fashion BRAIN project

Horizon 2020 Framework Programme

Grant Agreement: 732328 – FashionBrain

Document Information

Deliverable number: D5.3

Deliverable title: Early Demo for Trend Prediction

Deliverable description: This early demo will show how it will be possible to detect fashion trends (style) on fashion time series over time.

Due date of deliverable: 30.06.2018

Actual date of deliverable: 29.06.2018

Authors: Ines Arous, Mourad Khayati

Project partners: Zalando, MDBS

Workpackage: WP5

Workpackage leader: UNIFR

Dissemination Level: Public

Change Log

Version	Date	Status	Author (Partner)	Description/Approval Level
1	24/06/18	Initial	Zalando, MDBS	Public
2	29/06/18	Final	Zalando, MDBS	Public

Contents

Abstract

In this deliverable we propose to map the problem of trend prediction in fashion time series onto a simplified prediction problem which is the prediction of user preferences. This mapping will allow us to get a deeper understanding of the trend prediction problem and get the ground for a more general prediction solution in D5.5. Thus, this deliverable outlines a suite of algorithms for user preferences prediction. We first review the state of the art in this field. Then, we describe a new method that takes into account both the visual component of fashion items and the temporal dynamica of fashion data.

1 Introduction

Fashion time series reflect the evolution of user preferences to fashion items. These time series represent for example the sales of fashion items, the number of likes of an image on social media or the reviews in a retailer website along a period of time, etc. Compared to time series from other domains, the fashion time series exhibit two distinguishing features i) fast evolving trends and ii) visual representation of items. In fact, the increase of fashion sales or number of likes of a particular item means that users are showing an interest in this item and consider it to be fashionable. The frequently changing user preferences towards fashion items have put great challenge on the dynamic modeling of both users and items. Besides the temporal dynamics of fashion data, the visual appearance of fashion items plays a key role on user's decision-making behavior. For example, the visual representation of a fashion item is a key factor on the purchase of that item. In Figure 1, the user ratings of two fashion items are tracked over a period of time, where the ratings of T2 are missing starting from $t \geq 500$. These users give a rating that often ranges between 1 and 5. We propose to build a model able to predict the missing ratings in T2 and hence the upcoming trend for this item.

The aim of this early demo is to outline the state-of-the-art algorithms that capture changes in user preferences and predict the upcoming trends. Moreover, we extend one

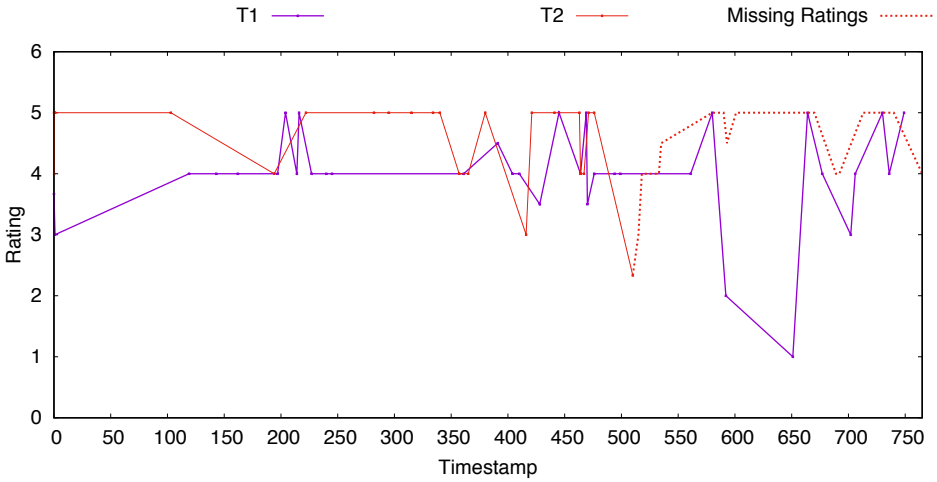


Figure 1: Rating of items over time

of the existing techniques and make it applicable to fashion time series by fulfilling the following requirements:

- The proposed technique should take into account both the visual appearance and the temporal dynamic of fashion data.
- The early demo should allow future extensions.

2 Background

2.1 Notations

Table 1 describes the notations used throughout this report.

