Horizon 2020

Understanding Europe's Fashion Data Universe

The Classification Algorithm and its Evaluation on Fashion Time Series

Deliverable number: D5.4

Version 2.0

Funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 732328

Project Acronym:	FashionBrain
Project Full Title:	Understanding Europe's Fashion Data Universe
Call:	H2020-ICT-2016-1
Topic:	ICT-14-2016-2017, Big Data PPP: Cross-sectorial and cross-lingual data integration and experimentation
Project URL:	https://fashionbrain-project.eu

Deliverable type	Report (R)
Dissemination level	Public (PU)
Contractual Delivery Date	30 June 2018
Resubmitted Delivery Date	27 February 2019
Number of pages	18, the last one being no. 12
Authors	Ines Arous, Mourad Khayati - UNIFR
Peer review	Alessandro Checco, Jennifer Dick - USFD

Change Log

Version	Date	Status	Partner	Remarks
0.1	01/06/2018	Draft	UNIFR	
1.0	30/06/2018	Final	UNIFR, MDBS	Rejected 15/10/2018
1.1	15/02/2019	Revised Draft	UNIFR	
2.0	27/02/2019	Resubmitted Final	UNIFR, MDBS	

Deliverable Description

As a result of task 5.3, this deliverable will consist of implemented algorithms that will be integrated within the data integration infrastructure developed within WP2 (T 2.3).

Abstract

Classification of time series is a useful tool for discovering repeated patterns in temporal data. Once these patterns have been discovered, seemingly complicated datasets can be interpreted as a temporal sequence of only a small set of clusters. Existing techniques are often non scalable and/or do not consider the correlation across them. In this deliverable, we introduce a new classification technique for fashion time series data. The proposed algorithm addresses the limitations of the existing techniques namely i) has a linear complexity and ii) is able to leverage the correlation (positive and negative) across fashion time series. The classified time series will be used as input the improve the prediction in D5.3 and D5.4. The proposed algorithm will be integrated into the MonetDB system.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

CD	Centroid Decomposition
CCD	Classification using CD
CT	Core Technology
TICC	Toeplitz Inverse Covariance-Based Clustering
WP	Work Package

1 Introduction

1.1 Motivation of the Deliverable

Classification of time series data is one of the most fundamental problems in many data analysis applications. An accurate classification of time series can be helpful to improve the accuracy of further data analytics tasks such as prediction. In this deliverable we are interested in improving the accuracy of the trend prediction introduced D5.3 and D5.5 by performing a subsequence classification of time series. The intuition is to first segment the time series into a sequence of states, each defined by a pattern, where the states can reoccur many times, and then similar sequences will be classified together. The identification of similar patterns is a challenging task especially for fashion time series where data continuously evolve over time. Take the example shown in Figure 1.1 representing six different time series from the Zalando dataset where each time series represents the number of items sold along a period of time. To be able to process different types of items, we have normalized the number of sold items using the z-score normalization technique yielding numbers in the interval [-3,3]. These time series are classified into four main classes depending on the discovered patterns. These patterns reflect a more fine-grained similarity between time series compared to the seasonal patterns. Thus, it is more beneficial to classify time series based on the subsequence similarity rather than the sold season.

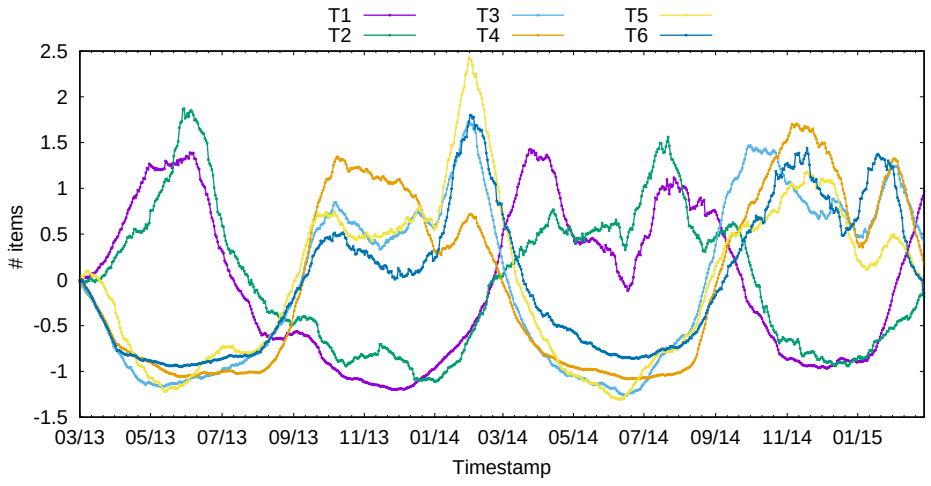


Figure 1.1: Example of fashion time series.

