Horizon 2020 Framework Programme

Grant Agreement: 732328 – FashionBrain

Document Information

Deliverable number: D5.4

Deliverable title: The classification algorithm and its evaluation on fashion time series

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).

Due date of deliverable: 30.06.2018

Actual date of deliverable: 29.06.2018

Authors: Ines Arous, Mourad Khayati

Project partners: MDBS

Workpackage: WP5

Workpackage leader: UNIFR

Dissemination Level: Public

Change Log

Version	Date	Status	Author (Partner)	Description/Approval Level
1	20/06/18	Initial	MDBS	Public
2	29/06/18	Final	MDBS	Public

Contents

Abstract

This deliverable describes an algorithm and sample code for a time-series classification package to be integrated into MonetDB. We review the Centroid Decomposition (CD) technique and outline a CD-based classification technique that leverages the correlation across fashion time series.

1 Introduction

Classification of time series data is one of the most fundamental problems in many data analysis applications. The application of classification on fashion time series is a challenging task since these fashion time series evolve over time. The main intuition is to group fashion data on the basis of their similarity, where data evolves over time. Discovering the patterns hidden in fashion time series is substantial and essential for understanding and further utilizing these data. Take the example of a time series that represents the number of items sold in a retailer website along a period of time. The items are originally classified per season: the autumn-winter season and the spring summer season. However, a big number of these items are sold either on a different season than the original one or all the year long. Thus, it is pivotal for the data analysis, to classify fashion time series w.r.t the sold seasons, in addition to the original classification. Moreover, as fashion time series are correlated, the produced classification should leverage this correlation. To sum up, the proposed classification technique should fulfill the following requirements:

- scalable solution that is linear with the number of time series
- takes into account the correlation across time series

Figure 1 shows an example of fashion time series. In this figure, six different time series from the Zalando dataset are represented. The six time series represent the number of different items sold in different seasons. These time series are classified into two main classes: the spring-summer season class and the autumn-winter season class.

The result of the classification of fashion time series will be used to improve the accuracy of the trend prediction task in D5.3 and D5.5.

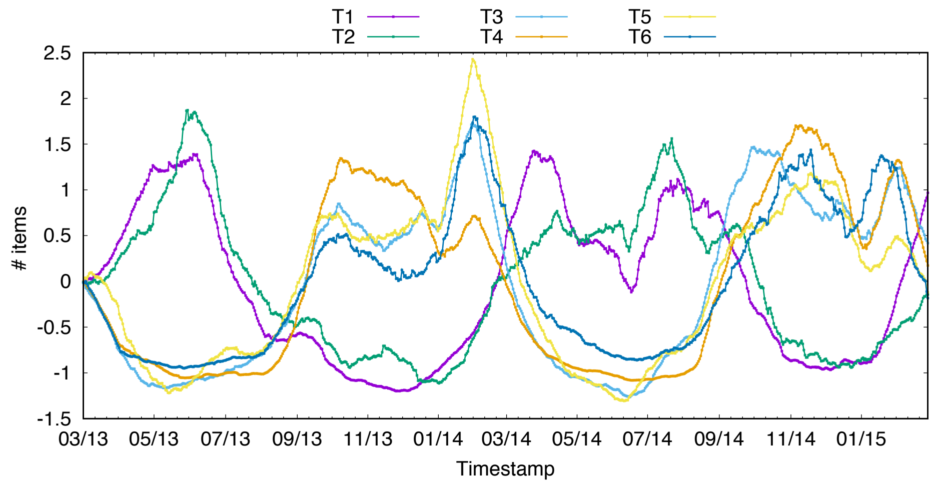

Figure 1: Example of fashion time series.

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 $X_1 = \{(0,2),(1,0),(2,\text{-}4)\}$ as the ordered set $X_1 = \{2,0,\text{-}4\}$. 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* $Z \in \{1,\text{-}1\}^n$ 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)* [?] 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 the Centroid Decomposition. 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 $\mathbf{X}_k = \mathbf{L}_k \cdot \mathbf{R}^T$.