2.2 Matrix factorization

Matrix factorization models map both users and items into a joint latent factor space such that user-item interactions are modeled as inner product in that space [**?**]. The latent space tries to explain user ratings by characterizing users and items with factors relevant to the application domain (e.g., drama vs comedies for movies, casual vs sport for clothes). Each user is associated with a vector γ_u and each item is associated with a vector γ_i. γ_i measures the extent to which item i possesses these factors while γ_u measures the extent of interest of user u in these factors. Even in cases where independent implicit feedback is absent, one can capture a the rating by accounting for which items users rate, regardless of their rating value. To this end, a second set of item factors is added, relating each item i to a factor vector $y \in \mathbb{R}^f$. We denote with α the overall average rating and with β_u and β_i the observed deviations of user u and item i, respectively, from the average. Thus a rating is predicted as follows:

Notation	Explanation
U, I	set of users, set of (fashion) items
$x_{u,i}$	the rating value of user u towards item i
$\hat{x}_{u,i}$	the predicted rating value of user u for item i
β_u, β_i	user bias, item bias
$\beta_u(t), \beta_i(t)$	user bias at time t, item bias at time t
γ_u	the latent factors of user u, $(K \times 1)$
γ_i	the latent factors of item i, $(K \times 1)$
θ_u	the visual factors of user u, $(K' \times 1)$
θ_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

Table 1: Notation

$$\hat{y}_{u,i} = \underbrace{\alpha}_{\text{rating average}} + \underbrace{\beta_u}_{\text{user bias}} + \underbrace{\beta_i}_{\text{item bias}} + \underbrace{\gamma_i^T \left(\gamma_u + \underbrace{|R(u)|^{-1/2} \sum_{j \varepsilon R(u)} y_j}_{\text{implicit feedback}} \right)}_{\text{user-item interaction}} \tag{1}$$

where the set $|R(u)|$ contains the items rated by user u.

2.3 Related methods: timeSVD++ and TVBPR+

2.3.1 timeSVD++

timeSVD++ is a time-aware model capturing the temporal dynamics of the data. timeSVD++ has been introduced in [**?**] and was applied on movie data (Netflix). It has been shown to perform well in such context. It is an extension of *SVD++* model presented in [**?**]. SVD++ is a static model that in addition to the "standard" matrix factorization, it takes into account implicit information such as user ratings. In order to extend the static model into a time-aware model, Koren [**?**] defines the parameters that are evolving with time and distinguishes between those which describe the evolution of users and others that describe the evolution of items.

The basic model introduced in (1) is static and is not able to capture the data evolution. In order to extend the static model into a time-aware model, Koren [**?**] defines the parameters that should be built as a function of time: user bias, item bias and user preferences. To capture item biases, the time range is split into time-based bins, where each bin corresponds to a predefined period of time. The function for capturing the evolution of items bias is formulated as follows:

$$\beta_i(t) = \beta_i + \beta_{i,Bin(t)} \tag{2}$$

where β_i captures the static part, whereas $\beta_{i,Bin(t)}$ captures the evolving part, where $Bin(t)$ is the index of the bin at time stamp t.

The binning approach is not convenient to be used to capture bias, since user changes are more frequent and more sudden than item changes. Thus, in order to capture user bias, there is a need for a function with a finer resolution, which would be able to capture these sudden and unexpected changes that eventually might occur. The following equation captures the possible gradual drifts on user's side:

$$\beta_u(t) = \beta_u + \alpha \cdot dev(u,t), \tag{3}$$

where α is a parameter of the model which is extracted from the input data, whereas function $dev(u,t)$ captures the deviation and a finer granularity. The deviation function is formulated as follows:

$$dev(u,t) = sign(t - t_u) \cdot |t - t_u|^{\eta}, \tag{4}$$

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 η is a value set set during the cross validation step of testing the model.

Since user preferences (expressed in (1) by γ_u), change over time, a function to capture these changes is needed. The user preference is treated as the user bias which leads to expression (5):

$$\gamma_{uk}(t) = \gamma_{uk} + \alpha_{uk} \cdot dev_u(t) + \gamma_{uk,t} \quad k = \{1, \ldots, f\} \tag{5}$$

2.3.2 TVBPR+

TVBPR+ is a one-class collaborative filtering technique for modeling the evolution of fashion trends. TVBPR+ was introduced in [**?**] and was applied on clothing data from Amazon. It has shown promising results. The evolution of fashion trends is learned by uncovering the factors that are considered to have impact on users' opinions at a given period of time. TVBPR+ adapts the 'basic' formulation of matrix factorization as well. But, considering the context where the data is really sparse (only a few items are rated), as described in [**?**], this basic formulation suffers from cold-start situations where items do not have ratings. 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, it uses the information about the visual characteristics of the item, and thus being able to uncover the preferred visual style for each user.

