University of Zurich^UZH

Department of Informatics

Recovery of Missing Values using Matrix Decomposition Techniques

A dissertation submitted to the Faculty of Economics, Business Administration and Information Technology of the University of Zurich

for the degree of
Doctor of Science (Ph.D.)

by

Mourad Khayati

from Tunisia

Accepted on the recommendation of

Prof. Dr. Michael H. Böhlen
Prof. Dr. Renato Pajarola

Abstract

Time series data is prominent in many real world applications, e.g., hydrology or finance stock market. In many of these applications, time series data is missing in blocks, i.e., multiple consecutive values are missing. For example, in the hydrology field around 20% of the data is missing in blocks. However, many time series analysis tasks, such as prediction, require the existence of complete data.

The recovery of blocks of missing values in time series is challenging if the missing block is a peak or a valley. The problem is more challenging in real world time series because of the irregularity in the data. The state-of-the-art recovery techniques are suitable either for the recovery of single missing values or for the recovery of blocks of missing values in regular time series. The goal of this thesis is to propose an accurate recovery of blocks of missing values in irregular time series.

The recovery solution we propose is based on matrix decomposition techniques. The main idea of the recovery is to represent correlated time series as columns of an input matrix where missing values have been initialized and iteratively apply matrix decomposition technique to refine the initialized missing values. A key property of our recovery solution is that it learns the shape, the width and the amplitude of the missing blocks from the history of the time series that contains the missing blocks and the history of its correlated time series. Our experiments on real world hydrological time series show that our approach outperforms the state-of-the-art recovery techniques for the recovery of missing blocks in irregular time series. The recovery solution is implemented as a graphical tool that displays, browses and accurately recovers missing blocks in irregular time series.

The proposed approach supports learning from highly and lowly correlated time series. This is important since lowly correlated time series, e.g., shifted time series, that exhibit shape and/or trend similarities are beneficial for the recovery process. We reduce the space complexity of the proposed solution from quadratic to linear. This allows to use time series with long histories without prior segmentation. We prove the scalability and the correctness of the solution.

Zusammenfassung

Zeitreihendaten sind in vielen praktischen Anwendungsgebieten, wie z.B. in der Hydrologie oder an der Börse, zu finden. Ein häufiges Problem in solchen Anwendungen ist das Fehlen von Daten in Blöcken, d.h., dass mehrere Werte hintereinander nicht vorhanden sind. In der Hydrologie beispielsweise fehlen ca. 20 % aller Daten in Blöcken. Dies ist problematisch, da bei den meisten Analysen von Zeitreihen, wie beispielsweise Vorhersagen, vollständigen Datensätze benötigt werden.

In Zeitreihen ist die Wiederherstellung von Blöcken fehlender Werte besonders schwierig, wenn es sich bei dem fehlenden Block um einen Berg oder ein Tal handelt. Die Wiederherstellung von praktischen Zeitreihen wird zusätzlich durch Irregularitäten innerhalb der Daten erschwert. Aktuelle Techniken können entweder nur zur Wiederherstellung von einzelnen Werten oder zur Wiederherstellung von Blöcken in regulären Zeitreihen verwendet werden. Das Ziel dieser Arbeit ist die akkurate Wiederherstellung von Blöcken fehlender Werte aus irregulären Zeitreihen.

Unsere Lösung zur Wiederherstellung fehlender Blöcke basiert auf Matrix-Zerlegungstechniken. Die Hauptidee umfasst die Repräsentation korrelierter Zeitreihen als Spalten einer Eingabe-Matrix, in der die fehlenden Werte initialisiert werden. Durch iterative Anwendung der Matrix-Zerlegungstechnik werden die fehlenden Werte angenähert. Ein Hauptmerkmal unserer Lösung ist, dass sie die Form, Breite und Amplitude der fehlenden Blöcke aus der Historie, die diese Blöcke enthält, aber auch aus den Historien korrelierter Zeitreihen lernt. Unsere Experimente auf realen hydrologischen Zeitreihen zeigen, dass unser Ansatz die aktuellen Techniken bei der Wiederherstellung von fehlenden Blöcken in irregulären Datensätzen übertrifft. Unsere Lösung wurde als ein Werkzeug mit grafischer Benutzeroberfläche umgesetzt, das erlaubt, fehlende Blöcke in irregulären Zeitreihen anzuzeigen, darüber zu navigieren und diese wiederherzustellen.