To address the aforementioned classification problem, we describe in this deliverable

a new classification technique that is able to leverage (positive and negative) correlation across time series and perform classification based on similar patterns.

1.2 Scope of the Deliverable

This deliverable (D5.4) is part of WP5 in which we perform fashion analysis using social media data and time series data. In this deliverable, we focus on the subsequence classification of fashion time series. D5.4 introduces a new technique to accurately classify time series based on their correlation. The outcome of this deliverable will feed into D5.3 and D5.5 where classified time series will be used for the prediction. The introduced classification technique will be integrated into T2.3 and in particular it will give support to the time series operators that will be implemented in D2.3 and D2.4. Also, D5.4 will contribute to the research challenges about Analytics for Business Intelligence (i.e., Challenge 4: Linking Entities to Product Catalogue and Challenge 7: Textual Time Trails) and to the Core Technologies about execution layer (i.e., CT2: Infrastructures for scalable cross-domain data integration and management).

This deliverable uses two sales datasets provided by Zalando and described in D2.2. The two sales datasets correspond to the sales in two different regions. Each dataset contain 3 main columns: the timestamp, the category of items and the corresponding sale of the category with respect to the timestamp. These two datasets are transformed to time series where each sale's category represent one time series. We obtain for each original dataset, 950 time series where the timestamps range from March 2013 to March 2015 with a granularity of 1 day and each time series contains up to 800 normalized observations.

2 Background

2.1 Notations

A *time series* $X = \{(t_1, v_1), \ldots, (t_n, v_n)\}$ is an ordered set of n temporal values v_i that are ordered according to their timestamps t_i. In the rest of the report, we omit the timestamps, since they are ordered, and write the time series $X_1 = \{(0, 2), (1, 0), (2, -4)\}$ as the ordered set $X_1 = \{2, 0, -4\}$. We write $\mathbf{X} = [X_1|\ldots|X_m]$ (or $\mathbf{X}_{n\times m}$) to denote an $n \times m$ matrix having m time series X_j as columns and n values for each time series as rows.

A *sign vector* $Z \in \{1, -1\}^n$ is a sequence $[z_1, \ldots, z_n]$ of n unary elements, i.e., $|z_i| = 1$ for $i = \{1, \ldots, n\}$.

We use $\times$ for scalar multiplications and $\cdot$ for matrix multiplications. The symbol $\|\|$ refers to the l-2 norm of a vector. Assuming $X = [x_1, \ldots, x_n]$, then $\|X\| = \sqrt{\sum_{i=1}^{n}(x_i)^2}$.

2.2 Centroid Decomposition

The Centroid Decomposition (CD) technique [6] decomposes an $n \times m$ matrix, $\mathbf{X} = [X_1|\ldots|X_m]$, into an $n \times m$ loading matrix, $\mathbf{L} = [L_1|\ldots|L_m]$, and an $m \times m$ relevance matrix, $\mathbf{R} = [R_1|\ldots|R_m]$, i.e.,

$$\mathbf{X} = \mathbf{L}{\cdot}\mathbf{R}^T = \sum_{i=1}^{m} L_i{\cdot}(R_i)^T$$

where $\mathbf{R}^T$ denotes the transpose of $\mathbf{R}$.

We assume that each column of $\mathbf{X}$ represents a specific time series. Algorithm 1 describes the pseudo code of CD. In each iteration i of CD, we first determine the *maximizing sign vector* Z that yields the maximal centroid value $\|\mathbf{X}^T{\cdot}Z\|$. Next, the centroid vector, C_i, and the centroid value $\|C_i\|$ are computed. Finally, vectors L_i and R_i are computed and added as columns to, respectively, $\mathbf{L}$ and $\mathbf{R}$. In order to eliminate duplicate vectors, the new loading vectors, L_{i+1}, and relevance vectors, R_{i+1}, are computed from $\mathbf{X} - L_i{\cdot}R_i^T$. The algorithm terminates when m centroid values and m loading and relevance vectors have been found.