TVBPR+ model has two main visual components. The first one is the temporal visual factors and the second one is the temporally evolving visual bias. The temporally visual factors are represented by the following formula:

$$\theta_i(t) = \underbrace{E f_i \odot \omega(t)}_{\text{base}} + \underbrace{\Delta_E(t) f_i}_{\text{deviation}} \tag{6}$$

We use Deep Deep Convolutional Neural (Deep CNN) [**?**] to extract features from the images of items. These features are linearly embedded in a matrix. Since the visual factors evolve with time, the embedding matrix representing the visual space should be time-aware. Therefore, the embedding matrix has two components: a static E, and a deviation $\Delta_E(t)$. In (6), f_i represents the visual features of an item extracted from the Deep CNN and $\omega(t)$ represents the weighting vector which reflects how visual dimensions are weighted differently by users.

The formulations above captures 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 modelled in a similar way to *timeSVD++*, such that, a similar formulation to (3) to capture user bias is used:

$$\theta_u(t) = \theta_u + sign(t - t_u) \cdot |t - t_u|^k \eta_u \tag{7}$$

The temporally evolving visual bias captures the bias for each dimension in the visual space. As stated in [**?**], it captures "the portion of the variance which is common to all factors". The proposed formulation for the visual bias is the following:

$$\beta(t) = \beta \odot b(t) + \Delta_\beta(t) \tag{8}$$

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

In the next section, we introduce a new model that represents both user and item variation over time, while taking into account the visual dynamics of fashion items. We will explain how to learn the model parameters and how to manipulate the provided python package.

3 TimeSVD++ with Visual component

*TimeSVD*_vc is composed of a temporal component and a visual one. The temporal component is taken from timeSVD++, while the visual component is inspired by TVBPR+. In this section, we explain how the parameters of the resulting model are learned and how to use the python package.

3.1 Model formulation

Combining all the components together introduced in Section 2, we introduce a new model named $timeSVD_{vc}$ as follows:

$$\hat{x}_{u,i} = \underbrace{\hat{y}_{u,i}(t)}_{\text{TimeSVD++}} + \underbrace{\langle \theta_u(t), \theta_i(t) \rangle}_{\text{temporal visual interaction}} + \underbrace{\langle \beta(t), f_i \rangle}_{\text{temporal visual bias}} \tag{9}$$

3.2 Learning the model

To learn the model, we adopt the Bayesian Personalized Ranking framework (BPR) [**?**]. In this framework, users are recommended only the items they have not interacted with before. Also, we consider that users feedback can be positive (we refer to as positive item) and negative (we refer to as negative item) and both types of feedback have great potentials to improve the prediction. Basically, the training set used for BPR consists of triples of the following form (u, i, j), where u denotes a user, $i \in P_u$ 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 σ is the sigmoid function, $\hat{x}_{u,i}$ 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. If we take Θ the set of the model parameters, then these parameters are learned by maximizing the following regularized log-likelihood function:

$$\sum_{(u,i,j)\in D_S} \log \sigma(\hat{x}_{u,i} - \hat{x}_{u,j}) - \frac{\lambda_\Theta}{2}||\Theta||^2 \tag{10}$$

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. One component is fitting the parameter set to the training set, while the other is finding the optimal segmentation of N given epochs. The first step is to fit the parameters to the training set. To do this, the stochastic gradient ascent has been used. The second step is to fit the epoch segmentation. First, we partition the timeline into N equal size bins. Afterwards, using dynamic programming, the optimal segmentation of the timeline is calculated. The two steps above are repeated until convergence is achieved.

3.3 Code

Code is available through this link: `https://github.com/FashionBrainTeam/timesvd_vc.git`

3.3.1 Installation

```
git clone https://github.com/FashionBrainTeam/timesvd_vc.git
```

3.3.2 Description of the timesvd_vc package

The "timesvd_vc" package contains the following python files:

- run.py: file for running the experiments
- timeSVDpp.py: the implementation of timeSVD++ model
- timeSVD_vc.py: the implementation of timeSVD_vc model

The following folders with respect to the data needed to run the code are provided as well:

- datasets: the folder containing the datasets used as a training set
- image_features: Download the file image_features_Men.b from `https://drive.google.com/file/d/1_pAPalsPluuBCxPEsJxWRwjj7NHc5jxA/view` and add it to the folder image_features

3.4 Running the code

In order to run an experiment, the run.py file is used. Other files represent the models that have been implemented. The arguments needed for the run.py file are the following:

- The name of the model, which can take the following values: timeSVDpp, TVBPR, or timeSVD_vc
- # of iterations, which takes any positive integer number
- # of epochs, which takes any positive integer number

- # of non-visual factors, which takes any positive integer number
- # of visual factors, which takes any positive integer number
- The dataset being used, which takes one of the following values: 390_actions, 780_actions, 1560_actions, 2340_actions, 4099_actions, where an action refers to a user rating an item

