicted products are shown. The results are shown in Table 6.

Examining this table the highest performing model in terms of precision overall is the BigCLAM model with 3 latent dimensions. When the precisions of the top 1, top 3, top 5 and top 10 it has the highest value of 68.78% compared to the MF and NMF models (58.77% and 51,14% respectively). Examining the table further in terms of precision overall, the worst performing model is the NMF model with 5 latent dimensions. Taking the sum of the precisions of the top 1, top 3, top 5 and top 10 the total value is 51.14%. Comparing the results of the benchmark to the models, it shows that the heuristic that was chosen as a benchmark has very poor results with a total sum of correctly predicted products of only 1.92%. Examining the number of latent dimensions of each model type shows that the BigCLAM model performs better on this data set with 3 latent dimensions. The NMF model with the lowest objective function value has 5 latent dimensions and the MF model with the lowest objective function has 10 latent dimensions. The objective function value on the train data set in terms SSPE is lowest for the MF model at 137,329 and highest for the BigCLAM model 159,972. The objective function value on the test data set in terms of SSPE is lowest for the NFM model at 4,138 and highest for the BigCLAM model at 11,551.

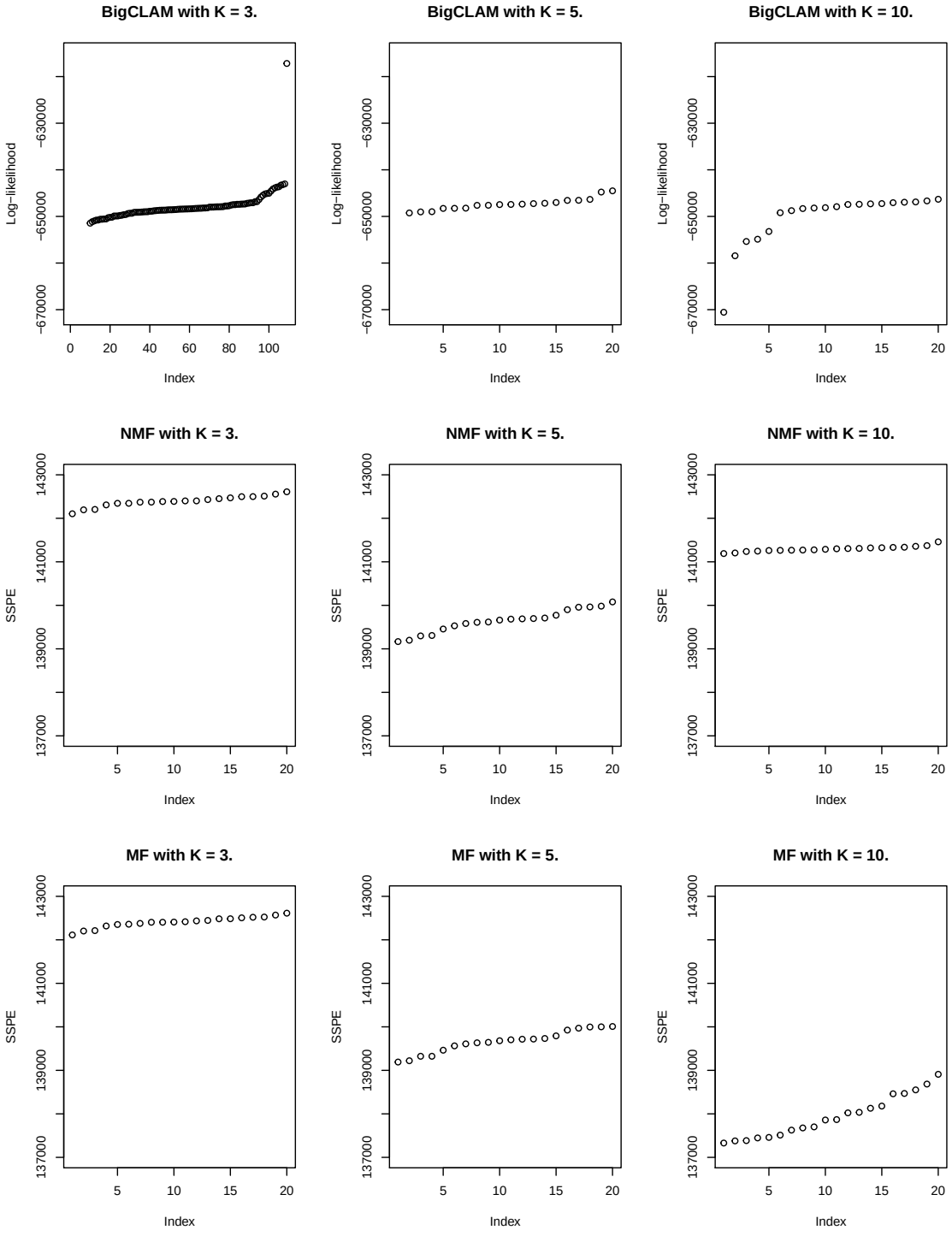

Figure 3: For each model the objective function value after estimation of each random start is plotted. These plots give insight in the stability of the local minima found. The objective values are sorted from low to high.

Table 6: The best results of the estimated models in terms of predictions. The percentages of the top 1, top 3, top 5 and top 10 correctly predicted products are shown. The SSPE for the select, and train data set are shown (S SSPE and T SSPE) and for the BigCLAM models the log-likelihood is reported. Lastly, the total sum of correctly predictions products is shown for each model.

Model	K	Top1	Top3	Top5	Top10	Sum	T SSPE	S SSPE	Steps	Time	LL Train	LL Select
Benchmark		0.30%	0.47%	0.50%	0.65%	1.92%						
BC	3	22.94%	15.45%	14.43%	10.95%	63.78%	159972	5085	79	10.9	-617185	-81105
MF	10	17.25%	15.66%	13.96%	11.90%	58.77%	137329	4641	1000	1.8		
NMF	5	14.87%	13.98%	11.80%	10.49%	51.14%	139166	4138	54	4.3		

5.3 Evaluating Predictions - Scored

The models are also evaluated based on the predictions they generated on the test data set in a weighted fashion. The higher the predicted product is in the ordering, the higher the score when predicted correctly. The results are reported for the benchmark and for each model type and the number of latent dimensions the models are estimated with. The top 1, top 3, top 5 and top 10 predictions are the generated predictions that are scored based on their ordering. The results are shown in Table 7.