The *truncated CD* computes a matrix $\mathbf{X}_k$ out of the CD of $\mathbf{X}$ by setting to 0 the k (non zero) last columns of $\mathbf{L}$, with $k < m$, to respectively get $\mathbf{L}_k$ and $\mathbf{X}_k = \mathbf{L}_k \cdot \mathbf{R}^T$.

Example 1 (Truncated Centroid Decomposition)**.** *To illustrate the truncated CD*

Algorithm 1: CD($\mathbf{X}$,n,m)

Input : $n \times m$ matrix $\mathbf{X}$, n,m

$i \coloneqq 1$;
repeat
  $Z_i \coloneqq FindMaxSV(\mathbf{X}, n, m)$;
  $C_i \coloneqq \mathbf{X}^T \cdot Z_i$;
  $R_i \coloneqq \frac{C_i}{\|C_i\|}$;
  $L_i \coloneqq \mathbf{X} \cdot R_i$;
  $\mathbf{X} \coloneqq \mathbf{X} - L_i \cdot R_i^T$;
  $i \coloneqq i + 1$;
until $i = m$;
return L, **R**;

decomposition, consider the input matrix $\mathbf{X}$*, that contains four time series with eight elements each.*

$$\mathbf{X} = \begin{bmatrix} 0 & 4 & 5 & 3 \\ 5 & 2 & 8 & 5 \\ 0 & 7 & 1 & 0 \\ 9 & 7 & 6 & 6 \\ 5 & 5 & 0 & 6 \\ 0 & 7 & 5 & 5 \\ 2 & 1 & 2 & 8 \\ 9 & 4 & 4 & 9 \end{bmatrix} = \underbrace{\begin{bmatrix} 6.07 & -2.93 & -1.99 & -0.77 \\ 9.65 & 1.59 & -3.23 & -3.44 \\ 4.10 & -4.85 & 3.09 & -0.04 \\ 13.68 & 1.64 & 2.17 & -2.70 \\ 8.30 & 1.63 & 2.88 & 2.44 \\ 8.83 & -4.37 & -1.10 & 0.81 \\ 7.00 & 2.39 & -2.57 & 3.40 \\ 13.01 & 4.88 & 0.74 & 0.30 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 0.42 & 0.66 & 0.47 & -0.40 \\ 0.52 & -0.66 & 0.53 & 0.08 \\ 0.44 & -0.22 & -0.61 & -0.62 \\ 0.59 & 0.27 & -0.35 & 0.67 \end{bmatrix}}_{\mathbf{R}}$$

The decomposition of the input matrix returns two matrices, i.e., the loading matrix $\mathbf{L}$ *and the relevance matrix* $\mathbf{R}$*. The decomposition has an important property that the columns of* $\mathbf{L}$ *are ordered w.r.t. their importance (quantified using the l-2 norm). The application of the truncated CD with* $k = 2$ *on* $\mathbf{L}$ *gives the following:*

$$\mathbf{L}_2 = \begin{bmatrix} 6.07 & -2.93 \\ 9.65 & 1.59 \\ 4.10 & -4.853 \\ 13.68 & 1.64 \\ 8.30 & 1.63 \\ 8.83 & -4.37 \\ 7.00 & 2.39 \\ 13.01 & 4.88 \end{bmatrix}$$

The truncated loading matrix $\mathbf{L}_2$ contains some sign properties that will be used to perform the classification of time series as explained in the next Section.

2.3 Fiedler Method

The *Fiedler* method is a clustering method that is commonly used to partition an input graph, represented as Laplacian matrix ($\mathbf{L}_A$) [2], into two clusters. It performs an eigen decomposition of the Laplacian matrix as follows:

$$\mathbf{L}_A = \mathbf{Q} \cdot \mathbf{\Sigma} \cdot \mathbf{Q}^{-1}$$

where the i-th column of the square matrix $\mathbf{Q}$ is the i-th eigen vector of $\mathbf{L}_A$ and $\mathbf{\Sigma}_{i,i}$ is its corresponding eigen value.

The eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix, called also *Fiedler* vector, can be used to partition the graph into two clusters only. More specifically, the *Fiedler* vector is used to partition the graph where rows with the same sign are placed in the same cluster [4].

