
# Modeling the Evolution of Fashion Trends using Matrix Factorization Techniques

## Master Thesis

Leutrim Kaleci

Supervisors: Dr. Mourad Khayati
Prof. Philippe Cudré-Mauroux

University of Fribourg

February, 2018



# Abstract

Nowadays, the job of a recommender system gets more and more complicated as the result of new data coming in constantly. This has impact also on the preferences of the users to whom the items are recommended - who tend to change their preferences as new items appear. Such that, different time-dependent models are proposed for capturing this dynamicity, by always considering the application context of the recommender system. According to the application context of the recommender system, different dynamics are at play. In this thesis, the focus is on the application of three different models in the context of fashion data. Respectively, firstly the *timeSVD++* model (applied earlier on movie data, and showed good performance) and *TVBPR* model (a model which, additionally, makes use of other source of information for learning the parameter values) are deeply analyzed, and then the performance with regard to their application in the context of fashion data is provided. Afterwards, a new model is introduced, *timeSVD*$_{VC}$, by extending the *timeSVD++* model with one more additional component, the visual component. The extended version of *timeSVD++* provided promising results, such that showing better performance when compared to two other models mentioned above.

Dr. Mourad Khayati, eXascale Infolab, Department of Informatics, University of Fribourg, Supervisor

Prof. Dr. Philippe Cudré-Mauroux, eXascale Infolab, Department of Informatics, University of Fribourg, Supervisor

# Contents


**1 Introduction** **5**
- 1.1 Motivation . . . 6
- 1.2 Goal . . . 7
- 1.3 Organization . . . 7

**2 Related Work** **8**

**3 Preliminaries** **10**
- 3.1 Notation . . . 10
- 3.2 Background . . . 11
- 3.3 Evaluation Methodologies . . . 11
  - 3.3.1 AUC (Area under the ROC curve) . . . 11

**4 Time-aware Models** **16**
- 4.1 timeSVD++ . . . 16
- 4.2 TVBPR+ . . . 22
  - 4.2.1 Temporally Visual Factors . . . 23
  - 4.2.2 Temporally Evolving Visual Bias . . . 24
  - 4.2.3 Non-visual Temporal Dynamics . . . 24
  - 4.2.4 Fashion Segmentation . . . 24
  - 4.2.5 Learning the Model . . . 24
  - 4.2.6 Example . . . 25

**5 TimeSVD++ with the Visual Component Included** **30**
- 5.1 Description . . . 30

**6 Experiments** **32**
- 6.1 Dataset . . . 33
  - 6.1.1 A Single Context Showing the Performance of the Models . . . 35
- 6.2 Accuracy - Comparisons and Analysis of the Performance of the Models . . . 35
  - 6.2.1 1st Task . . . 36
  - 6.2.2 2nd Task . . . 37
  - 6.2.3 3rd Task . . . 38
  - 6.2.4 4th Task . . . 39
  - 6.2.5 An Exploratory Study . . . 40
- 6.3 Efficiency - Comparisons and Analysis of the Performance of the Models . . . 49

**7 Discussion** **53**

**8 Conclusion** **55**

# List of Figures


6.1 The evolution of the number of ratings over time. . . . . 34
6.2 The performance of the models over different number of epochs. . . . . 36
6.3 The distribution of data across different number of epochs. . . . . 37
6.4 The performance of the models while the number of non-visual factors is changing. . . . 38
6.5 The performance of the models while the number of visual factors is changing. . . . . 39
6.6 The performance of the models while the dataset size is changing. . . . . 40
6.7 The number of items per day - calculated by considering only the first rating given to them. 41
6.8 Bias score vs user-item interaction score over time when *timeSVD++* is applied. . . . . 42
6.9 Bias score vs user-item interaction score over time when *TVBPR* is applied. . . . . 43
6.10 Visual user-item interaction score vs total user-item interaction score when *TVBPR* is applied. . . . . 44
6.11 Total bias score vs total user-item interaction score when *timeSVD*$_{VC}$ is applied. . . . . 45
6.12 Visual user-item interaction score vs total user-item interaction score when *timeSVD*$_{VC}$ is applied. . . . . 46
6.13 The running time of the models over different dataset sizes - the time is presented in seconds. 50
6.14 The running time of the models while the # of non-visual factors differs - the time is presented in seconds. . . . . 51
6.15 The running time of the models while the # of visual factors differs - the time is presented in seconds. . . . . 52

# List of Tables


3.1 Notation . . . 10
3.2 Input - binary-based information . . . 13

4.1 Input - explicit feedback given . . . 18
4.2 User bias-related parameters presented in Eq. 4.3, where the first two columns show the values of $\beta_u$ and $\alpha$. Whereas the next columns represent the parameter capturing the drifts on daily basis for the corresponding day, i.e. $\beta_{u,t}$, where 0 indicates there was no action or rating given on that day . . . 18
4.3 Item bias-related parameters from Eq. 4.2 . . . 19
4.4 Item factors denoted with $\gamma_i$ in Eq. 4.7 . . . 19
4.5 Item factors related to implicit information denoted with $y_j$ in Eq. 4.7 . . . 19
4.6 User static factors denoted with $\gamma_u$ in Eq. 4.6 . . . 20
4.7 User time-dependent related parameter $\alpha_{u,k}$, presented in Eq. 4.6 . . . 20
4.8 User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 1, presented in Eq.4.6 . . . 20
4.9 User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 2, presented in Eq.4.6 . . . 20
4.10 User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 3, presented in Eq.4.6 . . . 20
4.11 User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 4, presented in Eq.4.6 . . . 20
4.12 Item Bias and Factors, $\beta_i$ and $\gamma_i$ respectively . . . 26
4.13 User factors, $\gamma_i$ respectively . . . 26
4.14 User visual factors, $\theta_u$ respectively . . . 26
4.15 The static embedding matrix E . . . 26
4.16 The embedding matrix E(t) at epoch 1 . . . 27
4.17 The embedding matrix E at epoch 10 . . . 27
4.18 The visual factor weighting vector - $\omega(t)$ . . . 27
4.19 Visual bias - related parameters . . . 27
4.20 The visual bias weighting vector - $b(t)$ . . . 28

6.1 Dataset statistics (after processing) . . . 33
6.2 TVBPR Visualization . . . 47
6.3 timeSVD_VC Visualization . . . 48
6.4 Efficiency of the algorithms while applied on different dataset sizes in seconds . . . 49
6.5 Efficiency of the algorithms while the number of the non-visual factors varies . . . 50
6.6 Efficiency of the algorithms while the number of the visual factors varies . . . 51

# 1 Introduction

Developing customer-oriented applications that would match users' specific tastes, is of high importance in the today's big data era. These kind of applications would require to capture users' preferences toward items of their interest, and thus providing to the user the data that most likely would fit his or her interests. Some example of such applications are: Netflix, Amazon, Youtube, etc. Basically, such services are provided by the recommender systems. Nowadays, there are different techniques that help on designing recommender systems by modeling users' preferences based on their past activities, and thus suggesting to the users the products that most likely match their preferences. In fact, there are two general approaches when it comes to apply recommender systems. One approach is based on the so called content filtering, and the other approach termed as collaborative filtering (see Chapter 2).

As a consequence of new data coming in constantly, users tend to change their preferences. This has influenced researchers to design new models that would capture those changes, and thus modeling recommender systems that would suggest the most likely preferred item to user at given point in time. Thus, as data are evolving, modeling accurate model would require to capture the present nature of the data that has low impact on the future behavior, and at the same time keeping the underlying evolution of the data. As stated in [7], this problem is termed as concept drift. Such examples of concept drifts involve the change of the focus of the user as a consequence of something happening, e.g. the emergence of new products which might have impact on changing the users' shopping patterns. As these changes might be influenced from different reasons, they cannot be captured by methods that would capture the global concept drifts. Thus, there is a need for methods that would be able to capture different concept drifts which might occur at different points in time. Capturing such changes at the level of individual reduces the amount of concept drifts that are at play. According to [11], there exist three different approaches for handling concept drift: the instance selection approach, the instance weighting approach and ensemble learning. Regarding the instance selection approach, it takes into consideration only the relevant instances to the current concept. Windows-based techniques provide the functionality of this approach where only the recent instances are taken into account. Whereas, regarding the instance weighting approach, it uses the abilities of some learning algorithms for doing the weighting of the instances. According to some experiments, this approach proved that its performance on handling concept drift has been worse compared to the instance selection approach. When it comes to ensemble learning (learning with multiple concept description), this approach makes use of set of concepts description upon which some predictions are

performed. Then a weighting procedure to those prediction results is performed, which calculates the relevance of the instances.

An example of such application of capturing the concept drifts involved on the data being considered are provided by Koren [7], where specific techniques for modeling time drifting user preferences in the context of movie data are presented. Their models are evaluated on Netflix data, and proved to outperform earlier approaches. The provided methods are built based on two leading collaborative filtering approaches: matrix factorization based models and neighborhood-based models.

In this thesis, our main focus is the application of time-aware models in the context of fashion data where the visual appearances of the products are considered to play a key role on user's decision-making behavior. Such applications could be related also to other similar contexts where visual appearance of a product influences user's opinion. Thus, developing recommender systems that would make use of visual characteristics of products in such contexts, might increase the performance of the recommender systems, i.e. the characteristics of the product recommended to the user are better matched with the preferences of the user.

As shown in [4], along the positive aspects of taking into account the visual information while modeling users' preferences, there exist also some challenges. As different and complex visual factors might be at play, it makes it difficult to capture the factors with the highest influence. It would require a large corpora in order to capture any meaningful signal about visual information which would let us know which visual factors play a key role on a user's purchase. Also, since visual preferences are considered to be highly personal, it makes it more difficult to distinguish each user's unique preferences. Considering that visual factors evolve over time as well, such that different visual factors might have different importance at different points in time, it adds additionally complexity on capturing those visual factors. Especially, this can be noticed more in the context of fashion data where different products can be fashionable during different periods of time.

At the end, having dealt with all those challenges and having built such recommender systems, we would be able also to answer questions about how visual features have evolved over time, and how those features have affected users' decisions. Also, questions about which visual features are more preferred at some period of time, and which are not.

As the focus of this thesis is on the visual aspects of such applications in fashion context, we want to emphasize that incorporating directly the visual features describing the fashion-related products into certain model which does not make use of such information before, it has direct impact on the performance of the model. The importance of the visual aspects in fashion context has been also emphasized in [5], where is stated that "one wouldn't buy a t-shirt from Amazon without seeing the item in question". Such that, we want to incorporating the visual aspects into a model where do not exist currently.

## 1.1 Motivation

This work is motivated based on a previously work presented in [4], where the modeling of visual evolution of fashion trends is presented. One-Class Collaborative Filtering approach has been used for modeling fashion-aware personalized ranking for each user. Basically, this approach provides a fashion-aware recommender system that is time-dependent and such that, it is able to capture different visual features of products that are predominating (i.e. visual features that influence users' opinions) during different periods of time. Capturing such features, we are able to see the evolution of given product, i.e. knowing when a product has been fashionable, and when it has gone out of fashion. Modeling in way of finding the factors that influence users' opinions or choices at different time in points is a complex task, since considering that these kind of changes are sometimes unexpected, and thus hard to capture. Moreover, regarding the work presented in [7], *timeSVD++* is introduced. *timeSVD++* is a latent factor model, and such that its parameter values are only inferred from the input data (i.e. ratings). Thus, as *timeSVD++* does not make

use of other source of information for learning the parameter values, applying it in a context of fashion data (where the visual appearance of products plays a key role on users' choice) might not capture the right signals that influence users' preferences. Considering that, extending the *timeSVD++* with a component which makes use of visual information of the product (as presented in the earlier approach related to the visual evolution of fashion trends) might improve the performance.

## 1.2 Goal

The goal of this thesis is to, firstly experiment with the already existing models, and thus see the power of those models in fashion context. Then, to introduce a new model, which is a combination of *timeSVD++* model and the model presented in [4], *TVBPR*. Afterwards, having all the models in place, the goal is to provide some experiments, which would help us see how the models perform in the context of fashion data, and thus providing a comparison between the performances of the models.

## 1.3 Organization

Chapter 2 provides a description about the similar works done in the field. Next, Chapter 3 introduces some preliminaries needed for the subsequent chapters. Then, Chapter 4, provides detailed description about the already existing models that are used for the experiments. Chapter 5 presents the newly introduced model. Then, in Chapter 6, some experiments and their results showing the performance of each model are provided. Chapter 7 discusses the problems faced throughout the work of this thesis. Finally, Chapter 8 concludes the work.

# 2
# Related Work

There exist different efforts that have been done earlier with regard to modeling users' preferences. Some of them provide approaches where only visual aspects are taken into consideration, and some of them provide different approaches for modeling the temporal dynamics. Also, there are few approaches where both of aspects above are taken into consideration.

**Visual Dimensions** As described on [4], there has been extensive research going on with regard to the importance of the usage of images in e-comerce scenarios. For example, Wei et. al [2] try to understand the impact of images by using data mining and knowledge discovery approach. Their results show that the 'presence' of the images increases "the buyer's attention, trust and conversion rate". Another related study presented on [3], it shows also the importance of the image features of the online products by firstly looking for important feautres of the images considering online products, and then see if there can be achieved any imporvment on CTR(click-through rate) ranking models. Their results show statistically significant correlations between image features and CTR, and that the image features of products have impact on users' search behavior. Another study presented on [8], they provide a solution for a system which would tell which products go toghether. This is achieved by finding whether two images of the products are compatible or not, which is performed by learning procedure of the distances between those images. Also, another work presented on [12] is telling the user how fashionable he or she is by showing only a picture of the user. To tackle this, they make use of Conditional Random Field model.

There are also approaches where visual-aware collaborative filtering is used. As stated on [4], both users and items are mapped into visual space, and thus measuring the users' preferences. This work is presented on [5], but temporal dynamics are not taken into account.

Comparing to our work, the focus in our work is on showing the evolution of items, and thus being able to see which items are fashionable and for each of them being able to measure their visual popularity – basically finding the most important visual features of images at given period of time.

**Modeling Temporal Dynamics** Regarding temporal dynamics, it relates to the notion of the concept drift desribed in Chapter 1, where one of the three different approaches is used for handling concept drift.

Cebrián et al. [1] present the approach for designing a music recommender system where the preferences of the users are captured by creating micro-profiles for each user by using contextual information.

So, a segmentation of the contextual space is performed, and thus creating the micro profiles for each user with relevant contextual information.

Zhang et al. [14] present daily-aware personalized recommendation by making use of textual information (i.e. reviews) to extract explicit product features. Afterwards, having the features extracted, they analyze the seasonal effects and other involved movements on features to see how the preferences of the users change over time.

Also, Koren [7] designed models where temporal dynamics of the data were taken into consideration. Since his approach is used for our experiments, it is in details discussed (see Chapter 4).

The work presented in [4] provides an approach of combining the visual and temporal dynamics. Their approach is presented in details as well (see Chapter 4).

# 3 Preliminaries

## 3.1 Notation

The Table 3.1 below provides a description about the notation used throughout the report of the thesis:

| Notation | Explanation |
|---|---|
| $U, I$ | $userset, itemset$ |
| $x_{u,i}$ | the rating value of user u towards item i |
| $\hat{x}_{u,i}$ | the predicted rating value of user u towards item i |
| $\beta_u, \beta_i$ | user bias, item bias |
| $\beta_u(t), \beta_i(t)$ | user bias at time t, item bias at time t |
| $K$ | the dimensions of latent factors |
| $K'$ | the dimensions of visual factors |
| $\gamma_u$ | the latent factors of user u, $(K \times 1)$ |
| $\gamma_i$ | the latent factors of item i, $(K \times 1)$ |
| $\theta_u$ | the visual factors of user u, $(K' \times 1)$ |
| $\theta_i$ | the visual factors of item i, $(K' \times 1)$ |
| $\theta_u(t)$ | the visual factors of user u at time t |
| $\theta_i(t)$ | the visual factors of item i at time t |
| $R(u)$ | the set of items rated by user u |
| $E$ | $K' \times F$ embedding matrix |
| $E(t)$ | $K' \times F$ embedding matrix at time t |
| $f_i$ | the visual features of item i |
| $F$ | the dimensions of visual features |

Table 3.1: Notation

## 3.2 Background

**Recommender Systems** According to [10] and other literatures, there are two general approaches when it comes to apply recommender systems. One approach is based on the so called content filtering, and the other approach termed as collaborative filtering.

**Content filtering** This type of recommender systems use information that characterize both users and items, and thus a matching of characteristics procedure is performed. But, often that kind of information is hard to be gathered, such that it makes this method not very applicable on some circumstances.

**Collaborative Filtering** This method relies on past user's activities − transactions. These activities might include: browsing, mouse movements, clicks, etc. Having information about users' past activities, this method is able to recommend to a user the items that similar users to him or her have liked before, but he or she still has not interacted with.

## 3.3 Evaluation Methodologies

For measuring the performance of a recommender system, according to [13], two general classes are distinguished:

- The first class is related to the rating prediction task where the performance is calculated using the RMSE (Root Mean Squared Error) which is formulated as follows:

$$RMSE = \sqrt{\frac{1}{|testSet|} \sum_{(u,i) \in testSet} (\hat{x}_{u,i} - x_{u,i})^2} \tag{3.1}$$

- The second class is related to the item recommendation task, where the performance is measured using precision and recall based metrics. For example, regarding the [4], for measuring the performance of the model, the widely used AUC has been used:

$$AUC = \frac{1}{|U|} \sum_{u} \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\hat{x}_{u,i}(t_{ui}) > \hat{x}_{u,j}(t_{ui})) \tag{3.2}$$

As our focus is on item recommendation task, we will focus on AUC-based evaluation methodology by giving detailed description of its application on an example.