Examining this table the highest performing model in terms of score overall is the BigCLAM model with 3 latent dimensions. When the scores of the top 1, top 3, top 5 and top 10 are summed it has the highest value of 6,897 compared to the MF and NMF models (6,788 and 5,843 respectively). Examining the table further in terms of score overall, the worst performing model is the NMF model with 5 latent dimensions. Taking the sum of the scores of the top 1, top 3, top 5 and top 10 the total value is 5843. Comparing the results of the benchmark to the models, it shows that the heuristic that was chosen as a benchmark has very poor results with a total sum of scores of only 247.

5.4 Prediction Functions and Latent Factor Vectors

In order to compare the models in terms of effectiveness of their prediction function and in finding the optimal latent factor vectors, the estimated latent factor vectors on the training data set are used in combination with the prediction function of another type of model. Since the BigCLAM and NMF models both have the constraint that the latent factor vectors should be

Table 7: The models evaluated on their scored predictions. The predictions are scored in a decreasing order from N to 1, where N is the number of productions in the consideration set.

Model	K	Top1	Top3	Top5	Top10	Sum
Benchmark		2	13	40	192	247
BC	3	145	644	1486	4622	6897
MF	10	109	606	1410	4663	6788
NMF	5	94	538	1232	3979	5843

non-negative the estimated latent factor vectors of these models can be used with each other's prediction function. The latent factor vectors are estimated using the training data set and their predictive performance is evaluated on the test data set. The results are shown in Tables 8 and 9.

Examining the results, it becomes visible that the NMF model with the BigCLAM latent factors has the highest overall precision, the summed precision totals to 63.88%. Unexpectedly this combination has a higher total precision than the BigCLAM model with the BigCLAM latent factors. This model and latent factor combination has a summed precision of 63.78%. Further examining the results it also becomes visible that the BigCLAM model with the NMF latent factors has the lowest overall precision, the summed precision totals to 50.26%. This is lower than the NMF model with the NMF latent factors since this combination has a summed precision of 51.14%. Comparing the log-likelihood of the BigCLAM models it becomes clear that the log-likelihood is lowest on both the test and train data set with the BigCLAM latent factor vectors. The lowest SSPE on the train data set is found when using the NMF prediction function and the latent factor values estimated with the BigCLAM model. The highest SSPE on the training data is found when using the BigCLAM prediction function with the latent factor vectors estimated with the NMF. On the select data set the lowest SSPE is when using the NMF prediction function with the NMF latent factors and the highest is found when using the prediction function and the latent factor vectors of the BigCLAM model.