An extension of this method was proposed in [1]. Instead of looking at the sign patterns of the *Fiedler* vector only, the *Extended Fiedler* method looks at the sign pattern of multiple eigenvectors of the Laplacian matrix. Similarly, we can use the sign pattern to define clusters in an input matrix [3]. Specifically, our technique applies the *Extended Fiedler* to the result of the Centroid Decomposition to perform the classification of time series. chapterTime Series Classification

2.4 CD-based Classification

2.4.1 Algorithm

The Classification using CD (CCD) algorithm applies CD to classify time series. Algorithm 2 implements the CCD technique. First, given a matrix $\mathbf{X}$ of m time series and a truncation value k, we compute the truncated CD of $\mathbf{X}$ (Line 1). As a result of this decomposition, we obtain two matrices, i.e., $\mathbf{L}$ which is an $n \times k$ matrix and $\mathbf{R}$ which is an $m \times k$ matrix. Second, the loading matrix $\mathbf{L}$ is mapped to a binary matrix $\mathbf{E}$, where, if $l_{i,j} > 0$, $e_{i,j} = 1$, otherwise $e_{i,j} = 0$. Subsequently, each row of $\mathbf{L}_i$ has a binary representation $\mathbf{E}_i$ based on the sign of its elements. The rows with the same sign pattern, and hence the same binary representation, are grouped in the same class (line 3-4).

Example 2 (Classification using Centroid Decomposition)**.** *To illustrate the classification using CD algorithm, consider the same input matrix* $\mathbf{X}$ *introduced in*

Algorithm 2: CCD($\mathbf{X}$,k)

Input : $n \times m$ matrix $\mathbf{X}$, k
Output: Classes of time series

$\mathbf{L}, \mathbf{R} = CD(\mathbf{X}, k)$;
for $i = 1$ *to* n **do**
  for $j = 1$ *to* k **do**
    $e_{i,j} = \frac{l_{i,j}}{2 \times \|l_{i,j}\|} + \frac{1}{2}$;
// $e_{i,j}$ `is an element of` $\mathbf{E}$
for $i = 1$ *to* k **do**
  $classes_i = classes_i + 2^i \times E_i$;
return classes;

Example (1). We map the matrix $\mathbf{L}_2$ *to the corresponding binary matrix* $\mathbf{E}_2$ *and get*

$$\underbrace{\begin{bmatrix} 6.07 & -2.93 \\ 9.65 & 1.59 \\ 4.10 & -4.853 \\ 13.68 & 1.64 \\ 8.30 & 1.63 \\ 8.83 & -4.37 \\ 7.00 & 2.39 \\ 13.01 & 4.88 \end{bmatrix}}_{\mathbf{L}_{k=2}} \Longrightarrow \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{E}_{k=2}}$$

In this example, we obtain two clusters: row 1, 3 and 5 (rows in gray) belong to one class and all the other rows belong to the second class.

The runtime complexity of CCD is dominated by the call of CD() which is linear w.r.t. the length of time series as the CD technique [6]. Thus, CCD inherits the same linear runtime complexity from CD and can be efficiently applied for the classification of thousands of long time series.

2.4.2 Evaluation

We first evaluate the accuracy of our classification technique. We apply CCD to the fashion time series represented in Figure (1.1). These time series represent the number of items sold in Zalando between March 2013 and March 2015. Four of these items are autumn-winter items and two of them are spring-summer items. The autumn-winter items are labelled in Zalando's dataset HW which stands for Herbst-Winter (autumn-winter in German) and the spring-summer items are labelled FS which stands for Frühling-Sommer (spring-summer in German).

Applying CCD on Zalando's time series revealed that our technique is able to accurately partition the time series w.r.t the time range. For example, by applying CCD on time series representing the items belonging to the autumn-winter season, we obtain the classification shown in Figure (2.1). In these time series, the autumn-winter items have a lower appeal during spring and summer seasons as the number of autumn-winter items sold during FS is up to 4 times lower than during HW. CCD is able to distinguish two patterns (PA1 and PA2) during the spring and summer seasons and two other patterns during autumn-winter season (PA3 and PA4). Similarly, CCD is able to partition these four patterns when applying it all time series as it is shown in Figure 2.2. These discovered patterns exhibit the same correlation properties. For instance, the first occurrence of PA1 is negatively correlated with the second appearance of PA1 as shown in Table 2.1.

Subsequence1	Subsequence2	Pearson correlation
T1 [0-20]	T1 [100-200]	-0.95
T2 [0-20]	T2 [100-200]	-0.87
T3 [0-20]	T3 [100-200]	-0.98
T4 [0-20]	T4 [100-200]	-0.21
T5 [0-20]	T5 [100-200]	-0.33
T6 [0-20]	T6 [100-200]	-0.75

Table 2.1: Correlation between time series in the first and second occurrence of PA1.