### 3.3.1 AUC (Area under the ROC curve)

In the following, a detail description about the evaluation methodology used for measuring the performance of different models is given. Firstly, some basic concepts with the regard to AUC are given. After having an understanding of the basic concepts, we show an example when AUC is applied. The model that is being tested is *timeSVD++*, but as our focus is on the evaluation methodology, there are no information provided with regard to the training procedure of the model.

**Basic Concepts** AUC (Area under the ROC curve) is widely used for evaluating the performance of a binary classifier. Thus, we are able to see visually the performance of a classifier. ROC curve is used to plot the performance of the model by considering the TPR (True Positive Rate) against (False Positive Rate) with different threshold values being considered.

For illustration purposes, let's label a class p as a positive class, and class n as a negative one. An item which is part of the p class, and is classified as being part of the p class, the classification is considered as true positive (TP) classification. Whereas, if the item is classified as being of p class, but not being of the p class, it is considered as false positive (FP) classification. When an item is part of n class, and it is classified as being of n class, the classification is considered as true negative (TN). On the other hand, if the item is part of p class, but it is classified as being part of the n class, the classification is called false negative (FN). For calculating the TPR, the following formula is used:

$$TPR = \frac{TP}{(TP + FN)} \tag{3.3}$$

For calculating the FPR, the following formula is used:

$$TPR = \frac{FP}{(FP + TN)} \tag{3.4}$$

Having the ROC curve plotted by using different threshold values, we are able to calculate the AUC (area under the ROC Curve). As describe on [9] AUC is equal to the probability that a classifier will rank higher a positive instance compared to a randomly chosen negative instance. Whereas, regarding the context of a ranking task, according to [9], the AUC for the single user is calculated as defined in the following:

$$AUC(u) = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{u,i}(t_{ui}) > \hat{x}_{u,j}(t_{ui})) \tag{3.5}$$

where $|I_u^+|$ represents the set of items rated by user u (i.e. positive items), the $|I \setminus I_u^+|$ represents the set of items that have not been rated by user u (i.e. negative items), the $\hat{x}_{u,i}(t_{ui})$ represents the prediction value for the item i at time t, whereas the $\hat{x}_{u,j}(t_{ui})$ represents the prediction value for item j at time t. Whereas the $\delta$ function is defined as follows:

$$\delta(x) = \begin{cases} 1, & if(\hat{x}_{u,i}(t_{ui}) > \hat{x}_{u,j}(t_{ui})). \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

When AUC is applied for evaluating a model, regarding the first part of the formula $|I_u^+||I \setminus I_u^+|$, $I_u^+$ shows the positive items of the user u, by considering only the positive items from the testing set. Whereas regarding the subsequent part of the first part mentioned above($I \setminus I + u$), it is associated to all negative items of user u, such that considering all items user u did not have any interaction with (including the items from the training, testing and validation set). Considering this, we can formulate the formula above as follows:

$$AUC(u) = \frac{1}{|T_u||I \setminus I_u^+|} \sum_{i \in T_u} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{u,i}(t_{ui}) > \hat{x}_{u,j}(t_{ui})), \tag{3.7}$$

where $T_u$ represents the testing set with respect to the user u.

And the average AUC is defined as follows :

$$AUC = \frac{1}{|U|} \sum_{u \in U} AUC(u) \tag{3.8}$$

**Example** *timeSVD++* is the model that is being tested on fashion data, which is formulated as follows:

$$\hat{x}_{u,i} = \alpha + \beta_u(t) + \beta_i(t) + \gamma_i^T \left( \gamma_u(t) + |R(u)|^{-1/2} \sum_{j \epsilon R(u)} y_j \right) \tag{3.9}$$

where $\alpha$ is the global offset, and $\beta_u(t)$ and $\beta_i(t)$ represent user bias at time t and item bias at time t respectively. Whereas, the $\langle \gamma_i \rangle$ represents the latent factors related to item i, and $\langle \gamma_u(t) \rangle$ represents the latent factors related to user u at time t. Considering $|R(u)|^{-1/2} \sum_{j \epsilon R(u)} y_j$, it captures the implicit information with regard to user u, where $|R(u)|$ is the set of items rated by user u, and $y_j$ representing the vector of capturing those values.

In this context, we deal with two classes of items. The first class of items involves the items that the user has shown an implicit feedback to (positive items), and such that these items are considered as the ones being preferred by the user. Whereas, the other class involves the items that user has not interacted with (i.e. there is no implicit feedback between the user and the item, called as negative items), such that considered less preferred for the user [9]. Thus, after the model is being trained on the training dataset, for certain user, it should predict a higher preference value to a positive item compared to the preference value of a negative item.

Let's take the matrix presented in in Table 3.2 below as an input matrix. The binary values are used to represent which items were rated by which user. Such that, with 1 indicating that user u rated item i, and 0 otherwise:

Table 3.2: Input - binary-based information

| | user 1 | user 2 | user 3 | user 4 |
|---|---|---|---|---|
| i1 | 1 | 0 | 0 | 0 |
| i2 | 1 | 0 | 0 | 0 |
| i3 | 1 | 0 | 0 | 0 |
| i4 | 0 | 1 | 0 | 0 |
| i5 | 0 | 1 | 0 | 0 |
| i6 | 0 | 1 | 0 | 0 |
| i7 | 0 | 1 | 0 | 0 |
| i8 | 0 | 1 | 0 | 0 |
| i9 | 0 | 1 | 0 | 0 |
| i10 | 0 | 1 | 0 | 0 |
| i11 | 0 | 0 | 1 | 0 |
| i12 | 0 | 0 | 1 | 0 |
| i13 | 0 | 0 | 1 | 0 |
| i14 | 0 | 0 | 1 | 0 |
| i15 | 0 | 0 | 0 | 1 |
| i16 | 0 | 0 | 0 | 1 |
| i17 | 0 | 0 | 0 | 1 |
| i18 | 1 | 0 | 0 | 0 |
| i19 | 0 | 1 | 0 | 0 |
| i20 | 0 | 0 | 1 | 0 |
| i21 | 0 | 0 | 0 | 1 |
| i22 | 1 | 0 | 0 | 0 |
| i23 | 0 | 1 | 0 | 0 |
| i24 | 0 | 0 | 1 | 0 |
| i25 | 0 | 0 | 0 | 1 |

As it can be seen from the Table 3.2 above, it is divided into three parts: the blank part represents the

training set, the part in a lighter gray color represents the testing set, and the part with a darker gray color represents the validation set. As we are interested to show just an example of how AUC is applied in our context, only the test set will be considered (leave-two-out cross validation technique has been used for splitting the data as shown above). The positive items corresponding to each user are the following:

- $|I_1^+| : i1, i2, i3.$
- $|I_2^+| : i4, i5, i6, i7, i8, i9, i10.$
- $|I_3^+| : i11, i12, i13, i14.$
- $|I_4^+| : i15, i16, i17.$

The negative items corresponding to each user are the following:

- $|I \setminus I_1^+| : i4, \ldots, i20, 21, i23, i24, i25$
- $|I \setminus I_2^+| : i1, \ldots, i3, i11, \ldots, i22, i24, i25$
- $|I \setminus I_3^+| : i1, \ldots, i10, i15, \ldots, i23, i25$
- $|I \setminus I_4^+| : i1, \ldots, i14, i18, \ldots, i24$

Regarding the test set being considered, after training the values of the parameters, the formula presented in Eq. 3.7 is applied for each user as shown below:

- $AUC(1) = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{u,i}(t_{ui}) > \hat{x}_{u,j}(t_{ui})) = \frac{1}{20}\Big((\delta(5.1443 > 4.8620) + (\delta(5.1443 < 5.4961) + (\delta(5.1443 < 5.9138) + (\delta(5.1443 > 4.8904) + (\delta(5.1443 > 4.4228) + (\delta(5.1443 > 5.1157) + (\delta(5.1443 < 5.2502) + (\delta(5.1443 < 6.5588) + (\delta(5.1443 < 5.9096) + (\delta(5.1443 < 5.3342) + (\delta(5.4961 < 6.4100) + (\delta(5.1443 < 6.1531) + (\delta(5.4961 > 4.4866) + (\delta(5.1443 > 4.9167) + (\delta(5.4961 < 5.6979) + (\delta(5.1443 < 6.0161) + (\delta(5.1443 < 5.1638) + (\delta(5.4961 > 4.9764) + (\delta(5.1443 < 5.8984) + (\delta(5.4961 < 6.1268)\Big) = \frac{1}{20}\Big(1 + 0 + 0 + 1 + 1 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1 + 0 + 0 + 0 + 0 + 1 + 0\Big) = \frac{7}{20} = 0.35$
- $AUC(2) = \ldots = 0.9375$
- $AUC(3) = \ldots = 0.1052$
- $AUC(5) = \ldots = 0.45$

Then, to calculate the average performance of the model, the formulation in Eq. 3.8 is applied:

$$\begin{aligned} AUC &= \frac{1}{4}\Big(AUC(1) + (AUC(2) + AUC(3) + AUC(4)\Big) \\ &= \frac{1}{4}(0.35 + 0.9375 + 0.1052 + 0.45) = \frac{1}{4} \cdot 1.8427 \\ &= 0.460675 \end{aligned}$$

Basically, the $AUC(u)$ above shows the probability that the model will predict correct prediction (i.e. the positive item must be ranked higher than the negative item, meaning positive item must have higher predicted preference value than the negative item) for that user. This calculation is associated to recall - which is related to the information retrieval and calculated by considering the number of relevant retrieved

documents out of total number of relevant documents. Whereas in binary classification, it is denoted as sensitivity which is calculated with the formula related to TPR. So, coming back to our context, after doing the procedure above for each user, the overall $AUC$ is calculated by taking into account all the calculated $AUC(u)$ values, summing them up and dividing by the number of users. This way we will have the overall performance of the model being considered.

# 4
# Time-aware Models

In this chapter, a detailed description about the already existing time-aware models that are taken into consideration is given. Firstly, we start by describing the *timeSVD++* model. Then, having an understanding of *timeSVD++*, a description about *TVBPR+* is given, where additionally other source of information is used for modeling the evolution of fashion trends.

## 4.1 timeSVD++

**Desription** *timeSVD++* is a time-aware model capturing the dynamics that are at play on the data where the model is applied on. *timeSVD++* has been applied on movie data and has shown to perform well in such context. It is an extension of *SVD++* model presented in [6], where implicit information additionally are taken into consideration compared to the 'standard' matrix factorization method whose formulation looks as follows:

$$\hat{x}_{u,i} = \alpha + \beta_u + \beta_i + \langle \gamma_u, \gamma_i \rangle, \tag{4.1}$$

where $\alpha$ is the global offset, and $\beta_u$ and $\beta_i$ represent user bias and item bias respectively. Whereas, the $\langle \gamma_u, \gamma_i \rangle$ represents the dot product between the vector of the factors of user u and factor vector of item i of K dimensions, thus capturing the interaction between user $u$ and item $i$.

The basic model 4.1 above is a static one, and such that is not able to capture the evolving data. In order to extend the static model into a time-aware model, Koren [7] distinguishes between which parameter values change as data are evolving, and such that being able to model those parameters in such a way that would be able to capture the exact changes that occur on the data over time. Such that, there are parameters that describe the evolution of users and others that describe items' evolution. According to [7], where *timeSVD++* is applied on Netflix data, time affects the parameters that are involved within baseline predictors, and also the parameters about the factors describing users' characteristics.

Regarding the baseline predictors, there are two parameters that should be built as a function of time: user biases and item biases. For capturing item biases, splitting the dataset into time-based bins is proposed, where each bin corresponds to a defined period of time. Such that, the function for capturing the evolution of the item bias is formulated as follows:

$$\beta_i(t) = \beta_i + \beta_{i,Bin(t)} \tag{4.2}$$

The $\beta_i$ parameter in the equation 4.2 captures the static part, whereas the $\beta_{i,Bin(t)}$ captures the evolving part, where $Bin(t)$ is the index of the bin timestamp $t$ corresponds to.

Regarding the changes on the user side, the binning approach is not preferable. When designing functions that capture users' biases, there is a need for a function with a finer resolution, which would be able capture those sudden and unexpected changes that eventually might occur. Considering this, another way of modeling users' biases is used. The formula below captures the possible gradual drifts on user's side:

$$\beta_u(t) = \beta_u + \alpha \cdot dev(u,t), \tag{4.3}$$

where $\alpha$ is a parameter of the model which is learned from the data, whereas the $dev(u,t)$ function captures the deviation which is formulated as follows:

$$dev(u,t) = sign(t - t_u) \cdot |t - t_u|^{\eta}, \tag{4.4}$$

where $t$ is the timestamp of the rating given by user $u$, and $t_u$ is the mean rating time considering all the timestamps user $u$ has rated. Whereas $\eta$ is value which is set according to the cross validation, i.e. when testing the model.

The function 4.3 above capturing the dynamics of users' biases is only able to capture the gradual drifts of the users. Considering the fact that users' changes occur on daily basis, the function above is not able alone to capture those changes. Thus, another parameter is added to the function 4.3 above, which is associated with the daily-based changes. Such that, the user bias function gets the following form:

$$\beta_u(t) = \beta_u + \alpha \cdot dev(u,t) + b_{u,t} \tag{4.5}$$

As considered that users' preferences change over time, i.e. some user might prefer or be fan of something during one period of time, and prefer something else during another period of time. Such that, a function for capturing these changes is also provided. The formulation of the function looks as in the following:

$$\gamma_{uk}(t) = \gamma_{uk} + \alpha_{uk} \cdot dev_u(t) + \gamma_{uk,t} \quad k = \{1, \ldots, f\} \tag{4.6}$$

Combining all the formulas above, the 'standard' formulation of the matrix factorization formula gets the following form:

$$\hat{x}_{u,i} = \alpha + \beta_u(t) + \beta_i(t) + \gamma_i^T \left( \gamma_u(t) + |R(u)|^{-1/2} \sum_{j \epsilon R(u)} y_j \right) \tag{4.7}$$

**Example** In the following, an example is provided, which shows how the formulas above are applied. The input data used for this example is the same one as the one used for the experiment related to the application of AUC (see Chapter 2), but instead the real ratings given to items are provided, as shown in Table 4.1 - each column represents the ratings (in scale 1 to 5, and 0 indicating there is not rating given) of certain user towards the items, or vice versa each row representing the ratings given to certain item.

| | user 1 | user 2 | user 3 | user 4 |
|---|---|---|---|---|
| B0058VLRIA | 5.0 | 0 | 0 | 0 |
| B009V06HGG | 5.0 | 0 | 0 | 0 |
| B00B1VFWDW | 5.0 | 0 | 0 | 0 |
| B00517ABLA | 0 | 5.0 | 0 | 0 |
| B00671EMO6 | 0 | 4.0 | 0 | 0 |
| B007KFS10O | 0 | 2.0 | 0 | 0 |
| B0081U2VCG | 0 | 5.0 | 0 | 0 |
| B00AB93O5S | 0 | 3.0 | 0 | 0 |
| B00B5505VI | 0 | 3.0 | 0 | 0 |
| B00D5ZLMKE | 0 | 3.0 | 0 | 0 |
| B00062NHGQ | 0 | 0 | 4.0 | 0 |
| B004K1KR4E | 0 | 0 | 4.0 | 0 |
| B004SDYQI2 | 0 | 0 | 5.0 | 0 |
| B00866CKU8 | 0 | 0 | 3.0 | 0 |
| B00127QC0C | 0 | 0 | 0 | 4.0 |
| B004AE1X1C | 0 | 0 | 0 | 5.0 |
| B0072T3QTE | 0 | 0 | 0 | 5.0 |
| B003EAORCA | 5.0 | 0 | 0 | 0 |
| B001RNO3KC | 0 | 3.0 | 0 | 0 |
| B00025FROW | 0 | 0 | 4.0 | 0 |
| B000SKNYAU | 0 | 0 | 0 | 5.0 |
| B005LCQKGU | 5.0 | 0 | 0 | 0 |
| B0058XU98M | 0 | 1.0 | 0 | 0 |
| B0010FVNWS | 0 | 0 | 5.0 | 0 |
| B003TSTS94 | 0 | 0 | 0 | 5.0 |

Table 4.1: Input - explicit feedback given

Firstly, the learning process of the parameters is applied, and then the formulation in Eq. 4.7 is used to predict the preference value of certain user $u$ towards certain item $i$. In our example, we will predict the preference values of the users from the testing set. The global average rating value is, $\alpha = 4.117$.