Examining the results, it becomes visible that the BigCLAM model with the BigCLAM latent factors has the highest summed score, the summed score totals to 6977. This is followed by the NMF model with the BigCLAM latent factors

Table 8: The precision of the Top 1, Top 3, Top 5 and Top 10 predictions where LF indicates the latent factors, F indicates the prediction function and sum indicates the sum of the precisions.

F	LF	K	Top1	Top3	Top5	Top10	Sum	Train SSPE	Select SSPE	LL Train	LL Select
Benchmark			0.30%	0.47%	0.50%	0.65%	1.92%				
NMF	BC	3	22.94%	15.51%	14.46%	10.97%	63.88%	139150	4220		
BC	BC	3	22.94%	15.45%	14.43%	10.95%	63.78%	159972	5085	-617185	-81105
BC	NMF	5	13.13%	12.55%	13.23%	11.34%	50.26%	165757	4803	-646351	-81783
NMF	NMF	5	14.87%	13.98%	11.80%	10.49%	51.14%	139166	4138		

Table 9: The precision of the Top 1, Top 3, Top 5 and Top 10 predictions where LF indicates the latent factors, F indicates the prediction function and sum indicates the sum of the scores.

F	LF	K	Top1	Top3	Top5	Top10
Benchmark			2	13	40	192
NMF	BC	3	145	644	1486	4622
BC	BC	3	103	599	1491	4784
BC	NMF	5	81	481	1159	3922
NMF	NMF	5	94	538	1232	3979

which has a summed score of 6897. Further examining the results it also becomes visible that the BigCLAM model with the NMF latent factors has the lowest summed score, the summed score totals to 5643. This is lower than the NMF model with the NMF latent factors since this combination has a summed precision of 5843.

Comparing the results of the NMF models compared to the earlier results the model with 3 latent dimensions outperforms its counterpart with 5 latent dimensions. This is visible in the summed precision, the summed score and the lower objective function on both the training and select data sets. The determining factor here is caused by the latent factor vectors of the BigCLAM that are estimated on the training data set. The difference caused by the predictive functions of the models is a lot smaller in contrast. It would seem that the BigCLAM model has done a better job at finding the optimal latent factor vectors than the Non-Negative Matrix factorization model.

5.5 Estimation Speed

The results in terms of estimation speed are shown in Table 10. This table shows the average number of steps in the estimation procedure, the estimation time and the average time per step for each model type and each variation in number of latent dimensions. The average time is reported in minutes and the average time per step is reported in seconds.

The top 3 fastest models are the BigCLAM model with 3 latent dimensions, followed by the NMF model with 5 latent dimensions and lastly the BigCLAM model with 5 latent dimensions (7.67m, 8.58m and 9.24m). The bottom 3 slowest models are the MF model with 10 latent dimensions, the NMF model with 10 latent dimensions and the NMF model with 3 latent dimensions (105m, 58.64m and 19.66m). The 3 models with the lowest number of steps are the BigCLAM model with 5, with 3 and with 10 latent dimensions (54.85m, 61,84m, 64.85m). The 3 models with the highest number of steps are the are the MF model with 10 latent dimensions, the NMF model with 10 latent dimensions and the NMF model with 3 latent dimensions (879.40, 500.00, 373.00). The 3 models with the lowest average time per step are the MF model with 3 latent dimensions, the NMF model with 3 latent dimensions and the NMF model with 5 latent dimensions (2.76s, 3.16s and 3.82s). The 3 models with the highest average time per step are the BigCLAM models with 10, 5 and 3 latent dimensions (17.46s, 10.11s and 7.44s).

The increase in average time per step is generally lower than the factor increases in the number of dimensions. The only exception is the NMF model with 5 latent dimensions. Given that the models have a fixed set of computations that are required independent of the num-

Table 10: For each model this table shows the number of steps required on average, the time required on average and the average time per step required for each model type and number of latent dimensions used. Furthermore, the increase in factor is denoted for the increase in dimensions and the increase in average time per step.