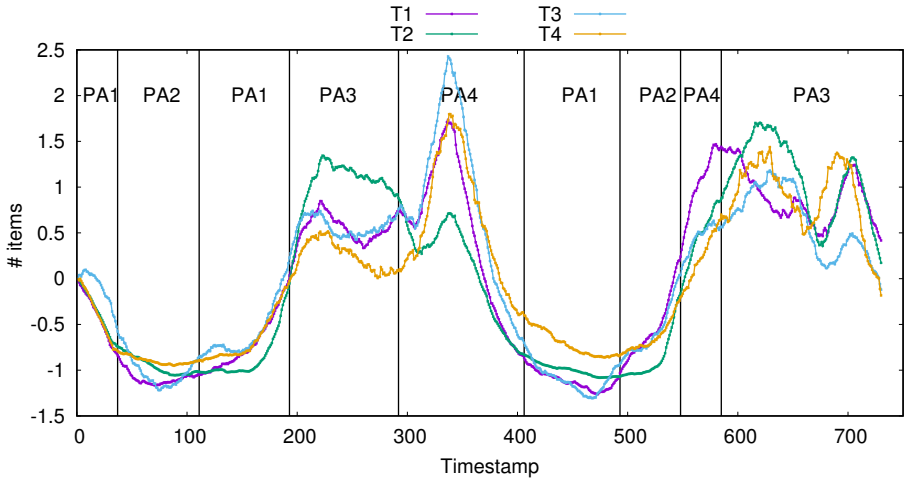


Figure 2.1: Classification of autumn-winter season fashion time series using CCD. Regular line represents the same class as the item' label and dotted line represents a different class.

Next, we evaluate the performance of the CCD technique and compare it against the state of the art technique is correlation-based classification, i.e., Toeplitz Inverse

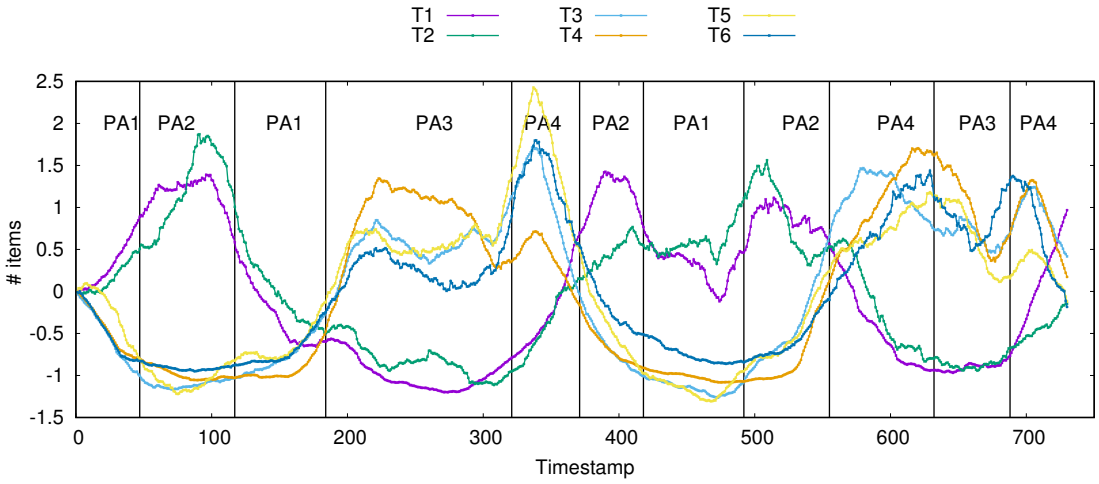
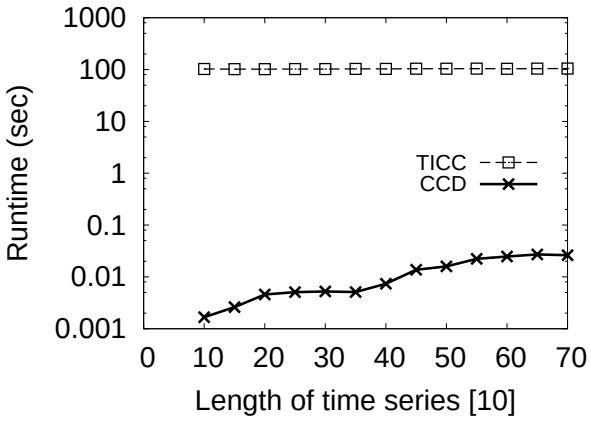