In the following, we will show the values of the parameters after the learning process. In Table 4.2 and Table 4.3, the values of the parameters related to biases are shown. Whereas, the factor-related values characterizing items and users are provided in Table 4.4 and 4.6 respectively. Regarding the parameter capturing the drifts of users' preferences on daily basis are shown in the following tables: Table 4.8, Table 4.9, Table 4.10 and Table 4.11.

| | $\beta_u$ | $\alpha$ | 881 | 1111 | 1298 | 1425 | 1502 | 1509 | 1542 | 1554 | 1643 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| user 1 | -0.2072 | 0 | 0 | 0 | 0 | 0 | -0.2072 | 0 | 0 | 0 | 0 |
| user 2 | -0.2050 | -0.0640 | 0 | -0.0964 | 0 | -0.1153 | 0 | 0 | 0 | 0 | 0 |
| user 3 | -0.2385 | -0.0104 | -0.0698 | 0 | -0.0536 | 0 | 0 | 0 | -0.0924 | 0 | -0.0297 |
| user 4 | -0.2312 | 0.03254 | 0 | 0 | 0 | 0 | 0 | -0.1261 | 0 | -0.1085 | 0 |

Table 4.2: User bias-related parameters presented in Eq. 4.3, where the first two columns show the values of $\beta_u$ and $\alpha$. Whereas the next columns represent the parameter capturing the drifts on daily basis for the corresponding day, i.e. $\beta_{u,t}$, where 0 indicates there was no action or rating given on that day

| | $\beta_i$ | bin 1 | bin 2 | bin 3 | bin 4 | bin 5 | bin 6 | bin 7 | bin 8 | bin 9 | bin 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B0058VLRIA | -0.0188 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0188 |
| B009V06HGG | -0.1509 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.1509 |
| B00B1VFWDW | -0.04155 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.04155 |
| B00517ABLA | 0.0638 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0638 | 0 |
| B00671EMO6 | 0.0572 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0572 | 0 |
| B007KFS10O | -0.1104 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.1104 | 0 |
| B0081U2VCG | -0.0964 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0964 | 0 | 0 | 0 |
| B00AB93O5S | -0.0037 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0037 | 0 |
| B00B5505VI | -0.0649 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0649 | 0 |
| B00D5ZLMKE | -0.0631 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0631 | 0 |
| B00062NHGQ | -0.0297 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0297 |
| B004K1KR4E | -0.0698 | 0 | 0 | 0 | 0 | 0 | -0.0698 | 0 | 0 | 0 | 0 |
| B004SDYQI2 | -0.0536 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0536 | 0 | 0 |
| B00866CKU8 | -0.0924 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0924 |
| B00127QC0C | -0.1103 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.1103 |
| B004AE1X1C | -0.0170 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.0170 |
| B0072T3QTE | -0.1085 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.1085 |

Table 4.3: Item bias-related parameters from Eq. 4.2

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0058VLRIA | 0.9493 | 0.2091 | 0.4715 | 0.7511 | 0.2070 | 0.4910 | 0.1217 | 0.1668 | 0.8779 | 0.5146 | 0.8643 | 0.7171 | 0.0738 | 0.5310 | 0.3525 | 0.2144 | 0.6095 | 0.1298 | 0.0785 | 0.5915 |
| B005LCQKGU | 0.8149 | 0.4851 | 0.1412 | 0.0859 | 0.0352 | 0.4575 | 0.7947 | 0.1597 | 0.7101 | 0.5717 | 0.6078 | 0.0643 | 0.8372 | 0.8671 | 0.2872 | 0.6695 | 0.0379 | 0.9968 | 0.6688 | 0.4298 |
| B009V06HGG | 0.7374 | 0.3310 | 0.3161 | 0.4467 | 0.0666 | 0.2088 | 0.1054 | 0.4189 | 0.3416 | 0.7260 | 0.6001 | 0.6467 | 0.1331 | 0.6607 | 0.8224 | 0.5988 | 0.7877 | 0.2492 | 0.5267 | 0.2973 |
| B00B1VFWDW | 0.2697 | 0.73298 | 0.1456 | 0.3130 | -0.0008 | 0.8942 | 0.1000 | 0.7956 | 0.3867 | 0.2726 | 0.0169 | 0.2509 | 0.9578 | 0.8005 | 0.8826 | 0.0874 | 0.4817 | 0.1194 | 0.9358 | 0.6369 |
| B001RNO3KC | 0.9509 | 0.7466 | 0.0470 | 0.1123 | 0.7388 | 0.1865 | 0.1241 | 0.1500 | 0.9940 | 0.5791 | 0.0787 | 0.9731 | 0.9765 | 0.3174 | 0.5083 | 0.3269 | 0.1354 | 0.5951 | 0.2832 | 0.4829 |
| B00517ABLA | 0.1902 | 0.2045 | 0.2857 | 0.1938 | 0.3174 | 0.7249 | 0.3013 | 0.8881 | 0.4212 | 0.4682 | 0.8850 | 0.5252 | 0.8397 | 0.2769 | 0.2228 | 0.9391 | 0.0930 | 0.0920 | 0.6068 | 0.3452 |
| B0058XU98M | 0.6027 | 0.4683 | 0.3579 | 0.0711 | 0.8470 | 0.3808 | 0.7635 | 0.9005 | 0.7490 | 0.5089 | 0.2834 | 0.7370 | 0.7379 | 0.9863 | 0.3958 | 0.9991 | 0.3048 | 0.0820 | 0.3753 | 0.7634 |
| B00671EMO6 | 0.0591 | 0.6454 | 0.3044 | 0.3712 | 0.3450 | 0.5589 | 0.1129 | 0.3843 | 0.6733 | 0.0512 | 0.8688 | 0.5854 | 0.2237 | 0.6509 | 0.8323 | 0.7111 | 0.1803 | 0.0887 | 0.4360 | 0.3214 |
| B007KFS10O | 0.7032 | 0.4377 | 0.0935 | 0.1133 | 0.4016 | 0.6898 | 0.7596 | 0.0862 | 0.9963 | 0.0026 | 0.8189 | 0.4060 | 0.1110 | 0.7227 | 0.6210 | 0.1571 | 0.2979 | 0.3475 | 0.1450 | 0.2905 |
| B0081U2VCG | 0.0789 | 0.0839 | 0.2402 | 0.2196 | 0.0741 | 0.0059 | 0.3050 | 0.8412 | -0.0524 | -0.0777 | 0.6114 | 0.2533 | 0.7333 | 0.4293 | 0.5961 | 0.1412 | 0.5378 | 0.2433 | 0.8923 | 0.4657 |
| B00AB93O5S | 0.8201 | 0.3883 | 0.9331 | 0.7580 | 0.4879 | 0.9292 | 0.4542 | 0.3974 | 0.8077 | 0.1728 | 0.1589 | 0.9348 | 0.7570 | 0.6237 | 0.8760 | 0.0973 | -0.0044 | 0.7669 | 0.0669 | 0.1771 |
| B00B5505VI | 0.0192 | 0.1925 | 0.7014 | 0.8177 | 0.0423 | 0.4304 | 0.9685 | 0.9117 | 0.3680 | 0.5840 | 0.8560 | 0.9876 | 0.4817 | 0.7084 | 0.7011 | 0.7987 | 0.8603 | 0.1537 | 0.5389 | 0.4374 |
| B00D5ZLMKE | 0.7671 | 0.4221 | 0.5252 | 1.0023 | 0.8667 | 0.5317 | 0.1039 | 0.8703 | 0.7345 | 0.2669 | 0.5540 | 0.7326 | 0.3233 | 0.2214 | 0.1242 | 0.2959 | 0.6664 | 0.4731 | 0.5615 | 0.6286 |
| B00025FROW | 0.2410 | 0.7118 | 0.2691 | 0.2028 | 0.0582 | 0.7482 | 0.7989 | 0.1675 | 0.8631 | 0.5359 | 0.3977 | 0.0442 | 0.3048 | 0.2731 | 0.6354 | 0.6548 | 0.0621 | 0.4991 | 0.07093 | 0.3376 |
| B00062NHGQ | 0.4371 | 0.1825 | 0.9025 | 0.4875 | 0.6990 | 0.9378 | 0.0855 | 0.8836 | 0.0638 | 0.7152 | 0.5308 | 0.8100 | 0.5170 | 0.2047 | 0.8149 | 0.9209 | 0.4800 | 0.2329 | 0.6034 | 0.8273 |
| B0010FVNWS | 0.7078 | 0.4937 | 0.4799 | 0.9802 | 0.8906 | 0.7548 | 0.5597 | 0.7127 | 0.9793 | 0.4572 | 0.1241 | 0.8016 | 0.2240 | 0.3220 | 0.4001 | 0.2892 | 0.0615 | 0.9388 | 0.9139 | 0.2161 |
| B004K1KR4E | 0.9221 | 0.0910 | 0.1712 | 0.2435 | 0.5767 | 0.6949 | -0.0395 | 0.0211 | 0.6000 | -0.0202 | 0.2997 | 0.2096 | 0.8163 | 0.6555 | 0.1800 | 0.0681 | 0.7365 | 0.3527 | 0.0118 | 0.7953 |
| B004SDYQI2 | 0.6942 | 0.7743 | 0.4889 | 0.0075 | -0.0223 | 0.6461 | 0.6608 | 0.2778 | 0.7562 | 0.5011 | 0.3246 | -0.0387 | 0.7974 | 0.9375 | 0.2828 | 0.1529 | 0.1066 | 0.4700 | 0.6091 | 0.0540 |
| B00866CKU8 | 0.8887 | 0.7707 | 0.7514 | 0.2907 | 0.9107 | 0.4278 | 0.0987 | 0.8079 | 0.0491 | 0.5688 | 0.5974 | 0.8965 | 0.1532 | 0.2064 | 0.7364 | 0.2176 | 0.4780 | 0.5124 | 0.3964 | 0.7546 |
| B000SKNYAU | 0.7889 | 0.9010 | 0.6935 | 0.6650 | 0.8378 | 0.8849 | 0.4766 | 0.8056 | 0.5666 | 0.7276 | 0.5263 | 0.1060 | 0.8995 | 0.3834 | 0.0728 | 0.2490 | 0.0564 | 0.6753 | 0.2292 | 0.4690 |
| B00127QC0C | 0.7999 | 0.2684 | 0.0873 | 0.1815 | 0.6348 | 0.6704 | 0.7016 | 0.1068 | 0.3314 | 0.9278 | 0.7902 | 0.8239 | 0.8008 | 0.3685 | -0.0544 | 0.6564 | 0.1465 | 0.2484 | 0.4528 | 0.0412 |
| B003TSTS94 | 0.3071 | 0.1479 | 0.2432 | 0.5721 | 0.0425 | 0.6721 | 0.3094 | 0.5666 | 0.6746 | 0.0374 | 0.0255 | 0.9214 | 0.4475 | 0.3554 | 0.9314 | 0.3165 | 0.1883 | 0.4763 | 0.2256 | 0.5327 |
| B004AE1X1C | 0.4262 | 0.8178 | 0.5655 | 0.2686 | 0.7124 | 0.6156 | 0.9162 | 0.5676 | 0.1339 | 0.4095 | 0.5548 | 0.9742 | 0.6361 | 0.9734 | 0.3343 | 0.1325 | 0.2204 | 0.9693 | 0.4403 | 0.2832 |
| B0072T3QTE | 0.1984 | 0.4259 | 0.8801 | 0.9275 | 0.4056 | 0.6957 | 0.0654 | 0.7468 | 0.8924 | 0.4806 | 0.7358 | 0.0015 | 0.3133 | 0.0957 | 0.6046 | 0.7894 | 0.6716 | 0.2116 | 0.6263 | 0.6911 |

Table 4.4: Item factors denoted with $\gamma_i$ in Eq. 4.7

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0058VLRIA | -0.0796 | -0.0498 | -0.0418 | -0.0590 | -0.0103 | -0.0447 | -0.0195 | -0.0592 | -0.0469 | -0.0803 | -0.0619 | -0.0704 | -0.0358 | -0.0812 | -0.0968 | -0.0614 | -0.0909 | -0.0298 | -0.0743 | -0.0531 |
| B005LCQKGU | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B009V06HGG | -0.0796 | -0.0498 | -0.0418 | -0.0590 | -0.0103 | -0.0447 | -0.0195 | -0.0592 | -0.0469 | -0.0803 | -0.0619 | -0.0704 | -0.0358 | -0.0812 | -0.0968 | -0.0614 | -0.0909 | -0.0298 | -0.0743 | -0.0531 |
| B00B1VFWDW | -0.0796 | -0.0498 | -0.0418 | -0.0590 | -0.0103 | -0.0447 | -0.0195 | -0.0592 | -0.0469 | -0.0803 | -0.06191022, -0.0704 | -0.0358 | -0.0812 | -0.0968 | -0.0614 | -0.0909 | -0.0298 | -0.0743 | -0.0531 | |
| B001RNO3KC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B00517ABLA | -0.0471 | -0.0204 | -0.03020 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B0058XU98M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B00671EMO6 | -0.0471 | -0.0204 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B007KFS10O | -0.0471 | -0.0204 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B0081U2VCG | -0.0471 | -0.0204 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.00520199 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B00AB93O5S | -0.0471 | -0.0204 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B00B5505VI | -0.0471 | -0.02048 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B00D5ZLMKE | -0.0471 | -0.0204 | -0.0302 | -0.0420 | -0.0300 | -0.0224 | -0.0563 | -0.0469 | -0.0400 | -0.0104 | -0.0503 | -0.0446 | -0.0308 | -0.0436 | -0.0468 | -0.0052 | -0.0577 | -0.0391 | -0.0443 | -0.0438 |
| B00025FROW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B00062NHGQ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B0010FVNWS | -0.0992 | -0.0693 | -0.0731 | -0.0360 | -0.0801 | -0.0762 | -0.0264 | -0.0647 | -0.0453 | -0.0576 | -0.0588 | -0.0650 | -0.0720 | -0.0624 | -0.0616 | -0.0393 | -0.0659 | -0.0539 | -0.0488 | -0.0802 |
| B004K1KR4E | -0.0992 | -0.0693 | -0.0731 | -0.0360 | -0.0801 | -0.0762 | -0.0264 | -0.0647 | -0.0453 | -0.0576 | -0.0588 | -0.0650 | -0.0720 | -0.0624 | -0.0616 | -0.0393 | -0.0659 | -0.0539 | -0.0488 | -0.0802 |
| B004SDYQI2 | -0.0992 | -0.0693 | -0.0731 | -0.0360 | -0.0801 | -0.0762 | -0.0264 | -0.0647 | -0.0453 | -0.0576 | -0.0588 | -0.0650 | -0.0720 | -0.0624 | -0.0616 | -0.0393 | -0.0659 | -0.0539 | -0.0488 | -0.0802 |
| B00866CKU8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B000SKNYAU | -0.0700 | -0.0539 | -0.0727 | -0.0713 | -0.0787 | -0.0920 | -0.0633 | -0.0581 | -0.0854 | -0.0985 | -0.1013 | -0.0622 | -0.0824 | -0.0400 | -0.0461 | -0.0992 | -0.0569 | -0.0430 | -0.0755 | -0.0557 |
| B00127QC0C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B003TSTS94 | -0.0700 | -0.0539 | -0.0727 | -0.0713 | -0.0787 | -0.0920 | -0.0633 | -0.0581 | -0.0854 | -0.0985 | -0.1013 | -0.0622 | -0.0824 | -0.0400 | -0.0461 | -0.0992 | -0.0569 | -0.0430 | -0.0755 | -0.0557 |
| B004AE1X1C | -0.0700 | -0.0539 | -0.0727 | -0.0713 | -0.0787 | -0.0920 | -0.0633 | -0.0581 | -0.0854 | -0.0985 | -0.1013 | -0.0622 | -0.0824 | -0.0400 | -0.0461 | -0.0992 | -0.0569 | -0.0430 | -0.0755 | -0.0557 |
| B0072T3QTE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 4.5: Item factors related to implicit information denoted with $y_j$ in Eq. 4.7

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user 1 | 0.0822 | 0.2629 | 0.7631 | 0.6840 | 0.3025 | 0.1328 | 0.7877 | 0.3515 | -0.0015 | 0.7541 | 0.1706 | 0.2351 | 0.1593 | 0.0728 | 0.2467 | 0.6672 | 0.6877 | 0.5001 | 0.7393 | 0.7930 |
| user 2 | 0.6424 | 0.5492 | 0.3299 | 0.1598 | 0.6869 | 0.2295 | 0.1986 | 0.0519 | 0.3549 | 0.6328 | 0.3271 | 0.5915 | 0.4710 | -0.0722 | 0.5344 | 0.5301 | -0.1029 | 0.7741 | 0.4161 | 0.7054 |
| user 3 | -0.0530 | 0.7669 | 0.5231 | 0.6359 | 0.7617 | -0.1180 | 0.4554 | 0.5740 | -0.0055 | 0.7951 | 0.3723 | 0.4836 | 0.6092 | 0.0791 | 0.0777 | 0.8633 | 0.7452 | 0.1484 | 0.4426 | 0.2340 |
| user 4 | 0.3072 | 0.2536 | 0.6987 | -0.0445 | 0.6494 | 0.0397 | 0.6141 | -0.0536 | 0.7736 | 0.6069 | 0.0309 | 0.0730 | 0.6275 | 0.1423 | 0.8735 | 0.7397 | 0.3743 | 0.4795 | 0.3760 | 0.7398 |