Model	K	Avg # steps	Avg Time	Avg time/step	Factor in dim.	Factor in time
BC	3	61.84	7.67	7.44	1.00	1.00
BC	5	54.85	9.24	10.11	1.67	1.36
BC	10	64.85	18.98	17.56	3.33	2.36
NMF	3	373.00	19.66	3.16	1.00	1.00
NMF	5	134.90	8.58	3.82	1.67	1.21
NMF	10	500.00	58.64	7.04	3.33	2.23
MF	3	286.95	13.21	2.76	1.00	1.00
MF	5	145.15	11.67	4.82	1.67	1.75
MF	10	879.40	105	7.18	3.33	2.60

ber of latent dimensions this makes sense. The average time per step exists out of a fixed set of computations and partly from computations dependent on the number of latent dimensions.

5.6 Wordclouds of the Latent Dimensions

For each model the latent dimensions are used to generate wordclouds in an attempt to interpret the latent dimensions. A wordcloud is generating by converting a body of text to a bag of words that contain the words and their frequencies. By sorting the words based on their frequencies from high to low an idea about the topic or underlying theme of the body of text can be found. Here the wordclouds that are generated are based on the product names of the 62 products that have the highest values within the latent dimension. The cut-off point of 62 products is chosen since it is roughly 5% of all the products in the sample used here. The product names are split in their separate words and these words are counted. The same process is done for the complete list of products in the sample used. The words and their frequencies from the complete sample set are used to scale the frequencies in the wordcloud with. This prevents terms such as "organic" to dominate every wordcloud since it occurs with virtually every single fruit and vegetable in the data set. An example of this type of visualization can be seen in Figure 4.

Dimension 1		Dimension 2		Dimension 3	
word	freq	word	freq	word	freq
cilantro	1,00	cilantro	1,00	bananas	1,00
bananas	1,00	asparagus	1,00	kirby	1,00
cluster	1,00	bananas	1,00	radish	1,00
anjou	1,00	cluster	1,00	feta	1,00
bunched	1,00	kirby	1,00	bell	0,67
michigan	1,00	anjou	1,00	honeycrisp	0,67
radish	1,00	bunched	1,00	fuji	0,67
dinosaur	1,00	dinosaur	1,00	avocado	0,50
blueberries	0,75	bell	0,50	hass	0,50
limes	0,50	limes	0,50	blueberries	0,50
zucchini	0,50	garnet	0,50	limes	0,50
asparagus	0,50	yam	0,50	zucchini	0,50
arugula	0,50	zucchini	0,50	blackberries	0,50
granny	0,50	blackberries	0,50	gala	0,50
smith	0,50	arugula	0,50	asparagus	0,50

Table 11: For each dimension of the estimated BigCLAM model the normalized word frequencies are shown. The frequencies are normalized by dividing each word by the total number of occurrences in the product data set sample used here.

Figure 4: A worldcloud based on the first latent dimension of the BigCLAM model with 3 latent dimensions.

Since wordclouds are a nice visualization to give a rough idea but difficult to analyse precisely, the top 15 words with the highest frequencies are also stored in tables. These tables 11 and 12 contain the scaled word frequencies of the top 5% products within each latent dimension of the BigCLAM model and NMF model respectively. The words between dimensions overlap a large amount and the interpretation of the dimensions by themselves is rather difficult. The dimensions of the BigCLAM model and the NMF model do not consist out of a specific product group such as fruit, vegetables or dairy. Nor do they exist out of a type of products such as unhealthy, low sugar or fat free. The first two dimensions of the BigCLAM model look quite similar to each other whilst the third dimension is somewhat different. Even though interpretation of the dimensions is difficult it is safe to say that structure within these dimensions work together in order to predict the right products that a customer will buy in the future. Calculating the word frequencies and generating the wordclouds the same way for the latent dimensions of the MF model generated similar results.

In order to examine the latent dimensions further the top 15 sold products in the test set are compared with the worldcloud. These top 15 products are shown in Table 5. Examining this table shows that the top sold products are also found within the wordclouds and word frequency tables. Products such as "Banana", "Bag of Organic Bananas", " Organic Cilantro" and "Organic Blueberries" are examples of this.