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

```
   Input  : $n \times m$ matrix $\mathbf{X}$, $n$,$m$
1  $i := 1$ ;
2  repeat
3      $Z_i := FindMaxSV(\mathbf{X}, n, m)$ ;
4      $C_i := \mathbf{X}^T \cdot Z_i$;
5      $R_i := \frac{C_i}{\|C_i\|}$;
6      $L_i := \mathbf{X} \cdot R_i$ ;
7      $\mathbf{X} := \mathbf{X} - L_i \cdot R_i^T$ ;
8      $i := i + 1$;
9  until $i = m$;
10 return L, R;
```

Example 1 (Truncated Centroid Decomposition) *To illustrate the truncated CD 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 $\mathbf{L}_A$ [?], 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 [?].

An extension of this method was proposed in [?]. 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 [?]. Specifically, our technique applies the *Extended Fiedler* to the result of the Centroid Decomposition to perform the classification of time series.

3 Time Series Classification

3.1 CD-based Classification

3.1.1 Algorithm

The CCD algorithm uses the CD decomposition 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).

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

```
Input  : $n \times m$ matrix $\mathbf{X}$, $k$
Output: Classes of time series
1 $\mathbf{L}, \mathbf{R} = CD(\mathbf{X}, k)$ ;
2 for $i = 1$ to $n$ do
3     for $j = 1$ to $k$ do
4         $e_{i,j} = \frac{l_{i,j}}{2 \times \|l_{i,j}\|} + \frac{1}{2}$ ;
  // $e_{i,j}$ is an element of $\mathbf{E}$
5 for $i = 1$ to $k$ do
6     $classes_i = classes_i + 2^i \times E_i$ ;
7 return classes;
```

Example 2 (Classification using Centroid Decomposition) *To illustrate the classification using CD algorithm, consider the same input matrix* $\mathbf{X}$ *introduced in 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 number of time series as the CD technique [**?**]. Thus, CCD inherits the same linear runtime complexity from CD and can be efficiently applied for the classification of thousands of time series.

3.1.2 Examples

We apply CCD to the fashion time series represented in Figure (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). We use the same labels for classes where we will use HW for the autumn-winter season and FS for the spring-summer season.

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). 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 accurately partition these two classes where the HW class range between September and February which indeed corresponds to the autumn-winter period. Similarly, CCD is able to partition these two seasons when applying it on spring-summer item's time series as it is shown in Figure (3).

3.2 Running the code

3.3 Code

Code is available **here**

3.3.1 Installation

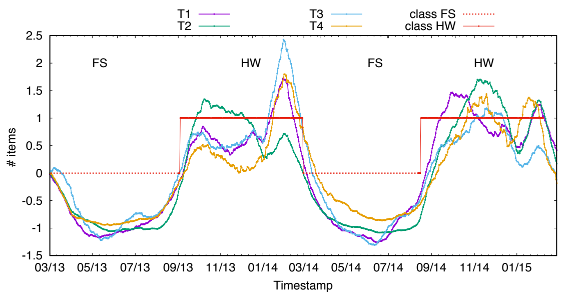

Figure 2: Classification of autumn-winter season fashion time series using CCD. regular line represents the same class as the item' label and dotted line represents a different class.

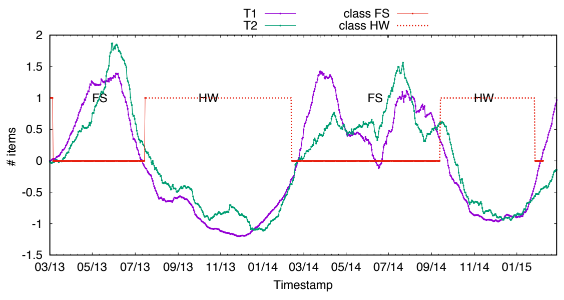

Figure 3: Classification of spring-summer season fashion time series using CCD.

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

3.3.2 Description of the CCD package

The ”CCD” package contains the following files:

- cd_cluster.py: The implementation of 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 the 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
```

3.4 Summary

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 Zalando to improve the task of predicting fashion trends in T5.4.