Table 4.6: User static factors denoted with $\gamma_u$ in Eq. 4.6

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| user 2 | -0.2654 | -0.0772 | -0.0797 | -0.1811 | -0.1462 | -0.1360 | -0.2361 | 0.0866 | -0.2846 | -0.0871 | -0.0337 | -0.1622 | 0.1640 | -0.0924 | -0.0109 | 0.0595 | -0.1484 | -0.1264 | 0.1419 | -0.0532 |
| user 3 | 0.0240 | -0.1111 | -0.1677 | -0.0550 | -0.1590 | 0.0529 | 0.0298 | -0.2385 | 0.2020 | -0.1503 | -0.0899 | -0.2477 | 0.1956 | 0.1834 | -0.1768 | -0.0906 | 0.0195 | -0.0186 | -0.0774 | -0.0673 |
| user 4 | 0.1305 | -0.0069 | -0.1392 | -0.1384 | 0.0657 | 0.0132 | 0.1516 | -0.1088 | -0.1064 | 0.0997 | 0.0270 | 0.1901 | 0.1142 | 0.0821 | -0.1198 | -0.0223 | -0.0967 | 0.0355 | -0.0211 | -0.1195 |

Table 4.7: User time-dependent related parameter $\alpha_{u,k}$, presented in Eq. 4.6

- user 1:

| day | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1502 | -0.1379 | -0.0863 | -0.0725 | -0.1022 | -0.0180 | -0.0775 | -0.0337 | -0.1026 | -0.0813 | -0.1392 | -0.1072 | -0.1221 | -0.0621 | -0.1407 | -0.1677 | -0.1064 | -0.1574 | -0.0517 | -0.1287 | -0.0921 |

Table 4.8: User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 1, presented in Eq.4.6

- user 2:

| day | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1111 | -0.0149 | -0.0138 | -0.0270 | -0.0241 | -0.0140 | -0.0052 | -0.0334 | -0.0826 | -0.0013 | 0.0021 | -0.0641 | -0.0313 | -0.0749 | -0.0434 | -0.0636 | -0.0187 | -0.0524 | -0.0305 | -0.0896 | -0.0512 |
| 1425 | -0.1117 | -0.0415 | -0.0550 | -0.0895 | -0.0670 | -0.0550 | -0.1185 | -0.0468 | -0.1058 | -0.0299 | -0.0734 | -0.0894 | -0.0110 | 0.0753 | -0.0646 | 0.0039 | -0.1043 | -0.0755 | -0.0331 | -0.0683 |

Table 4.9: User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 2, presented in Eq.4.6

- user 3:

| day | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 881 | -0.0691 | -0.0133 | -0.0209 | -0.0244 | -0.0505 | -0.0547 | -0.0011 | -0.0101 | -0.0442 | -0.0079 | -0.0272 | -0.0226 | -0.0664 | -0.0497 | -0.0178 | -0.0155 | -0.0610 | -0.0281 | -0.0073 | -0.0634 |
| 1298 | -0.0369 | -0.0435 | -0.0278 | -0.0016 | -0.0003 | -0.0343 | -0.0365 | -0.0167 | -0.0401 | -0.0290 | -0.0183 | 0.0006 | -0.0434 | -0.0501 | -0.0159 | -0.0101 | -0.0067 | -0.0255 | -0.0340 | -0.0031 |
| 1542 | -0.0841 | -0.0758 | -0.0700 | 0.0299 | -0.0880 | -0.0388 | -0.0121 | -0.0747 | -0.0076 | -0.0549 | -0.0567 | -0.0830 | -0.0204 | -0.0225 | -0.0660 | -0.0225 | -0.0500 | -0.0490 | -0.0375 | -0.0710 |
| 1634 | -0.0141 | -0.0100 | -0.0318 | -0.0181 | -0.0261 | -0.0292 | -0.0046 | -0.0317 | -0.0013 | -0.0268 | -0.0190 | -0.0289 | -0.0180 | -0.0063 | -0.0272 | -0.0328 | -0.0181 | -0.0084 | -0.0215 | -0.0275 |

Table 4.10: User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 3, presented in Eq.4.6

- user 4:

| day | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 | f16 | f17 | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1509 | -0.0969 | -0.0454 | -0.0261 | -0.0251 | -0.0869 | -0.0844 | -0.0967 | -0.0216 | -0.0462 | 0.1135 | -0.0964 | -0.1062 | -0.1034 | -0.0574 | -0.0080 | -0.0811 | -0.0238 | -0.0474 | -0.0606 | -0.0165 |
| 1554 | -0.0258 | -0.0493 | -0.1020 | -0.1006 | -0.0512 | -0.0772 | -0.0142 | -0.0809 | -0.1042 | -0.0593 | -0.0817 | -0.0027 | -0.0412 | -0.0127 | -0.0733 | -0.0934 | -0.0765 | -0.0280 | -0.0722 | -0.0817 |

Table 4.11: User time-dependent related factors $\gamma_{uk,t}$ for the corresponding days to user 4, presented in Eq.4.6

In the following, the set of items rated by each user is provided. Thus, in Table 4.1, the items where users have different value on the rating section than 0 show the items that have been rated by certain user, such that:

- $R(1) = \{B0058VLRIA, B009V06HGG, B00B1VFWDW\}$

- $R(2) = \{B00517ABLA, B00671EMO6, B007KFS10O, B0081U2VCG, B00AB93O5S, B00B5505VI, B00D5ZLMKE\}$
- $R(3) = \{B00062NHGQ, B004K1KR4E, B004SDYQI2, B00866CKU8\}$
- $R(4) = \{B00127QC0C, B004AE1X1C, B0072T3QTE\}$

Having learned the parameter values, now the application of each component from the model can be applied to predict the preference value of given user and item at given point in time.

- $X_{1,B003EAORCA} \underbrace{(1383177600)}_{day\ 1502, bin\ 9} = ?$
  - **item bias** (applying Eq. 4.2)
    $\beta_{B003EAORCA}(1383177600)$ = $\beta_{B003EAORCA} + \beta_{B003EAORCA,bin(9)}$ = 0 + 0 = 0
  - **user bias** (applying Eq. 4.5)
    $\beta_1(1383177600) = \beta_1 + \alpha_1 * dev(1, 1502) + \beta_{1,1502}$
    = $-0.2072 + 0 * \underbrace{(sign(1502 - t1)|1502 - t1|^{0.2})}_{dev(1,1502)} + (-0.2072)$
    = $-0.2072 + 0 * (sign(1502 - 1502/1)|1502 - 1502/1|^{0,2}) + (-0.2072) = -0.4144$,
    where t1 is the mean rating date
  - **user factors** (applying Eq. 4.6)
    $\gamma_{1,1}(1383177600) = \gamma_{1,1} + \alpha_{1,1} * dev(1, 1502) + \gamma_{1,\ \underbrace{1502}_{factor\ 1, day\ 1502}}$
    = $0.0822 + 0 + (-0.1379) = -0.0557$
    $\gamma_{1,2}(1383177600)$ = $\gamma_{1,2} + \alpha_{1,2} * dev(1, 1502) + \gamma_{1,\ \underbrace{1502}_{factor\ 2, day\ 1502}}$
    = 0.0822 + 0 + (-0.0863) = -0.0041
    .
    .
    .
    $\gamma_{1,3}(1383177600) = \gamma_{1,20} + \alpha_{1,20} * dev(1, 1502) + \gamma_{1,\ \underbrace{1502}_{factor\ 20, day\ 1502}}$
    = 0.0822 + 0 + (-0.0921) = -0.0099
  - $|\boldsymbol{R(1)}|^{\boldsymbol{-1/2}} = \boldsymbol{1^{-1/2}} = 1$
  - **factors related to implicit info.**

    $R(1) = \sum_{j \epsilon R(1)} y_j$
    $= \{-0.000504358336, -0.000123505992, -7.3034632e-05, -0.000205379, -1.092727e-06, -8.9314623e-05, -7.414875e-06, -0.000207474688, -0.000103161709, -0.000517781627, -0.000237176659, -0.000348913664, , -4.5882712e-05, -0.000535387328, -0.000907039232, -0.000231475544, -0.000751089429, -2.6463592e-05, -0.000410172407, -0.000149721291\}$

– **predicting the preference value** (applying Eq. 4.7)

$$\begin{aligned} x_{1,B003EAORCA}(t) &= 4.117 + \beta_1(t) + \beta_{B003EAORCA}(t) + \\ &+ \gamma^T_{B003EAORCA}\left(\gamma_1(t) + |R(1)|^{-1/2} \sum_{j \epsilon R(u)} y_j\right) \\ &= 4.117 + 0 + (-0.4144) + 0.0001 \\ &= 3.7027 \end{aligned}$$

- $X_{2,B001RNO3KC}\underbrace{(1376524800)}_{day\ 1425, bin\ 8} =?$
  After applying the same procedure as the one for user 1, the result is the following:
  $x_{2,B001RNO3KC}(t) = 4.3543$
- $X_{3,B00025FROW}\underbrace{(1299110400)}_{day\ 529, bin\ 3} =?$
  After applying the same procedure as the one for user 1, the result is the following:
  $x_{3,B00025FROW}(t) = 5.0000$
- $X_{4,B000SKNYAU}\underbrace{(1310860800)}_{day\ 665, bin\ 4} =?$
  After applying the same procedure as the one for user 1, the result is the following:
  $x_{4,B000SKNYAU}(t) = 5.0000$

As we can see from the example above, user's 1 preference value on day 1502 against item $B003EAORCA$ shows a preference above the average. With regard to item bias, as item $B003EAORCA$ did not have any interaction with any other user on the training set, the bias value remained 0 (as initialized in the beginning). Regarding the user 1 bias, user 1 has been active only one day, such that not any big change with regard to the evolution of his preferences can be captured. Moreover, the day-related parameter $\gamma_{1,1502}$ could be associated to shorter periods of time than the lasting of one day (e.g.breakfast time, lunch time, etc.), and thus we would be able to capture those small dynamics (but in such context, more data would be needed).

## 4.2 TVBPR+

This section is based on [4]. *TVBPR+* is one-class collaborative filtering for modeling the evolution of fashion trends. The evolution of fashion trends is learned by uncovering the factors that are considered to have impact on users' opinions at given period of time. *TVBPR+* is applied on the data provided by Amazon, and has shown to provide promising results. *TVBPR+* adapts the 'basic' formulation of matrix factorization as well, where the preference of the user towards an item is predicted by using formulation 4.1 above, which has proven to capture rich interaction between users and items. But, considering the context where the data is really sparse (only few items get rated), as described in [4], the formulation above suffers in cold-start situations. Thus, involving other source of information about users or items helps dealing with such an issue. Regarding *TVBPR+*, as the focus is on uncovering the visual dimensions from the data being at play, they use the information about the visual characteristics of the item, and thus being able to uncover the preferred visual style for each user. The formulation that involves the visual characteristics of the items looks as follows:

$$\hat{x}_{u,i} = \alpha + \beta_u + \beta_i + \langle \gamma_u, \gamma_i \rangle + \langle \theta_u, \theta_i \rangle, \tag{4.8}$$

where $\theta_u$ represents the visual factors characterizing user $u$, whereas $\theta_i$ represents the visual factors describing item $i$.

The Deep CNN has been used to extract the features from the images of the items. The Deep CNN features has demonstrated to capture visual dimensions that resemble to the human notions of the visual style. Such that, an embedding matrix $E$ has been introduced to linearly embed the high−dimensional features extracted from Deep CNN representing items into a $K'$ lower−dimensional visual space. Let $f_i$ denote the features of item $i$ extracted from the Deep CNN, and $F$ the number of dimensions representing the item $i$. Then, $\theta_i$ is formulated as follows:

$$\theta_i = Ef_i \tag{4.9}$$

To model the temporal dynamics that are at play with regard to the fashion data, three types of temporal dynamics are considered:

- temporally visual factors
- temporally evolving visual bias
- non-visual temporal dynamics

### 4.2.1 Temporally Visual Factors

Regarding the temporally-evolving visual factors, here the focus is on modeling the visual dimensions being time-aware. This is based on the fact that, as new items come in new visual dimensions might come at play. Thus, the embedding matrix representing the visual space should be time-aware and change with new data coming in accordingly. Such that, the following formulation represents the time-dependent embedding matrix *E*:

$$E(t) = E + \Delta_E(t) \tag{4.10}$$

The formulation above captures the 'attractiveness' of items as some items might lose or gain it. Considering the formulation of the time−dependent embedding matrix $E$, the visual factors of item $i$ at given point in time $t$ are formulated as follows:

$$\theta_i(t) = E(t)f_i \tag{4.11}$$

Considering that visual dimensions are weighted differently by different users, a weighting vector is proposed. Such that, the formulation of $\theta_i$ looks as follows:

$$\theta_i(t) = Ef_i \odot \omega(t), \tag{4.12}$$

where $\odot$ refers to Hadamard product.
If all the formulas above are combined, the following formulation follows:

$$\theta_i(t) = \underbrace{Ef_i \odot \omega(t)}_{\text{base}} + \underbrace{\Delta_E(t)f_i}_{\text{deviation}} \tag{4.13}$$

The formulations above capture only the global drifts that are shared by all users, and thus we can capture the fashion evolution with respect to the entire population. But, considering the personal tastes, the

user $u$ at time $t$ is modeled in the same way as modeled in *timeSVD++*. Such that, the similar formulation to *timeSVD++* above for capturing user bias is used:

$$\theta_u(t) = \theta_u + sign(t - t_u) \cdot |t - t_u|^k \eta_u \tag{4.14}$$

### 4.2.2 Temporally Evolving Visual Bias

A visual bias term is also introduced, which captures the biases for each dimension in the visual space of K' dimensions. As stated on [4], it captures "the portion of the variance which is common to all factors". The proposed formulation for the visual bias is the following:

$$\beta(t) = \beta \odot b(t) + \Delta_\beta(t) \tag{4.15}$$

Having formulated the visual bias, then the visual bias of item $i$ at time $t$ is calculated by using the following inner product $\langle \beta(t), f_i \rangle$.

### 4.2.3 Non-visual Temporal Dynamics

Besides the dynamics of the visual dimensions, also non-visual dynamics might be at play. Non-visual dynamics involve: sales, promotions, etc. In order to capture such dynamics, two approaches has been proposed. One approach provides capturing per-item level dynamics, whereas the other provides capturing per-subcategory dynamics.

Per-item level dynamics is formulated by making the static bias item term $\beta_i$ being time-dependent. Regarding the per-subcategory dynamics, a new parameter is introduced, $\beta_{C_i}(t)$, which makes use of subcategory information, and thus checking if there is any drifting behavior of a user u with regard to the subcategory of item i falls into.

At the end, combining all the components together, the final model looks as follows:

$$\hat{x}_{u,i} = \alpha + \beta_u + \beta_i(t) + \beta_{C_i}(t) + \langle \beta(t), f_i \rangle + \langle \gamma_u, \gamma_i \rangle + \langle \theta_u(t), \theta_i(t) \rangle \tag{4.16}$$

### 4.2.4 Fashion Segmentation

Having already described the parameters involved in the model, now the way learning them such that the involved dynamics in data are captured is a challenging part. As described on [4], some experiments have been performed by trying similar approaches as the ones proposed for *timeSVD++*, but those approaches have not been expressive enough for capturing fashion dynamics. Thus, as the task is to show the evolution of fashion trends, such that uncovering the visual dimensions and finding the partition of the dataset that most accurately shows that certain visual dimensions are predominating, time-windows design has been more appropriate one for learning the parameters.

### 4.2.5 Learning the Model

For learning the model, the BPR (Bayesian Personlized Ranking) [9] state of the art ranking optimization framework has been used. It accounts both positive and negative items to learn the preferences of the users. Thus, a ranking of items that users have not interacted with is provided for each user - BPR considers only the negative items when a personalized ranking is provided for the users, such that to the users are recommended only the items they have not interacted with before. Basically, the training set used for BPR consists of triples of the following form $(u, i, j)$, where $i \in P_u$ and $j \in I \setminus P_u$, with $P_u$ being the set of items user $u$ showed positive feedback to. Thus, given a training tuple, BPR models the probability that user *u* prefers item *i* over item *j* with $\sigma(\hat{x}_{u,i} - \hat{x}_{u,j})$, where $\sigma$ is the sigmoid function and $\hat{x}_{u,i}$ is the

predicted preference of user *u* towards the positive item *i* and $\hat{x}_{u,j}$ is the predicted preference of user *u* towards the negative item *j*. Then, the parameters are learned by maximizing the following regularized log-likelihood function:

$$\sum_{(u,i,j)\in D_S} \log \sigma(\hat{x}_{u,i} - \hat{x}_{u,j}) - \frac{\lambda_\Theta}{2}||\Theta||^2 \tag{4.17}$$

As we deal with learning the parameters for different epochs or partitions of the dataset, the formulation above takes the following form:

$$\hat{\Theta}, \hat{\Lambda} = \underset{\Theta,\Lambda}{\operatorname{argmax}} \sum_{(u,i,j,t_{u,i})\in D_S} \log \sigma(\hat{x}_{u,i}(t) - \hat{x}_{u,j}(t)) - \frac{\lambda_\Theta}{2}||\Theta||^2 \tag{4.18}$$

As it can be seen, now we deal with two components that are needed to fit to the training set, in order to maximize the function 4.18. One of the component is fitting the parameter set to the training set, and the other one is finding the optimal segmentation of N given epochs. Such that, the first step is to fit the parameters to the training set. For doing this, the stochastic gradient ascent has been used. And, given quadruple $(u, i, j, t_{ui})$ , the update is formulated as follows:

$$\Theta \leftarrow +\epsilon \cdot (\sigma(-\hat{x}_{uij}(ep(t_{ui})))\frac{\partial \hat{x}_{uij}(ep(t_{ui}))}{\partial \Theta} - \Lambda_\Theta \Theta), \tag{4.19}$$

where $\hat{x}_{uij}(ep(t_{ui})) = \hat{x}_{u,i}(ep(t_{ui})) - \hat{x}_{u,j}(ep(t_{ui}))$

The second step, fitting the epoch segmentation respectively, involves operation that finds the optimal segmentation of the training set, such that the best performance is achieved. For doing this, firstly partitioning of the timeline is applied into N equal size bins. Afterwards, using dynamic programming, the optimal segmentation of the timeline is calculated. As the ranking quality is evaluated by considering all non-observed items for each positive item, it is time-consuming when dealing with a large dataset, such that it is considered as a naive approach. But, as proposed and turned out to be correct, finding the optimal segmentation by considering a small amount of non-observed items (1000 non-observed items) for each positive item has shown to approximate the full log-likelihood.