6 Conclusion

The starting values with which MF and NMF models are initialized can positively impact the fit that is found. The results show that for this data set, predictive performance did not increase when allowing latent factor values to become negative. There is some spread between the estimated models, but the models did find similar fits to each other and no clear local minimum was found. For the BigCLAM models that were estimated one of the models with 3 latent dimensions had the best fit that different greatly from the rest of the models that were estimated. This however did not reflect in the SSPE on the select data set nor in the predictive performance of the estimated models. Furthermore,

Dimension 1		Dimension 2		Dimension 3		Dimension 4		Dimension 5	
word	freq	word	freq	word	freq	word	freq	word	freq
cilantro	1,00	bananas	1,00	feta	1,00	cilantro	1,00	cilantro	1,00
michigan	1,00	cluster	1,00	bananas	1,00	cluster	1,00	asparagus	1,00
bananas	1,00	feta	1,00	anjou	1,00	anjou	1,00	cluster	1,00
bunched	1,00	anjou	1,00	bunched	1,00	kirby	1,00	cantaloupe	1,00
radish	1,00	radish	1,00	dinosaur	1,00	michigan	1,00	kirby	1,00
raspberries	0,67	hothouse	1,00	michigan	1,00	bunched	1,00	bunched	1,00
hass	0,50	panko	1,00	kirby	1,00	cantaloupe	1,00	arugula	0,50
zucchini	0,50	avocado	0,50	towels	1,00	bananas	1,00	blackberries	0,50
limes	0,50	blueberries	0,50	raspberries	0,67	bell	0,67	zucchini	0,50
gala	0,50	hass	0,50	fuji	0,67	grape	0,67	granny	0,50
granny	0,50	limes	0,50	honeycrisp	0,67	honeycrisp	0,67	smith	0,50
smith	0,50	arugula	0,50	blueberries	0,50	cabbage	0,50	garnet	0,50
heavy	0,50	asparagus	0,50	limes	0,50	zucchini	0,50	yam	0,50
crown	0,50	cilantro	0,50	zucchini	0,50	asparagus	0,50	limes	0,50
asparagus	0,50	zucchini	0,50	asparagus	0,50	roma	0,50	dill	0,50

Table 12: For each dimension of the estimated NMF model the normalized word frequencies are shown. The frequencies are normalized by dividing each word by the total number of occurrences in the product data set sample used here.

the BigCLAM model with 10 latent dimensions had several models that had a relatively poor fit compared to the others. Based on the results here the terms the BigCLAM model seems to be more sensitive to local minima than the (N)MF models.

Comparing the models based on their predictive performance it seems that the BigCLAM model outperformed the other models both when comparing based on the precision and the scoring metric. The performance of the BigCLAM models seem to be better with fewer latent dimensions and the performance of the (N)MF models seem to improve when increasing the number of latent dimensions. The results show that this is most likely due to the fact that the BigCLAM model is more effective in finding the right latent dimension factor vectors compared to the other models. This can be concluded since when switching the prediction function and the latent dimensions of the BigCLAM and NMF models the determining factor in performance were the latent dimensions vectors that were estimated by the BigCLAM model and not the prediction functions of the models. The predictive performance of the latent dimension vectors estimated by the BigCLAM model even improved when using the prediction function of the NMF model here, but the increase was rather small, and this could also purely be random.

Comparing the models based on their estimation speed the BigCLAM model tends to be the fastest model type to estimate. Even though the average time per step is higher, the model converges a lot faster in terms of the required steps compared to the (N)MF. The average estimation time per step being higher for the BigCLAM model can be explained by the more complex objective function and corresponding derivatives that have to be calculated. Furthermore, the BigCLAM model tends to perform better with a lower number of latent dimensions and the other models tend to perform better with a higher number of latent dimensions. This would also provide an advantage in terms of estimation speed since the model estimation procedure increases with the number of latent dimensions.

Interpreting the latent dimensions of the estimated models did not yield clearly interpretable results in terms of underlying theme or topic. The latent dimensions estimated seem to be high latent factor values for products that were purchased often in the test set. A possible explanation could be an underlying theme or topic that the model can find but that we do not understand. Another possible explanation could be that the models are adapt at finding the right latent dimensions that model the purchases within the train set without finding an underlying theme or topic.

All in all, the results of the BigCLAM model

show promising results. The alternative objective function outperforms (N)MF in terms predictive performance, estimation speed on this data set, when used as a recommender system, and required a lower number of latent dimensions.