Example: Running an experiment of timeSVD_vc with the following parameters:

- model name = timeSVD_vc
- #iterations = 100
- #epochs =10
- #non-visual factors = 20
- #visual factors = 20
- dataset =390_actions

In this case, the corresponding command line to predict missing ratings using timeSVD_vc is the following:

```
run.py timeSVD_vc 100 10 20 20 390_actions
```

We run some experiments to evaluate the accuracy of the introduced model and compare it against the two state-of-the-art techniques, *timeSVD++* and *TVBPR*. We use the Area under the ROC curve (AUC) metric[**?**] to evaluate the accuracy of each model to predict ratings. Figure 2 shows the accuracy by varying the number of time epochs. The results of this experiment show the *timeSVD*_vc outperforms the state of the art for all chosen epoch values. The results show also that all techniques models achieve the best accuracy with a number of epochs set to 5. This result is explained by the fact that the fashion dataset we used is relatively small.

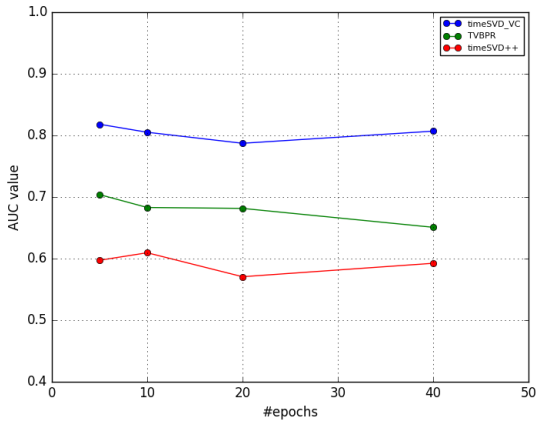
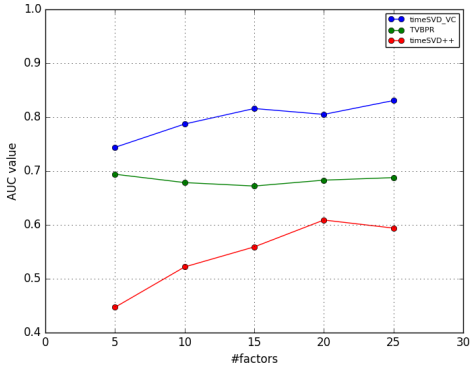
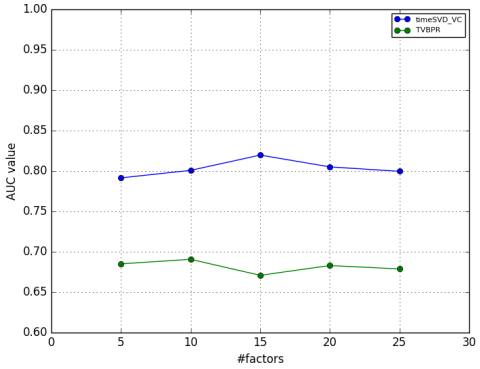


Figure 2: Accuracy of the models with varying # of epochs.

Figure 3 shows the accuracy of each model by varying the number of temporal and visual features. Figure 3(a) shows the accuracy while varying the number of temporal factors, and Figure 3(b) shows the accuracy while varying the number of visual factors. The results show that *timeSVD*_vc achieves a higher AUC with varying the number of temporal and visual factors compared to *timeSVD++* and *TVBPR* models. Our technique is able to simultaneously leverage the temporal and the visual features to produce an accurate prediction of ratings.



(a) varying # of temporal factors.



(b) Varying # of visual factors.

Figure 3: Accuracy of the models using AUC

We also evaluate the efficiency of the introduced model and compare it against the two models using datasets of different sizes as described in Table 2. The results show that our technique is as efficient as *TVBPR* while producing more accurate results. TimeSVD++ is faster than the two other techniques since it does not embed the visual component.

4 Summary

In this deliverable, we have introduced a new prediction technique of user preferences called *TimeSVD*_vc. The proposed technique extends the TimeSVD++ technique to

Dataset	$timeSVD++$	$TVBPR$	$timeSVD\_VC$
Dataset1	57.7556290627s	2181.24168897s	2204.37025881s
Dataset2	1.24761009216s	335.848999023s	328.515271902s
Dataset3	0.324027061462	120.352180004s	128.471446991s

Table 2: Efficiency of the algorithms while applied on different dataset sizes in seconds

take into account the visual part of fashion items. The accuracy of the introduced technique outperforms the state of the art while having similar efficiency. The outcome of this work will be the basis of the graphical trend prediction tool we will introduce in D5.5.