The two steps above are repeated until the convergence is achieved.

### 4.2.6 Example

In this section, an example of the application of *TVBPR* (which does not make use of the non-visual item bias being time-dependent compared to *TVBPR+*), is shown. For this experiment, the same input as the one used for *timeSVD++* example is used, which is shown in Table 4.1.

For applying *TVBPR*, firstly the parameters are initialized with some initial random values in a range of (0,1). After having the initialization performed, the learning process related to *TVBPR+* described above is applied. Since the focus is on how to apply the formulations comprising *TVBPR+*, we will show the learned parameter values after the training process, and then the necessary calculations for predicting the preference values of the users toward items.

The learned values of the parameters grouped as non-visual and visual parameters are the following:

- Non-visual Parameters

| | $\beta_i$ | f1 | f2 | f3 | $\cdots$ | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|---|
| B003EAORCA | -0.126497080523 | 0.79291628 | 0.0942274 | 0.12946723 | $\cdots$ | 0.00450682 | 0.23187143 | 0.59745123 |
| B0058VLRIA | 0.0360332618834 | 0.44927379 | 0.27674382 | 0.07707751 | $\cdots$ | 0.00750875 | 0.47936094 | 0.50171547 |
| . | | | | | | | | |
| . | | | | | | | | |
| . | | | | | | | | |
| B004AE1X1C | 0.056327757216 | 0.54971403 | 0.23841549 | 0.47751374 | $\cdots$ | 0.50562419 | 0.50174873 | 0.39999973 |
| B0072T3QTE | 0.105598827606 | 0.04724386 | 0.46893825 | 0.35245932 | $\cdots$ | 0.27777953 | 0.30072391 | 0.15983745 |

Table 4.12: Item Bias and Factors, $\beta_i$ and $\gamma_i$ respectively

| | f1 | f2 | f3 | $\cdots$ | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|
| user 1 | 0.11456169 | -0.00334641 | 0.05491303 | $\cdots$ | -0.01743169 | 0.1317918 | 0.1086825 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| user 4 | 0.06634314 | 0.14788008 | 0.21030185 | $\cdots$ | 0.14978394 | 0.05361435 | 0.13154633 |

Table 4.13: User factors, $\gamma_i$ respectively

- Visual Parameters

| | f1 | f2 | f3 | $\cdots$ | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|
| user 1 | -0.02821587 | -0.02273798 | -0.03115407 | $\cdots$ | -0.03079309 | -0.02479417 | -0.02862124 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| user 4 | -0.02776621 | -0.02219125 | -0.02451551 | $\cdots$ | -0.02617196 | -0.02854347 | -0.02287346 |

Table 4.14: User visual factors, $\theta_u$ respectively

Table 4.15: The static embedding matrix E

| | feat. 1 | feat. 2 | feat. 3 | ... | feat. 4094 | feat. 4095 | feat. 4096 |
|---|---|---|---|---|---|---|---|
| dim. 1 | 0.58330717 | 0.52471568 | 0.67077409 | ... | 0.13782374 | 0.82254404 | 0.60793853 |
| dim 2 | 0.32496333 | 0.96775872 | 0.12271576 | ... | 0.40672053 | 0.85121154 | 0.91168397 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| dim. 19 | 0.78688158 | 0.301584 | 0.12155727 | ... | 0.74460072 | 0.75458812 | 0.01627602 |
| dim. 20 | 0.58541302 | 0.53279108 | 0.22762106 | ... | 0.98218826 | 0.58276142 | 0.71937913 |

| | feat. 1 | feat. 2 | feat. 3 | ... | feat. 4094 | feat. 4095 | feat. 4096 |
|---|---|---|---|---|---|---|---|
| dim. 1 | 0.00766428 | 0.26873741 | 0.04666985 | ... | 0.06897313 | 0.27515169 | 0.21383483 |
| dim. 2 | 0.72086256 | 0.83005209 | 0.11380367 | ... | 0.95005043 | 0.42140941 | 0.54632146 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| dim. 19 | 0.59722358 | 0.68681844 | 0.86969353 | ... | 0.2340534 | 0.18622787 | 0.29477795 |
| dim. 20 | 0.58806162 | 0.79240562 | 0.1954462 | ... | 0.2967964 | 0.45799249 | 0.06106239 |

Table 4.16: The embedding matrix E(t) at epoch 1

| | feat. 1 | feat. 2 | feat. 3 | ... | feat. 4094 | feat. 4095 | feat. 4096 |
|---|---|---|---|---|---|---|---|
| dim. 1 | 0.69727987 | 0.14991916 | 0.74905501 | ... | 0.22856041 | 0.984096 | 0.71723818 |
| dim. 2 | 0.47607473 | 0.06242676 | 0.31915043 | ... | 0.43664639 | 0.12154159 | 0.61208573 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| dim. 19 | 0.1482202 | 0.98006643 | 0.11739262 | ... | 0.62164919 | 0.73375898 | 0.42174419 |
| dim. 20 | 0.96662511 | 0.86619557 | 0.96428079 | ... | 0.6678679 | 0.95800838 | 0.43764076 |

Table 4.17: The embedding matrix E at epoch 10

| | f1 | f2 | f3 | ··· | f18 | f19 | f20 |
|---|---|---|---|---|---|---|---|
| epoch 1 | 0 | 0 | 0 | ··· | 0 | 0 | 0 |
| epoch 2 | 0 | 0 | 0 | ··· | 0 | 0 | 0 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| epoch 9 | -0.0330897 | -0.03301007 | -0.03064873 | ··· | -0.03312334 | -0.03137209 | -0.03511399 |
| epoch 10 | -0.04046583 | -0.05153918 | -0.08073335 | ··· | -0.05576696 | -0.06817314 | -0.04643101 |

Table 4.18: The visual factor weighting vector - $\omega(t)$

Table 4.19: Visual bias - related parameters

| | f1 | f2 | f3 | ... | f4 | f5 | f6 |
|---|---|---|---|---|---|---|---|
| $\beta$ | 0.05594762 | 0.7334017 | 0.55102493 | ... | 0.81667302 | 0.00930684 | 0.9288046 |
| epoch 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| . | | | | | | | |
| . | | | | | | | |
| . | | | | | | | |
| epoch 9 | 2.15334724e-04 | -1.43662100e-03 | -7.41753571e-03 | ... | -6.16337720e-03 | 4.30688062e-05 | 1.10145173e-02 |
| epoch 10 | -2.89831271e-04 | -1.75205729e-04 | -7.83686021e-03 | ... | 7.99366815e-03 | -4.40916704e-05 | -1.65295899e-02 |

| | f1 | f2 | f3 | ... | f4 | f5 | f6 |
|---|---|---|---|---|---|---|---|
| epoch 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| epoch 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| .<br>.<br>. | | | | | | | |
| epoch 9 | 0.00384893 | -0.00195887 | -0.01346312 | ... | -0.00754734 | 0.00462774 | 0.01186071 |
| epoch 10 | -0.00518049 | -0.00023889 | -0.01422415 | ... | 0.00978863 | -0.00473767 | -0.01779932 |

Table 4.20: The visual bias weighting vector - $b(t)$

Now, having already the learned values, we can apply the necessary formulations of *TVBPR* to predict the preference values of the users toward certain items. In order to get the $\beta(t)$ (visual bias) and $\theta_{i(t)}$ for the corresponding items, the formulation in Eq. 4.15 and the formulation in Eq. 4.13 are applied respectively. In our example, the prediction of the preference value of user 1 towards item *B003EAORCA* at timestamp of *1383177600* (which corresponds to epoch 9) is considered. Firstly, the calculation that differ from *timeSVD++* are shown. Such that, for getting the visual biases at given epoch, the following calculation is performed:

- $\langle \beta(t), f_i \rangle$

$$\begin{aligned}
\beta(1383177600) &= \Big\langle (\beta \odot b(1383177600) + \Delta_\beta(1383177600)), f_{B003EAORCA} \Big\rangle \\
&= \Big\langle ( \begin{pmatrix} 0.05594762 & \cdots & 0.9288046 \end{pmatrix} \odot \begin{pmatrix} 2.15334724e-04 & \cdots & 1.10145173e-02 \end{pmatrix} \\
&+ \begin{pmatrix} 0.00384893 & \cdots & 0.01186071 \end{pmatrix} ), f_{B003EAORCA} \Big\rangle \\
&= 0.0839220950793
\end{aligned}$$

Whereas in order to get the visual factors for the corresponding epoch of the item *B003EAORCA*, the following calculation is performed:

- $\theta_i(t)$

$$\theta_{B003EAORCA}(1383177600) = Ef_{B003EAORCA} \odot \omega(1383177600) + \Delta_E(1383177600) f_{B003EAORCA}$$

$$= \underbrace{\begin{pmatrix} 0.58330717 & \cdots & 0.60793853 \\ \vdots & \ddots & \vdots \\ 0.58541302 & \cdots & 0.71937913 \end{pmatrix}}_{E} f_{B003EAORCA} \odot \underbrace{\begin{pmatrix} -0.0330897 & \cdots & -0.03511399 \end{pmatrix}}_{\omega(t)}$$

$$+ \underbrace{\begin{pmatrix} 0.40736556 & \cdots & 0.306776 \\ \vdots & \ddots & \vdots \\ 0.45930571 & \cdots & 0.96964655 \end{pmatrix}}_{\Delta_E(t)} f_{B003EAORCA} = \underbrace{\begin{pmatrix} 12.51976488 & \cdots & 13.04875802 \end{pmatrix}}_{\theta_i}$$

Afterwards, in order to calculate the preference of user 1 towards item *B003EAORCA*, the following calculation is performed:

- $x_{u,i}(t)$
  $X_{1,B003EAORCA}(1383177600)$
  $= \beta_{B003EAORCA} + \underbrace{\langle \beta(t), f_{B003EAORCA} \rangle}_{visual\ bias} + \underbrace{\langle \gamma_1, \gamma_{B003EAORCA} \rangle}_{non-visual\ factors}$
  $+ \underbrace{\langle \theta_1(t), \theta_{B003EAORCA}(t) \rangle}_{visual\ factors} = -0.126497080523 + \underbrace{0.0839220950793}_{after\ applying\ Eq.\ 4.15}$
  $+ \underbrace{\left( \begin{pmatrix} 0.11456169 & \cdots & 0.1086825 \end{pmatrix} \begin{pmatrix} 0.79291628 & \cdots & 0.59745123 \end{pmatrix} \right)}_{non-visual\ factors}$
  $+ \underbrace{\left( \begin{pmatrix} 0.11456169 & \cdots & 0.1086825 \end{pmatrix} \begin{pmatrix} 0.79291628 & \cdots & 0.59745123 \end{pmatrix} \right)}_{visual\ factors}$
  $= (-0.126497080523) + 0.0839220950793 + 0.96314291182201561 + (-7.2675641472693639)$
  $= -6.346996220891048$

The same procedure is followed for predicting the preferences of other users toward other items.

As we can see from the calculation above, we do not consider the offset value of the model and user bias (which was considered when *timeSVD++* was applied). As proposed on [4], we deal with an item recommendation task, such that making it unnecessary of using those parameters (whereas, *timeSVD++* is designed for a rating prediction task). If we consider the predicted preference value of the user 1 against item $B003EAORCA$, we deal with a bigger absolute value of the result compared to *timeSVD++*, as a result of the image features being involved (e.g. the visual interaction score between user 1 and item $B003EAORCA$ is around -7.2675, whereas the non-visual interaction score is around 0.9631).

Regarding the time-dependent parameters being involved, *TVBPR* models the time-dependent characteristics by mostly considering the visual aspects, and such that considering the non-visual user-item interaction between user and item as static one - where this part is defined as a time-dependent one with regard to *timeSVD++*.

# 5

# TimeSVD++ with the Visual Component Included

After the introduction of two reference models in Chapter 4, we introduce in this Chapter a new model to predict fashion using visual components. The introduced model is a combination of the two models presented in Chapter 4, which is, an extended version of *timeSVD++* with the visual component of *TVBPR*. From now on, we will refer at the newly introduced model as *timeSVD*$_{VC}$. Such that, in the following section, a detailed description for *timeSVD*$_{VC}$. As *timeSVD*$_{VC}$ involves the same calculations (with respect to the parameters used from other models) as the ones used for *timeSVD++* and *TVBPR*, an example for *timeSVD*$_{VC}$ is not provided.

## 5.1 Description

As *timeSVD*$_{VC}$ is an extension of *timeSVD++*, it makes use of all parameters that comprise *timeSVD++*. Thus, the item bias and user bias formulations shown in Eq. 4.2 and the formulation in Eq. 4.5 respectively are used as baseline predictors. Regarding the component capturing the interaction between the user and item, the component from *timeSVD++* formulated as follows is used:

$$\gamma_i^T \left( \gamma_u(t) + |R(u)|^{-1/2} \sum_{j \epsilon R(u)} y_j \right) \tag{5.1}$$

Whereas, with regard to the additionally introduced component, the visual-related parameters presented in *TVBPR+* are used, which is formulated as follows:

$$\underbrace{\langle \theta_u(t), \theta_i(t) \rangle}_{\text{temporal visual interaction}} + \underbrace{\langle \beta_u(t), f_i \rangle}_{\text{temporal visual bias}} \tag{5.2}$$

All the visual parameters from above are formulated in the same way as presented in *TVBPR+*, such that:

- $\theta_i(t) = \underbrace{Ef_i \odot \omega(t)}_{\text{base}} + \underbrace{\Delta_e(t) f_i}_{\text{deviation}}$

- $\theta_u(t) = \theta + sign(t - t_u) \cdot |t - t_u|^k \eta_u$
- $\beta(t) = \beta \odot b(t) + \Delta_\beta(t)$

After extending the *timeVDD++* model with the visual component shown in Eq. 5.2, the newly introduced model is formulated as follows:

$$\hat{x}_{u,i} = \alpha + \beta_u(t) + \underbrace{\beta_i(t)}_{\text{temporal non-visual bias}} + \underbrace{\gamma_i^T \left( \gamma_u(t) + |R(u)|^{-1/2} \sum_{j \epsilon R(u)} y_j \right)}_{\text{temporal non-visual interaction}} + \underbrace{\langle \theta_u(t), \theta_i(t) \rangle}_{\text{temporal visual interaction}} + \underbrace{\langle \beta_u(t), f_i \rangle}_{\text{temporal visual bias}} \tag{5.3}$$

**Fashion Segmentation** In order to model the time-dependent parameters in such a way that the model would be able to capture the signals that influence users' opinions, or signals that indicate change in the characteristics of products being trendy at given point in time, a very "clear" segmentation of the dataset should be provided. As described on [4], fashion tends to move in an abrupt manner and unexpected way, thus being harder to capture those unexpected changes with just simple fixed-size split of the dataset. Thus, the same approach as the one used on *TVBPR+* is applied on *timeSVD*$_{VC}$.

**Learning the Model** For learning the *timeSVD*$_{VC}$ model, different approaches have been tested (see Chapter 7). The chosen approach is the one that is used for *TVBPR+* as well, as shown in Eq. 4.19.

# 6
# Experiments

In order to see the power of the models, different experiments were performed. In this chapter, we show the performed experiments and their results. Firstly, a description of the datasets where the experiments are applied on is given. Then, an experiment is performed which aims to give a 'feeling' about the performance of the models, such that it involves observing the performance of the models while using the same hyper-parameter values, and thus giving an overview why certain model performs better than the other one in given context. Afterwards, in order to compare the models, several tasks are defined, such that the models are tested in different settings.

Regarding the first task, considering the fact that the way how the dataset is split for learning the time-dependent parameters is important as it influences the performance of the model, this task involves testing the models while the number of epochs is changing. Assuming that in fashion context the items are characterized by more complex features, the number of factors or characteristics should play an important role on the performance of the model. Thus, the second tasks involves testing the models while setting the number of factors to different values. Then, with regard to the third task, it involves testing the models while the number of the visual factors is set to different values, i.e. thus these experiments involve only the models that include the visual component, *TVBPR* and *timeSVD*$_{VC}$ respectively. Afterwards, in order to see how the performance of the models is affected by dealing with different sizes of the dataset, a new set of experiments is provided where different datasets are considered for each model. Additionally, an exploratory study is presented, where visually are shown the items representing the visual dimensions captured by the models which involve the visual component.

The experiments above show only the performance with regard to the accuracy of the models, and not showing the performance with regard to the efficiency - the running time each model needs to learn the parameter values. Thus, the following experiments involve tests which show the running time each model needs for learning the parameter values.