7 Discussion and Future Research

In this thesis there we some points that could be improved, a few examples will be given. During model estimation procedure of the NMF models the maximum number of steps was unfortunately limited to 500 when estimating the model with 5 and 10 latent dimensions. The maximum number of steps for the MF model was limited to 1000. This most likely negatively impacted the estimation procedure by stopping it prematurely. In other words, the estimation procedure was halted before the model converged. For future research the number of steps should not be limited when using the same stopping criterium that was used in this thesis.

The objective function value of the NMF model with 10 latent dimensions was relatively high compared to the MF counterpart. The values should have been similar and possibly better. Furthermore, the results of the BigCLAM models are an indication of problems revolving local minima. There was a single estimated model with a very high log-likelihood value found for the BigCLAM model with 3 latent dimensions compared to the rest of the estimated BigCLAM models. This estimated model however did not correspond with conclusive better predictive performance than the rest of the estimated models. The BigCLAM models did not increase in likelihood whilst increasing the number of latent dimensions. This is unexpected since the models are nested and therefore this is another indication of problems with local minima. For future research a larger data set is advised and more time should be spent on finding appriopiate starting values.

The results in this paper with respect to the performance of the models only gives insight in their performance in correctly predicting reorders. Furthermore, the frequency of the purchases of a certain product and time between purchases was not taken into account. Given the limited computing power and time constraint the sample size was relatively small, being only 1% of the complete data set, and the number of random starts was rather small. Increasing the sample size and the number of random starts will likely yield, more complete, insightful and indisputable results. Furthermore, the process of choosing this 1% would have been more insightful if the sampling process was random instead of the methodology used here. This should be implemented in future research.

Lastly the BigCLAM model was specified to estimate probabilities the model makes use of the fact that the complement of a probability is known and therefore only goes over the purchases in the matrix on which the model is estimated. Doing this for lowers the estimation speed especially for sparse matrices. This is also however possible for (N)MF and would most likely also improve the estimation speed of these models. This is recommended for future research.

The BigCLAM model seems to be an interesting and promising alternative to NMF as a recommender system, but more research is still required for more definitive findings. For future research the methodology could be repeated on a larger sample and different data sets. Comparing the BigCLAM model with other models than used here should yield more insights in the (dis)advantages.

References

[1] A. S. Das, M. Datar, A. Garg, and S. Rajaram, Google news personalization: scalable online collaborative filtering, Proceedings of the 16th International Conference on World Wide Web (WWW '07) (New York), ACM Press, 2007, pp. 271–280.

[2] A. Felfernig, E. Teppan, G. Leitner, R. Melcher, B. Gula, and M. Maier, Persuasion in knowledge-based recommendation, Proceedings of the 2nd International Conference on Persuasive Technologies (Persuasive '08) (Oulu, Finland), vol. 5033, Springer, 2008c, pp. 71–82.

[3] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Advances in neural information processing systems. 2002.

[4] Bollen, Dirk, et al. "Understanding choice overload in recommender systems." Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010.

[5] Burke, Robin. "Integrating knowledge-based and collaborative-filtering recommender systems." Proceedings of the Workshop on AI and Electronic Commerce. 1999.

[6] C.-N. Hsu, H.-H. Chung, and H.-S. Huang, Mining skewed and sparse transaction data for personalized shopping recommendation, Machine Learning 57 (2004), no. 1–2, 35–59.

[7] D. Cosley, S. Lam, I. Albert, J. Konstan, and J. Riedl, Is seeing believing? How recommender system interfaces affect users' opinions, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03) (Fort Lauderdale, FL), 2003, pp. 585–592.

[8] G. Adomavicius and Y. O. Kwon, New recommendation techniques for multicriteria rating systems, Intelligent Systems, IEEE 22 (2007), no. 3, 48–55.

[9] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Transactions on Information Systems 23 (2005), no. 1, 103–145.

[10] G. Carenini and R. Sharma, Exploring more realistic evaluation measures for collaborative filtering, Proceedings of the 19th National Conference on Artificial Intelligence (AAAI) (San Jose, CA), AAAI Press, 2004, pp. 749–754.

[11] Hill, Will, et al. "Recommending and evaluating choices in a virtual community of use." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press/Addison-Wesley Publishing Co., 1995.