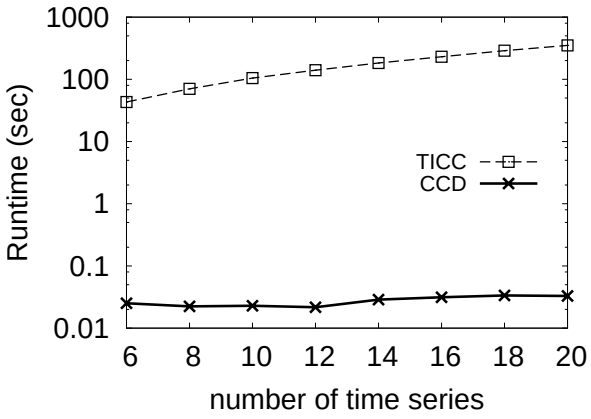
Figure 2.2: Classification of spring-summer season fashion time series using CCD.

Covariance-Based Clustering (TICC) [5]. In Figure 2.3(a) we vary the length of time series (n) from 100 to 700 and record the runtime in seconds (notice the log scale on the y-axis). In Figure 2.3(b) we vary the number of time series (m) from 6 to 20 each containing 700 values and record the runtime. The results of both experiments show that CCD is three orders of magnitude faster than TICC. For instance, to classify a 20 time series each of 700 values, CCD takes 26 milliseconds while TICC takes 1.7 minutes. The results show also that CCD shows a linear linear complexity with the length of time series and a constant complexity with the number of time series. We obtain similar results by varying the number of classes to find.

The comparison of the accuracy show that both techniques perform comparable classification that takes into account the correlation across the input time series.



(a) Varying length of TS (n).



(b) Varying # of TS (m).

Figure 2.3: Classification Efficiency.

2.5 Running the code

2.5.1 Code

Code is available **here**

2.5.2 Installation

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

2.5.3 Description of the CCD package

The "CCD" package contains the following files:

- cd_cluster.py: The implementation of the CCD algorithm

- Input folder: This folder contains the running example input matrix in the file "example.txt"
- Result folder: This folder contains the results of CCD on the running example

In order to run an experiment, the cd_cluster.py file is used. The arguments needed for the cd_cluster.py file are the following:

- The path for the input file
- The path for the output file
- # of rows n, which takes any integer number
- # of columns m, which takes any integer number
- # of truncated columns k, which takes any integer number and needs to be less than m

Example: To run an experiment of CCD with the following parameters:

- input file=./Input/example.txt
- output file= ./Result/classes.txt
- #rows= 8
- #cols = 4
- #truncated cols = 2

the corresponding command line would be the following:

```
python cd_cluster.py ./Input/example.txt ./Result/classes.txt 8 4 2
```

3 Conclusions

In this deliverable, we have implemented the CCD algorithm, a CD based classification method, that accurately partitions fashion time series. CCD leverages the correlation across fashion time series to identify items' classes. The results of the application of CCD on Zalando's sale dataset show that our technique i) is able to accurately partition the sales time series w.r.t the time range and ii) scalable with the number of time series, i.e., linear runtime complexity. This classification will help to improve the task of predicting fashion trends described in T5.4.

Bibliography

[1] Charles J Alpert and So-Zen Yao. Spectral partitioning: the more eigenvectors, the better. In *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, pages 195–200. ACM, 1995.

[2] Ravindra B. Bapat. *Graphs and Matrices*. Universitext. Springer-Verlag London, second edition, 2010. doi: 10.1007/978-1-4471-6569-9.

[3] Emmeline P Douglas. Clustering datasets with singular value decomposition. Master's thesis, College of Charleston, United States – South Carolina, 2008.

[4] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4): 619–633, 1975. URL http://dml.cz/dmlcz/101357.

[5] David Hallac, Sagar Vare, Stephen P. Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 5254–5258, 2018. doi: 10.24963/ijcai.2018/732. URL https://doi.org/10.24963/ijcai.2018/732.

[6] Mourad Khayati, Michael H. Böhlen, and Johann Gamper. Memory-efficient centroid decomposition for long time series. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 100–111, 2014. doi: 10.1109/ICDE.2014.6816643. URL https://doi.org/10.1109/ICDE.2014.6816643.