For all the experiments, the number of iterations is set to 100. Whereas, other parameters are varying according to the experiment being considered. Regarding the evaluation of the models, the dataset is split into training set, testing set and validation set. Leave-two-out cross validation technique has been used to split the data. For our experiments, the evaluation is performed only by considering the items from the testing set. Note, regarding the *TVBPR+* model, the version that has been used for the experiments is *TVBPR*, which doesn't make use of making the non-visual item bias parameter time-dependent. Regarding

the performance of the machine where the experiments were run, its characteristics are provided in the following:

- Processor: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
- RAM: 16GB

## 6.1 Dataset

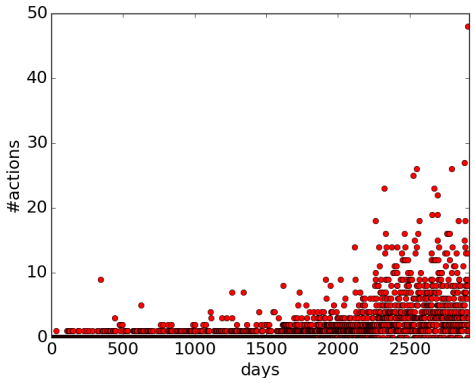
The dataset used for the experiments is the one provided by Amazon. The whole dataset provides product reviews of categories such as: clothing, shoes and jewelry. Using the tools provided in [4], we were able to extract different categories of products, which are the following: women, men, girls and babies. For our experiments, the product reviews of men category have been used. Moreover, from the men category datasets, different subsets of data have been created. Firstly, a dataset is created which involves the users who have rated at least 40 items. Then, four more datasets have been created from the same dataset (i.e. users having at least 40 actions), by reducing the number of actions, and setting them to certain number of actions. The statistics of the created datasets are provided in Table 6.1 below:

Table 6.1: Dataset statistics (after processing)

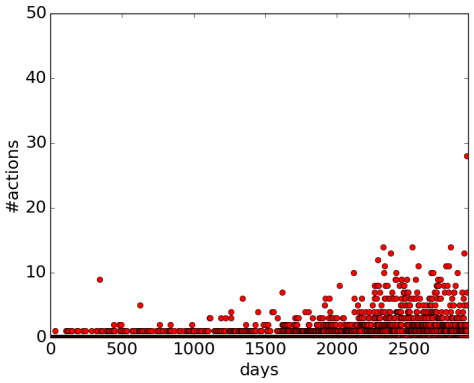
| Dataset | #users | #items | #actions | #actions/user | Timespan |
|---|---|---|---|---|---|
| Dataset1 | 78 | 4099 | 4136 | $\geq 40$ | Jul. 2006 – Jul. 2014 |
| Dataset2 | 78 | 2196 | 2325 | 30 | Jul. 2006 – Jul. 2014 |
| Dataset3 | 78 | 1460 | 1549 | 20 | Jul. 2006 – Jul. 2014 |
| Dataset4 | 78 | 722 | 774 | 10 | Jul. 2006 – Jul. 2014 |
| Dataset5 | 78 | 359 | 388 | 5 | Jul. 2006 – Jul. 2014 |

As we can see from the Table 6.1 above, the number of items is much bigger than the number of users, and the number of actions is close to the number of items. This means that only few items get rated more than once, as new items are constantly coming in. For example, considering the Dataset1 from Table 6.1 (from where the other datasets are extracted), after an analysis, we saw that there are only few items which have been rated more than once. Such that, 161 items out of 4099 items are rated more than once, where 27 out of 161 items have more than two ratings.

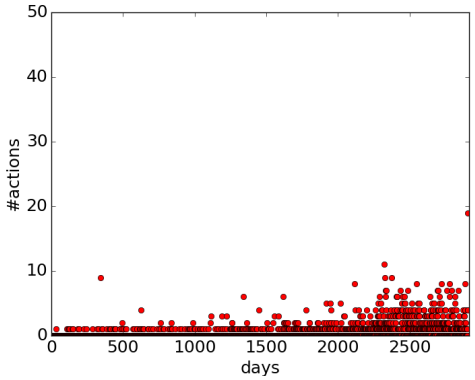
In order to have a better overview of how the data are spread out over time, in Figure 6.1 some graphs are provided.



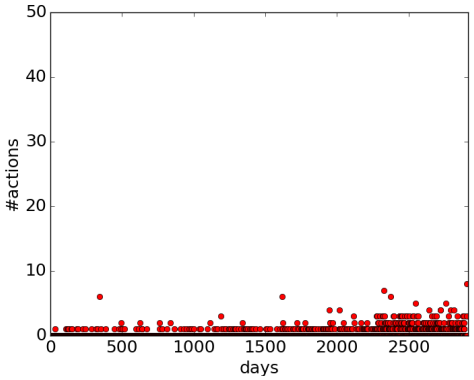

(a) Dataset1 - #actions per day.



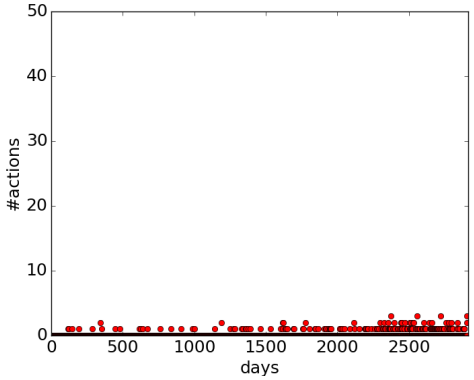

(b) Dataset2 - #actions per day.




(c) Dataset3 - #actions per day.




(d) Dataset4 - #actions per day.




(e) Dataset5 - #actions per day.

Figure 6.1: The evolution of the number of ratings over time.

The plots in Figure 6.1 show , for each dataset, the actions of users over time on daily basis - from the action given on the first day, until the action given on the last day for the dataset that is being considered.

### 6.1.1 A Single Context Showing the Performance of the Models

As described above, in this part, we show a single context where each model is tested. This experiment aims to give an overview of why one model performance better than the other one.

Each model was tested using the following setting:

- Dataset used: Dataset1
- # of non-visual factors: K = 20
- # of visual factors: K' = 20
- # of Epochs: epochs = 10
- # of iterations: iterations = 100

After running the experiments, the performance of the models was the following:

- *timeSVD++*: AUC = 0.6090
- *TVBPR*: AUC = 0.6830
- *timeSVD*$_{VC}$: AUC = 0.8051

As we can see from the results above, *timeSVD++* is showing the worst results compared to other models. Whereas, the model which showed to provide a better performance is *timeSVD*$_{VC}$. The reason of having such results are described in the following. As *timeSVD++* is designed for rating prediction task, its performance is poor when applied to an item recommendation task. Moreover, *timeSVD++* is trained by considering only the positive items from the training set (compared to two other models, where both positive and negative items are considered for training the parameter values). With regard to the performance of*TVBPR* and *timeSVD*$_{VC}$, the second showed a better performance for the fact that more parameters are involved related to the parameters describing the non-visual factors of the users (thus having more impact on maximizing the difference between positive items and negative items). In order to prove this, an experiment has been performed where the parameters which describe the non-visual factors of user-item interactions were reduced, such that the *timeSVD*$_{VC}$ gets the following form:

$$\hat{x}_{u,i} = \alpha + \beta_u(t) + \underbrace{\beta_i(t)}_{\text{temporal non-visual bias}} + \underbrace{\langle \gamma_i, \gamma_u(t) \rangle}_{\text{temporal non-visual interaction}} + \underbrace{\langle \theta_u(t), \theta_i(t) \rangle}_{\text{temporal visual interaction}} + \underbrace{\langle \beta_u(t), f_i \rangle}_{\text{temporal visual bias}} \quad (6.1)$$

As we can see from the Eq. 6.1, this version of the model does not make use of the parameters related to the implicit information (i.e. the set of items rated by certain user). Having reduced the number of the parameters of *timeSVD*$_{VC}$, the performance of the model reduced as well. Such that, AUC = 0.7898.

## 6.2 Accuracy - Comparisons and Analysis of the Performance of the Models

In this section, we show the performances of the models in different settings with regard to their accuracy (AUC value), thus being able to see in which situation certain model is performing better, and in which situation the performance of the model is worsen. As mentioned above, certain tasks are defined, in order to measure the accuracy of the models. In the following, we show the dataset (from the datasets described in Table 6.1) that is being used for each defined task:

- Task 1: Dataset1 is used
- Task 2: Dataset1 is used
- Task 3: Dataset1 is used
- Task 4: Dataset1, Dataset2, Dataset3, Dataset4 and Dataset5 are used

### 6.2.1 1st Task

In this task, the performance of each model is evaluated by setting the number of epochs to different values. Considering the fact that as the number of epochs is increasing, the number of actions(user u rated item i is considered as single action) within epochs is decreasing, we should not expect improvements in the performance of the models.

Figure 6.2 shows how the performance of each model varies while the number of epochs is changing.

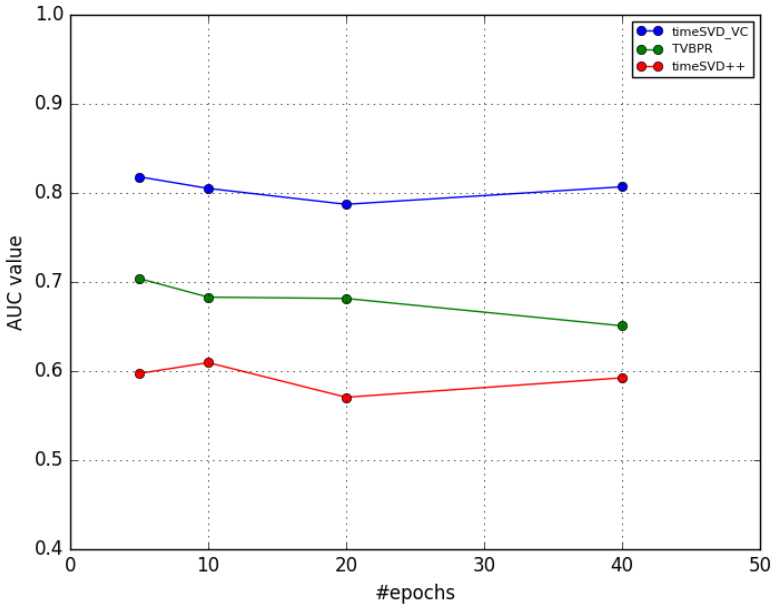



Figure 6.2: The performance of the models over different number of epochs.

As we can see in Figure 6.2, with regard to *TVBPR* and em *timeSVD*$_{VC}$ model, a better performance is achieved when the number of epochs is set to 5. The reason of having such results is for the fact that, we deal with small dataset. The time-dependent parameters get it harder to capture the right effects occurring on the data, since splitting the dataset and learning the time-dependent parameters for certain partition of the dataset which does not have enough data results in poor performance of the model (as proven with our results, and thus having better results with a smaller number of epochs). This can be proven if we run the same model over different sizes of the datasets, but having the number of epochs set to the same value. This is shown in Task 4, Figure 6.6, where the results of each model run over different sizes of the datasets are shown. For example, *TVBPR* and *timeSVD*$_{VC}$ showed an improvement of the accuracy while more data were considered.

Figure 6.3 shows how the dataset is split while different number of epochs is used, and thus being able to see the volume of the data in each epoch for given number of epochs.

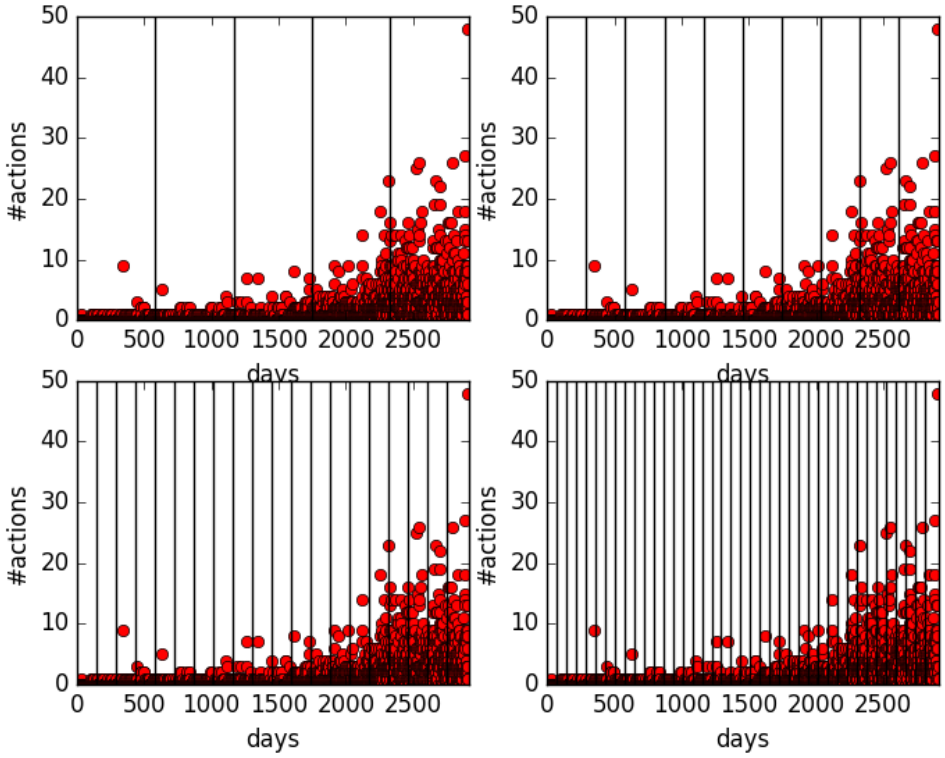



Figure 6.3: The distribution of data across different number of epochs.

### 6.2.2 2nd Task

As already described above, since we deal with fashion-related data, we can assume that the products in such context can be characterized by more complex factors. Such that, our assumption is that, increasing the number of factors helps on characterizing better the items, and thus resulting in having a better performance of the model. This was proven to work in the context of movie data, where the performance of *timeSVD++* was improving as the number of factors was increasing [7]. Such that, the second task involves experiments where the number of non-visual factors is set to different values.

Figure 6.4 shows the performance of each model while the number of non-visual factors is changing.

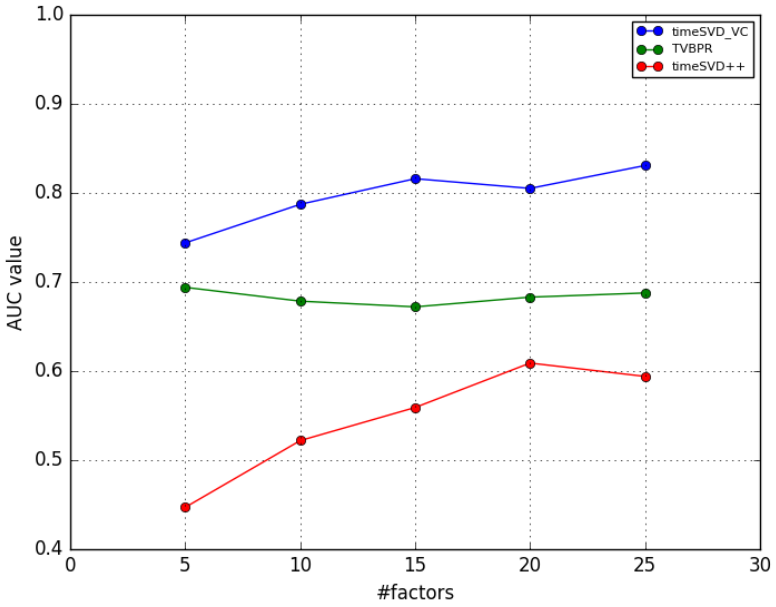



Figure 6.4: The performance of the models while the number of non-visual factors is changing.

As we can see from the Figure 6.4 above, a better effect on the performance caused as a consequence of changing the number of non-visual factors can be seen in the *timeSVD++* model and the *timeSVD*$_{VC}$ model. This is for the fact that, *timeSVD++* and*timeSVD*$_{VC}$ model involve more parameters for describing users' non visual characteristics compared to *TVBPR* (where the non-visual factors are set to static, thus less parameters needed). Additionally, as *timeSVD++* does not involve the visual component, it makes use of only the ratings as input data, and thus uncovers the factors accordingly by not making use of other source of information. Thus, the effect of changing the number of non-visual factors is more emphasized in the *timeSVD++* model. Whereas, with regard to *TVBPR*, its performance is not affected as the other models being affected.

### 6.2.3 3rd Task

The third set of experiments comes from the same idea used for the second task. Such that, as mentioned above, it involves experiments where the performance of the models is observed while the number of visual factors is changing. As only the *TVBPR* model and *timeSVD*$_{VC}$ include the visual component, these experiments consider only those models. Here, our assumption is that, we do not expect to have a better performance while the number of visual factors is increased, but instead, finding a better representation of the visual space which is "described" by the visual dimensions.