[12] Hoyer, Patrik O. "Non-Negative matrix factorization with sparseness constraints." Journal of machine learning research 5.Nov (2004): 1457-1469.

[13] Jacobs, Bruno JD, Bas Donkers, and Dennis Fok. "Model-based purchase predictions for large assortments." Marketing Science 35.3 (2016): 389-404.

[14] Jacobs, Bruno. Marketing Analytics for High-Dimensional Assortments. No. EPS-2017-445-MKT. 2017.

[15] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: A constant time collaborative filtering algorithm, Information Retrieval 4 (2001), no. 2, 133–151.

[16] K. Hegelich and D. Jannach, Effectiveness of different recommender algorithms in the mobile internet: A case study, Proceedings of the 7th Workshop on Intelligent Techniques forWeb Personalization and Recommender Systems (ITWP) at IJCAI '09 (Pasadena, CA), 2009, pp. 41–50.

[17] Klema, Virginia, and Alan Laub. "The singular value decomposition: Its computation and some applications." IEEE Transactions on automatic control 25.2 (1980): 164-176.

[18] Knijnenburg, Bart P., et al. "Explaining the user experience of recommender systems." User Modeling and User-Adapted Interaction 22.4-5 (2012): 441-504.

[19] Knijnenburg, Bart P., Martijn C. Willemsen, and Stefan Hirtbach. "Receiving recommendations and providing feedback: The user-experience of a recommender system." International Conference on Electronic Commerce and Web Technologies. Springer, Berlin, Heidelberg, 2010.

[20] Knijnenburg, Bart P., Niels JM Reijmer, and Martijn C. Willemsen. "Each to his own: how different users call for different interaction methods in recommender systems." Proceedings of the fifth ACM conference on Recommender systems. ACM, 2011.

[21] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.

[22] Lee, Daniel D., and H. Sebastian Seung. "Algorithms for non-negative matrix factorization." Advances in neural information processing systems. 2001.

[23] Lops, Pasquale, Marco De Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends." Recommender systems handbook. Springer, Boston, MA, 2011. 73-105.

[24] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, Combining content-based and collaborative filters in an online newspaper, Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation (Berkeley, CA), 1999.

[25] Miettinen, Pauli, and Jilles Vreeken. "MDL4BMF: Minimum description length for Boolean matrix factorization." ACM Transactions on Knowledge Discovery from Data (TKDD) 8.4 (2014): 18.

[26] Mnih, Andriy, and Ruslan R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems. 2008.

[27] O. Celma and P. Herrera, A new approach to evaluating novel recommendations, Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08) (Lausanne, Switzerland), ACM Press, 2008, pp. 179–186.

[28] P. Cotter and B. Smyth, PTV: Intelligent personalised tv guides, Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, AAAI Press/MIT Press, 2000, pp. 957–964.

[29] R. Burke, P. Brusilovsky and A. Kobsa and W. Nejdl, Hybrid web recommender systems, The Adaptive Web: Methods and Strategies of Web Personalization, Springer, Heidelberg, Germany, 2007, pp. 377–408.

[30] R. M. Bell, Y.Koren, and C.Volinsky, The BellKor solution to the Netflix Prize, Tech.Report http://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf, AT&TLabs Research, 2007.

[31] Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.

[32] Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[33] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

[34] Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating "word of mouth"." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press/Addison-Wesley Publishing Co., 1995.

[35] Snášel, Václav, et al. "Investigating boolean matrix factorization." Proc. Workshop on Data Mining using Matrices and Tensors. 2008.

[36] T. Nathanson, E. Bitton, and K. Goldberg, Eigentaste 5.0: constant-time adaptability in a rec-

ommender system using item clustering, Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07) (Minneapolis, MN), ACM, 2007, pp. 149–152.

[37] Trewin, Shari. "Knowledge-based recommender systems." Encyclopedia of library and information science 69.Supplement 32 (2000): 180.

[38] Van Meteren, Robin, and Maarten Van Someren. "Using content-based filtering for recommendation." Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop. 2000.

[39] Webb, Brandyn. "Netflix update: Try this at home." Blog post sifter. org/simon/journal/20061211. html (2006).

[40] Yang, Jaewon, and Jure Leskovec. "Community-affiliation graph model for overlapping network community detection." 2012 IEEE 12th International Conference on Data Mining. IEEE, 2012.