Figure 6.5 shows the performance of both models mentioned above:

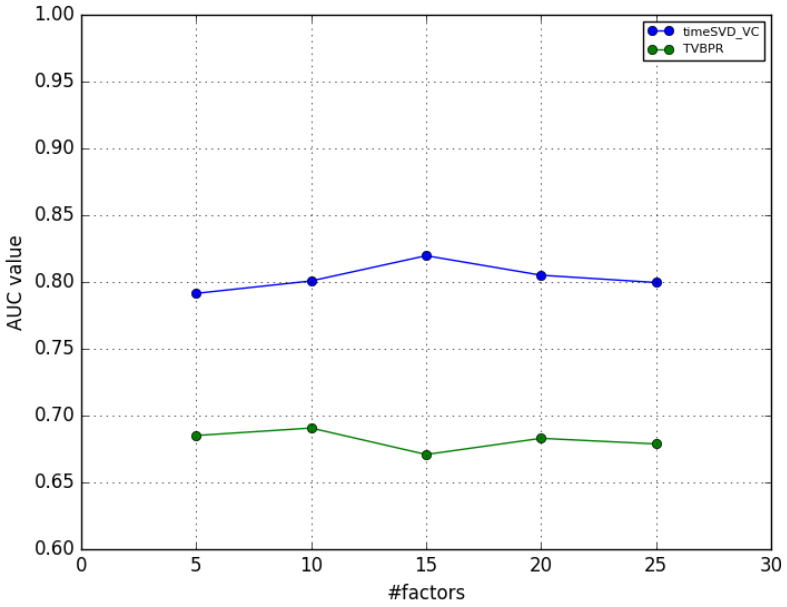



Figure 6.5: The performance of the models while the number of visual factors is changing.

As we can see from the Figure 6.5, the accuracy of both models while the number of visual factors is changing does not show any improvement, but it almost has a constant trend when the number of visual factors is changing. Here we can apply the fact that, the visual dimensions representing the visual style space are represented by the embedding matrix E (as described in section 4.2), whose values are learned from the image features of the items (or products) - this helps in the efficiency of the algorithm, and also better representing the human notion of fashion style rather than having impact on the performance of the model.

### 6.2.4 4th Task

In order to see how the models are influenced by different dataset sizes, some experiments were performed where each model is tested on different sizes of the datasets. The statistics of the datasets are shown in Table 6.1. Thus, in Figure 6.6, a graph showing the results of the models while applied in different dataset sizes is presented.

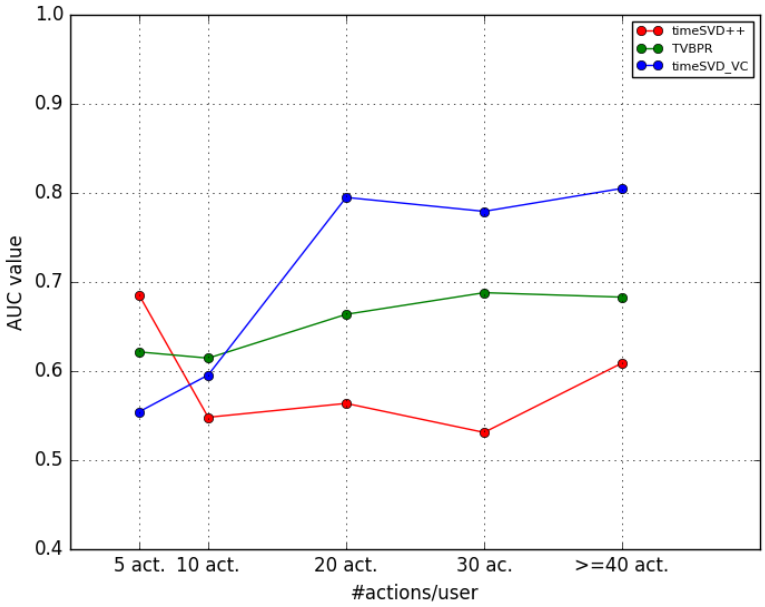



Figure 6.6: The performance of the models while the dataset size is changing.

As it can be seen from Figure 6.6, *TVBPR* has gradual improvement on the performance when the size of the dataset is increased. Also, *timeSVD*$_{VC}$'s performance is improving while the number of data is increased. This comes from the fact that, the more data we have, more accurate models we have. Here, with regard to the models involving the visual component, we can emphasize that the amount of data plays a key role for learning the values of the embedding matrix $E$.

### 6.2.5 An Exploratory Study

In this section, some insights are provided based on what models where able to capture from the dataset being used. As we see from Figure 6.1 with regard to Dataset1, the number of actions is constantly increasing after day 1500. This effect is occurring as consequence of the number of new items coming in is increasing. In order to show this, in Figure 6.7 below, a graph is presented which shows the number of items coming in per day with regard to the time span of the dataset:

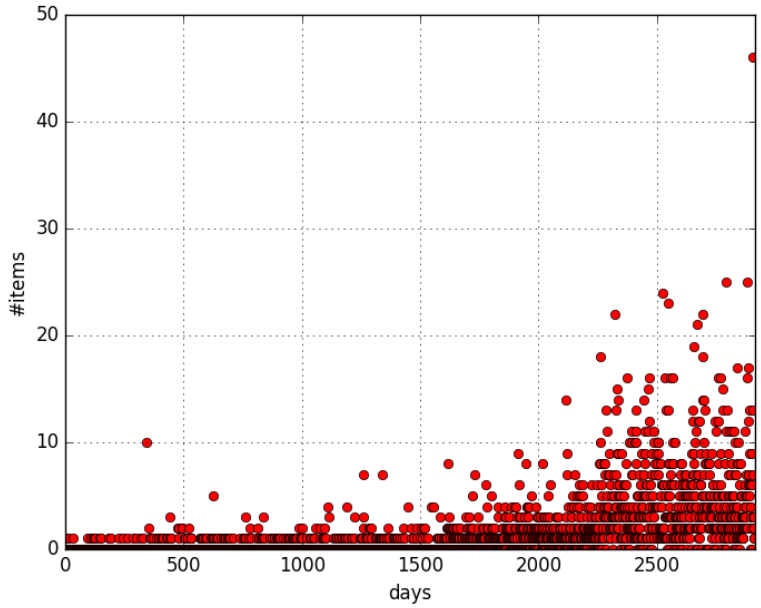



Figure 6.7: The number of items per day - calculated by considering only the first rating given to them.

Considering the effect described above, one would be interested to see how the parameters of the models express this effect. Firstly, as more items appear, we should expect a better match between users and items (considering the fact that the more data we have, the better performing model we have). Also, we want to see how the bias values are affected by this phenomenon. Considering that, for each model, we show a graph of how user-item interaction and item bias values evolve over time, as the number of new data coming in is increasing. The user-item interaction and item bias values are calculated by considering each action that has occurred on the dataset. A higher user-item interaction value indicates a better fit between user and item. Thus, in the following, with respect to each model, we show the graphs visualizing the evolution of the parameters mentioned above:

- ***timeSVD++***: Figure 6.8, shows the item bias and user-item interaction score over time.

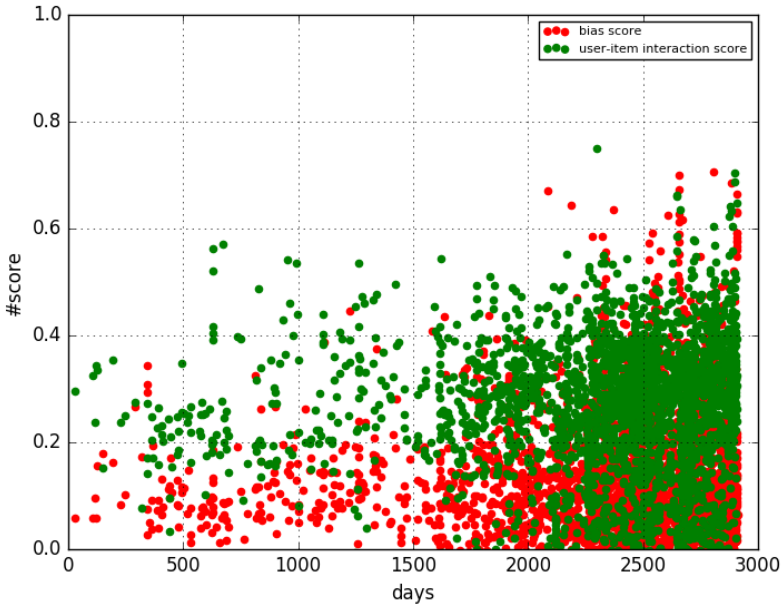



Figure 6.8: Bias score vs user-item interaction score over time when *timeSVD++* is applied.

As we can see from the Figure 6.8, as expected, the user-item interaction score is more emphasized on the area where more items are present, and at the same time showing high value.

- ***TVBPR***: In Figure 6.9 below, a graph showing the evolution of the total item bias values (i.e. including visual and non-visual bias) and the total user-item interaction score (i.e. summing up both the non-visual interaction score and the visual interaction) over time is presented.

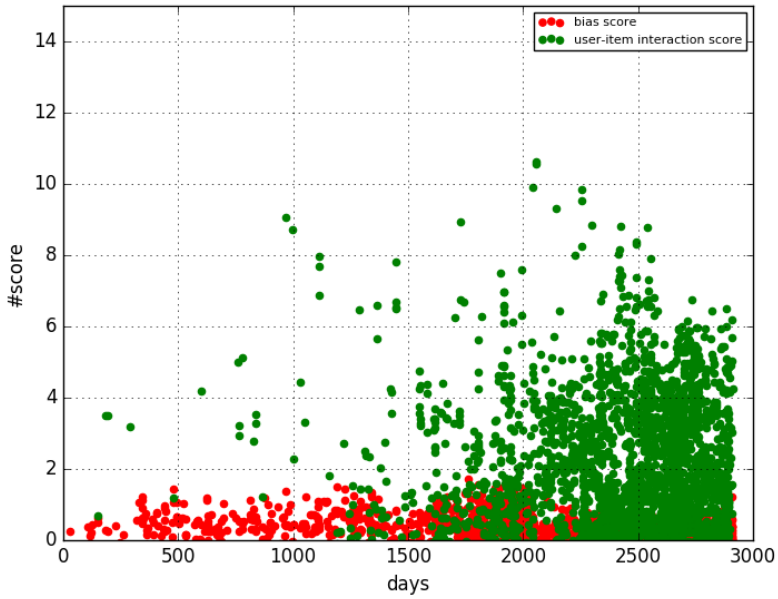



Figure 6.9: Bias score vs user-item interaction score over time when *TVBPR* is applied.

As we can see from the Figure 6.9, the user-item interaction score increases (i.e. showing a better fit between the user and item), as the number of data increases. Such that, here can also be stated that, the more data provided the better the performance is. Whereas, regarding the item bias, there is not any unexpected shift happening.

In order to see how the visual user-item interaction score is affecting the total user-item interaction score, in Figure 6.10, a graph showing the evolution of visual user-item interaction against the total user-item interaction score over time is presented.

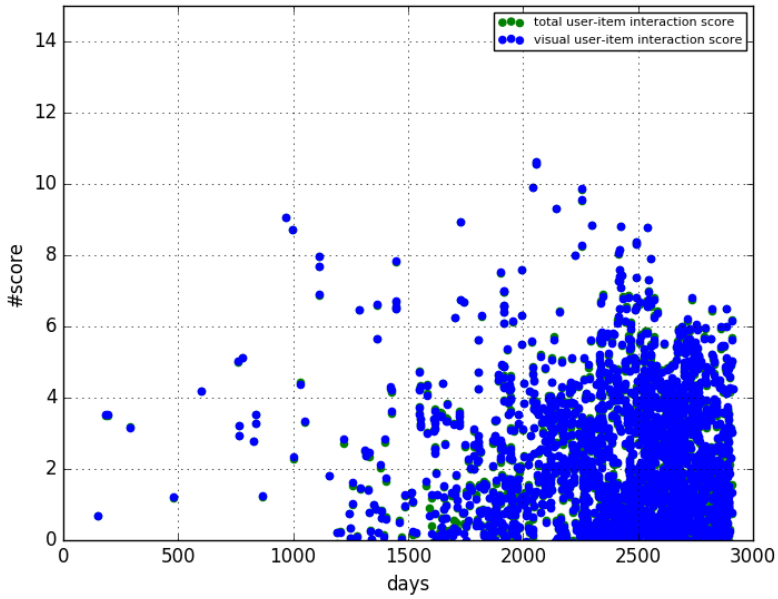



Figure 6.10: Visual user-item interaction score vs total user-item interaction score when *TVBPR* is applied.

As we can see from the Figure 6.10 above, a same trend is shown by both interactions involved. This means that, the visual user-item interaction score has a high impact on the total user-item interaction score, which is a consequence of dealing with a context of fashion data.

- ***timeSVD***$_{VC}$: As *timeSVD*$_{VC}$ involves the same components as *TVBPR*, similar graphs are provided. Such that, the graph in Figure 6.11 is showing how the total user-item interaction and the total bias values evolve over time.

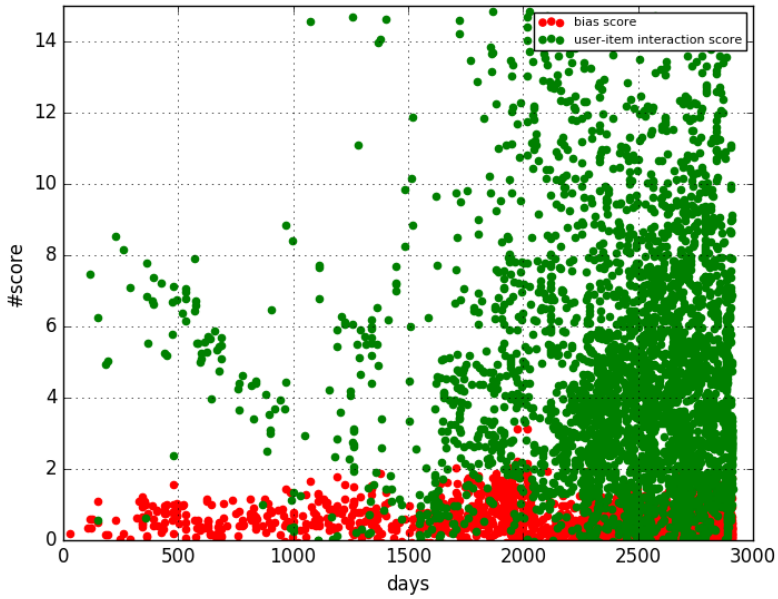



Figure 6.11: Total bias score vs total user-item interaction score when *timeSVD*$_{VC}$ is applied.

As we can see from the Figure 6.11 above, as the number of new items is increasing, the user-item interaction value is increasing as well, thus showing a better fit between users and items, i.e. better performance of the model. Whereas, the bias values shows the same trend as the one which is present on the *TVBPR* model.

Since *timeSVD*$_{VC}$ involves a time-dependent component with regard to the non-visual factors (as opposed to *TVBPR*, which involves a static non-visual component with regard to user-item interaction), one would be interested to see how these both components evolve over time. Thus, in Figure 6.12, a graph is presented showing the evolution of the components mentioned above.

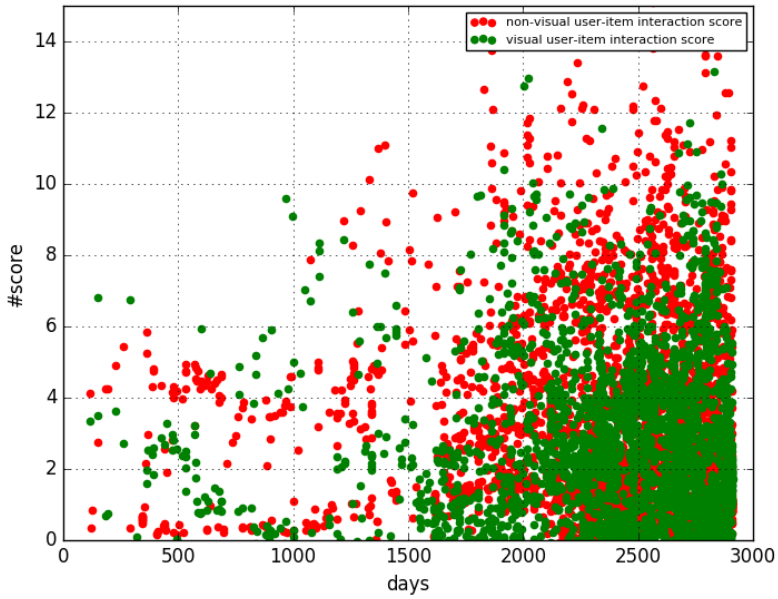



Figure 6.12: Visual user-item interaction score vs total user-item interaction score when *timeSVD*$_{VC}$ is applied.

As it can be seen from the Figure 6.12, the value of the non-visual user-item interaction is increasing as new items appear (for the reasons described in the experiments above), and the same trend is followed by the visual user-item interaction value.

We would also be interested in showing what kind of dimensions are captured by each model. For doing this, for each epoch, the top items representing each dimension are extracted. In fact, only the top items for 5 dimensions out of 20 dimensions are shown considering all the epochs (with the number of epochs being 10). To get the top item of certain dimensions, the following calculation is applied:

$$\underset{i}{\operatorname{argmax}}\, E_k f_i \odot \omega(ep) + \Delta_{E_k(ep)} f_i, \tag{6.2}$$

where $k$ is the $k$-th row from the embedding matrix $E$ representing the learned visual dimension, f representing the Deep CNN extracted features of item i, and $\omega(ep)$ and $\Delta_{E_k(ep)}$ representing the weighting vector value and the kth row of the time-dependent embedding matrix $E$ (capturing the deviation) at given epoch respectively. The item that provides the highest value with respect to the visual dimension considered is the item that best ”describes” that visual dimension. Thus, in the following , in Table 6.2 and Table 6.3, the top items of certain visual dimensions captured by *TVBPR* and *timeSVD*$_V C$ are shown respectively.

| Epoch | Dimension 1 | Dimension 2 | Dimension 3 | Dimension 4 | Dimension 5 |
|---|---|---|---|---|---|
| Epoch 10 |  |  |  |  |  |
| Epoch 9 |  |  |  |  |  |
| Epoch 8 |  |  |  |  |  |
| Epoch 7 |  |  |  |  |  |
| Epoch 6 |  |  |  |  |  |
| Epoch 5 |  |  |  |  |  |
| Epoch 4 |  |  |  |  |  |
| Epoch 3 |  |  |  |  |  |
| Epoch 2 |  |  |  |  |  |
| Epoch 1 |  |  |  |  |  |

Table 6.2: TVBPR Visualization

| Epoch | Dimension 1 | Dimension 2 | Dimension 3 | Dimension 4 | Dimension 5 |
|---|---|---|---|---|---|
| Epoch 10 |  |  |  |  |  |
| Epoch 9 |  |  |  |  |  |
| Epoch 8 |  |  |  |  |  |
| Epoch 7 |  |  |  |  |  |
| Epoch 6 |  |  |  |  |  |
| Epoch 5 |  |  |  |  |  |
| Epoch 4 |  |  |  |  |  |
| Epoch 3 |  |  |  |  |  |
| Epoch 2 |  |  |  |  |  |
| Epoch 1 |  |  |  |  |  |

Table 6.3: timeSVD_VC Visualization

In order to compare the dimensions captured by both models in the figures above, we will consider

only the items representing the dimensions at epoch 9 and 10 - the number of items appearing during that period is bigger, and thus expecting better representation. Thus, as we can see from the figures above, both of the models capture similar style. For example, if we compare the items representing the dimensions above at epoch 10, we can see that both models capture 2 watches, but not in the exact same dimensions - here we can consider the fact that adjacent items should provide similar style. The same occurs at epoch 9. As we can see, there cannot be made a clear distinction between the models, since more data are needed to learn the embedding matrices, and thus providing much clearer dimensions. But, considering the Figure 6.10 showing the trend of visual user-item interaction captured by *TVBPR*, and the Figure 6.12 showing the trend of visual user-item interaction captured by *timeSVD*$_{VC}$, we can see that both models follow the same trend, i.e. similar visual dimensions are captured.

## 6.3 Efficiency - Comparisons and Analysis of the Performance of the Models

In this section, the performance with respect to the efficiency of the algorithms is shown. In order to compare the efficiency between the models, different settings are provided. Such that, in order how the running time of the model is influenced by the size of the dataset, experiments with different dataset sizes are provided. Thus, the Table 6.4 below shows the time each model needs per iteration, while being applied in different dataset sizes.

Table 6.4: Efficiency of the algorithms while applied on different dataset sizes in seconds

| Dataset | $timeSVD++$ | $TVBPR$ | $timeSVD\_VC$ |
|---|---|---|---|
| Dataset1 | 57.7556290627s | 2181.24168897s | 2204.37025881s |
| Dataset2 | 10.9023280144s | 1181.67354703s | 1241.4301590s |
| Dataset3 | 4.83943891525 | 731.152724981s | 780.174357891s |
| Dataset4 | 1.24761009216s | 335.848999023s | 328.515271902s |
| Dataset5 | 0.324027061462 | 120.352180004s | 128.471446991s |

In order to have better overview of how the dataset size influence the running time of the models, in Figure 6.13, graph is presented showing the performance of the models over different dataset sizes.

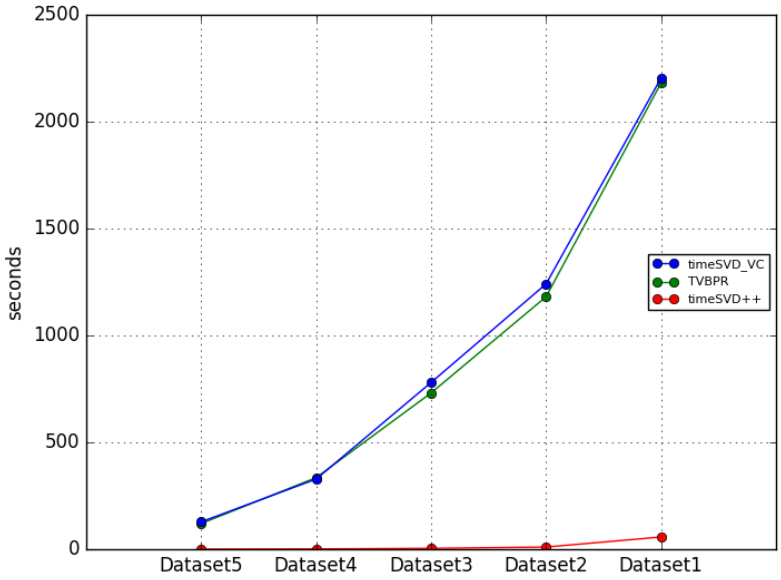



Figure 6.13: The running time of the models over different dataset sizes - the time is presented in seconds.

As we can see from the Figure 6.13, the size of the dataset has higher impact on the models that involve the visual component, $TVBPR$ and $timeSVD\_VC$ respectively. This is for the fact that more parameters are involved as well.

In the following, in order to see how certain hyper-parameters influence the performance of the models, some experiments with different values of those hyper-parameters have been performed. Such that, Table 6.5 shows the performance of the models, while the number of non-visual factors is changing. Whereas Table 6.6 shows the performance of the models while the number of visual factors is changing (this involves only the models that include the visual component). Each experiment measures the time elapsed in one iteration.

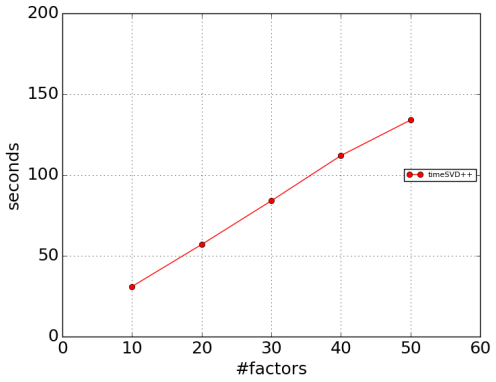
Table 6.5: Efficiency of the algorithms while the number of the non-visual factors varies

| #non-visal factors | $timeSVD++$ | $TVBPR$ | $timeSVD\_VC$ |
|---|---|---|---|
| 10 | 31.0021281242s | 2187.32572913s | 2185.51214814s |
| 20 | 57.7556290627s | 2181.24168897s | 2204.37025881s |
| 30 | 84.3528568745s | 2290.82228398s | 2183.78547597s |
| 40 | 112.017258167s | 2123.2769351s | 2265.52259803s |
| 50 | 134.62360692s | 2151.59400511s | 2218.22223496s |

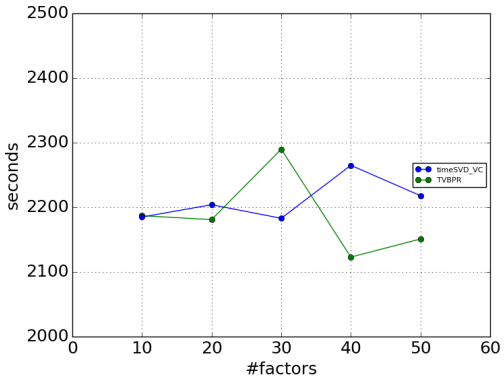
| #visual factors | $TVBPR$ | $timeSVD\_VC$ |
|---|---|---|
| 10 | 1148.070225s | 1259.39852905s |
| 20 | 2181.24168897s | 2204.37025881s |
| 30 | 3222.12467694s | 3370.24087381s |
| 40 | 4424.4775281s | 4437.88044596s |
| 50 | 5351.33963895s | 5288.38298702s |

Table 6.6: Efficiency of the algorithms while the number of the visual factors varies

In order to have a better representation of the tables above, some graphs were provided for showing the results. Such that, Figure 6.14 below shows the running time of the models while the number of non-visual factors varies. Whereas Figure 6.15 shows the running time of the models over different number of visual factors.




(a) The running time of *timeSVD++*.




(b) The running time of *TVBPR* and *timeSVD*$_V C$.

Figure 6.14: The running time of the models while the # of non-visual factors differs - the time is presented in seconds.

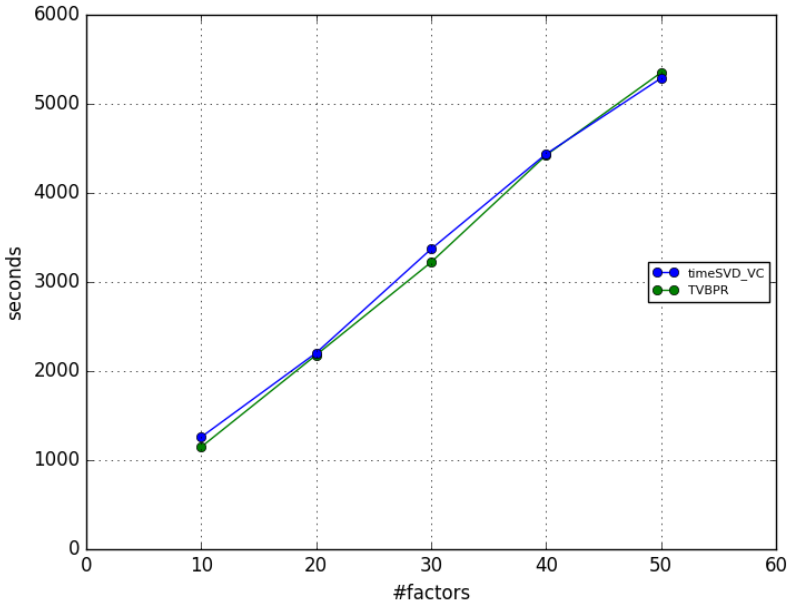



Figure 6.15: The running time of the models while the # of visual factors differs - the time is presented in seconds.

As we can see from the figures above, the running time of the models while number of non-visual factors is changing, it affects more the *timeSVD++* model as it only depends on those factors, no other factors involved. Whereas, with regard to *TVBPR* and *timeSVD*$_V C$, it does not show any significant influence while the number of non-visual factors is increased. This is as a consequence of the visual component being involved. From Figure 6.15, we can see that, while the number of visual factors increases, the running time of the models involving the visual component is also increased. Thus, *timeSVD++* requires $O(K)$ to update the parameter values, with $K$ denoting the number of non-visual factors. Whereas, as *TVBPR* and *timeSVD*$_V C$ involve the visual component additionally, both of them require $O(K + K')$ to update the parameters, where with $K'$ is denoted the number of visual factors.

# 7 Discussion

In this chapter, a discussion about the problems faced throughout this work, and possible improvements of the results from the experiments is provided.

The main part of this thesis was to extend the *timeSVD++* model with the visual component from the *TVBPR* model, and thus forming the *timeSVD*$_{VC}$ model. As *timeSVD++* and *TVBPR* model are designed for different tasks, choosing the right learning approach for the parameter values of *timeSVD*$_{VC}$, which is a combination of both, was a challenging part. As described earlier, *timeSVD++* is designed for rating prediction task, and makes use of stochastic gradient descent(SGD) to reduce the loss function. Whereas *TVBPR*, as mentioned above, is designed for item recommendation task, and applies the Bayesian Personalized Ranking (BPR) pair-wise ranking optimization framework to learn the parameter values. Respectively, the first one makes use of explicit feedback, whereas the second one makes use of implicit feedback. Thus, there are few approaches that have been tested with regard to the learning approach for *timeSVD*$_{VC}$:

- *timeSVD++*-based Approach: Intuitively, one would try to apply the same learning approach as the one used for the *timeSVD++*, since *timeSVD*$_{VC}$ is considered as an extension of *timeSVD++*. Thus, the first approach was simply by applying the stochastic gradient descent as applied on *timeSVD++*, and learning the parameter values by reducing the loss function (i.e. the different between the real rating and the predicted rating), and making use of only explicit feedback (i.e. only positive items). But, this approach was not suitable as the prediction value was a high value, such that not helping on minimizing the loss function.

- Using Implicit Feedback: The second approach tested is related to the same idea for learning the parameter values as the one used for *TVBPR*, but instead it makes use of only positive items. This method did not provide an objective function, and thus making it not applicable (even though some experiments were performed). Basically, applying this approach, it will only increase the values of the parameters being involved, and thus might help on showing the evolution of the visual dimensions uncovered from the positive items (since only these items are considered).

- Using Explicit Feedback: This approach is making use of positive items, and by using the explicit feedback(i.e. ratings), the following function is minimized:

$$\sum_{(u,i,t)\in T} -x_{u,i}(t)\log(\hat{x}_{u,i}(t)) - (1 - x_{u,i}(t))log(1 - \hat{x}_{u,i}(t)), \tag{7.1}$$

where T refers to the training set, and the values of $x_{u,i}(t)$ and $\hat{x}_{u,i}(t)$ represent the real rating and the predicted rating of user $u$ towards item $i$ at time $t$ respectively. As we were dealing with high values with regard to the predicted values (because of the visual component being involved), the values of the real rating $x_{u,i}(t)$ and the predicted values $\hat{x}_{u,i}(t)$ were bounded into (0,1) range by using the sigmoid function. But, as the predicted value is always a big value, and when the sigmoid function is applied in high values, let's say of 178, the output is 1.0. Afterwards, when log is applied to 1.0, the output is 0. Such that, this approach was not the appropriate one for applying in our context.

- *TVBPR*-based Approach: The fourth approach applies the same learning approach as the one used for *TVBPR*. Such that, the objective function is to maximize the difference between positive and negative items (as shown in Eq. 4.18. This approach showed suitable for applying it on *timeSVD*$_{VC}$ model.

With regard to finding the optimal segmentation of the dataset by applying the DP (Dynamic Programming) approach mentioned above, for our experiments, the DP did not improve the performance when applied on our datasets. Different tests have been performed while applying the dynamic segmentation on *TVBPR* which maximizes the log-likelihood function presented in Eq. 4.18. The reason for DP not providing an improvement of the performance is because of the small dataset being involved. DP makes use of the validation set to evaluate the performance of each positive item (i.e. positive item from the validation set) at given bin against 1000 negative item with respect to the positive item being considered. But, as the items from the validation set have only few ratings or not at all, it makes it hard to find the optimal segmentation for the given number of epochs.

# 8 Conclusion

The focus of this thesis has been, firstly to evaluate two existing models, then introducing a new model, and at the end evaluate them on fashion datasets. We have provided implementation of the two existing algorithms: *timeSVD++* providing the state-of-the-art with regard to rating prediction task, and *TVBPR* providing the state-of-the-art with regard to item recommendation task. Moreover, we introduced a new algorithm, which is a combination of both algorithms mentioned above. The newly introduced model, called *timeSVD*$_{VC}$, comprises the parameters of *timeSVD++*, and additionally uses the visual component presented in *TVBPR*. In order to see the performance of the models, we evaluated them on real data (Amazon data), such that providing empirical comparison with respect to the accuracy and the efficiency of the algorithms. *timeSVD*$_{VC}$ showed to outperform the other algorithms with regard to accuracy. Whereas, regarding the efficiency, *timeSVD++* is the most efficient algorithm when it comes to learning the parameter values compared to the two others, since it does not make use of any other source of information compared to the other ones. Whereas, *timeSVD*$_{VC}$ and *TVBPR* require more time to learn the parameter values since they make use of the image features extracted through Deep CNN to learn the visual-related parameters.

Since our experiments have been performed on a small dataset, a cold-start situation (i.e. all items the model are evaluated on having less than 5 ratings given to according to [4]) for all the experiments performed was provided. Considering that, a potential future work might involve the same experiments on a larger dataset. As shown on [4], $TVBPR$ model and other models considered, their performance was increasing as the size of the dataset was larger. Moreover, since different dynamics might be at play between men's actions and women's actions, also experimenting on a women dataset is suggested.

# Bibliography


[1] Toni Cebrián, Marc Planagumà, Paulo Villegas, and Xavier Amatriain. Music recommendations with temporal context awareness. *Proceedings of the fourth ACM conference on Recommender systems*, pages 349–352, 2010.

[2] Wei Di, Neel Sundaresan, Robinson Piramuthu, and Anurag Bhardwaj. Is a picture really worth a thousand words?:-on the role of images in e-commerce. *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 633–642, 2014.

[3] Anjan Goswami, Naren Chittar, and Chung H Sung. A study on the impact of product images on user clicks for online shopping. *Proceedings of the 20th international conference companion on World wide web*, pages 45–46, 2011.

[4] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517, 2016.

[5] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. *AAAI*, pages 144–150, 2016.

[6] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[7] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[8] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, 2015.

[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461, 2009.

[10] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[11] Jeffrey C Schlimmer and Richard H Granger. Beyond incremental processing: Tracking concept drift. *AAAI*, pages 502–507, 1986.

[12] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. Neuroaesthetics in fashion: Modeling the perception of fashionability. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2015.

[13] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. *Proceedings of the 7th ACM conference on Recommender systems*, pages 213–220, 2013.

[14] Yongfeng Zhang, Min Zhang, Yi Zhang, Guokun Lai, Yiqun Liu, Honghui Zhang, and Shaoping Ma. Daily-aware personalized recommendation based on feature-level time series analysis. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1373–1383, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.