Der eingeführte Ansatz unterstützt das Lernen aus stark und schwach korrelierten Zeitreihen. Dies ist wichtig, da schwach korrelierte Zeitreihen, wie z.B. verschobene Zeitreihen, welche Ähnlichkeiten in der Form und/oder dem Trend aufweisen, sich vorteilhaft auf den Wiederherstellungsprozess auswirken. Weiterhin haben wir die Speicherkomplexität des präsentierten Ansatzes von einer vormals quadratischen auf eine lineare Lösung reduziert. Dies erlaubt die Nutzung von grossen Zeitreihen mit langer Historie ohne vorherige Segmentierung. Abschliessend haben wir die Skalierbarkeit und Korrektheit der Lösung bewiesen.

We cannot think of being acceptable to others until we have first proven acceptable to ourselves.

Malcolm X

Acknowledgments

First and foremost I wish to thank my advisor Prof. Michael Böhlen for providing me with the opportunity to complete my PhD thesis in his research group. As my supervisor, he has constantly helped me to remain focused on achieving my goal. Michael has always been available for discussions and provided insightful comments and suggestions about my research, papers and presentations. This work would not have been possible without his guidance, support and encouragement.

Many thanks to Prof. Renato Pajarola for accepting to co-advise my PhD thesis and to Prof. Elaine Huang for serving as my committee chair even at hardship.

I would like to express my gratitude to my co-authors Prof. Johann Gamper and Prof. Philippe Cudré-Mauroux for their critical comments and valuable feedback about our joint papers.

I thank my colleagues from the Database Technology Group (DBTG) at the University of Zurich for their useful comments during our group meetings. A special thank to Anton Dignös for being a supportive and very helpful office mate.

Last but not the least, I would like to thank my family: my parents, my brothers and sister. And finally a heartfelt thank to my dear wife, who has always been by my side throughout this PhD.

Mourad Khayati

Contents

List of Figures **xv**

List of Tables **xix**

1 Introduction **1**

1.1 Recovery of Missing Values 1

1.2 Contributions 3

1.2.1 REBOM Recovery of Missing Values 4

1.2.2 Recovery using Mixed Correlated Time Series 8

1.2.3 Scalable Centroid Decomposition 9

1.3 Organization of the Thesis 11

2 REBOM **13**

2.1 Introduction 14

2.2 Related Work 16

2.3 Preliminaries and Background 18

2.3.1 Notation 18

2.3.2 Preprocessing of Time Series 19

2.3.3 Low Rank Matrix Decomposition 20

2.4 REBOM 22

2.4.1 Correlation Ranking Matrix 22

2.4.2 Stepwise Correlation Monotonicity 24

2.4.3 Iterative Recovery of REBOM 25

2.4.4 Algorithm 28

2.5 Experiments 30

2.5.1 Experimental Setup 30

2.5.2 Experiments with Hydrological Time Series 31

2.5.3 Experiments with Synthetic Regular Time Series 38

2.6 Conclusion 40

3 Using Lowly Correlated Time Series to Recover Missing Values 41

3.1 Introduction 42

3.2 Related Work 43

3.3 Preliminaries 45

3.3.1 Notation 45

3.3.2 Background 45

3.3.3 Matrix Decomposition 46

3.4 Decomposition Comparison 48

3.4.1 Recovery Process 48

3.4.2 SVD recovery 49

3.4.3 CD recovery 51

3.4.4 Complexity 52

3.5 Experiments 53

3.5.1 Recovery using real world TS 53

3.5.2 Recovery using synthetic TS 56

3.5.3 Comparison to SGD based recovery 59

3.5.4 Approximation accuracy 60

3.5.5 Number of iterations of CD 60

3.6 Conclusion 60

4 Memory-efficient Centroid Decomposition 63

4.1 Introduction 64

4.2 Related Work 65

4.3 Preliminaries 68

4.3.1 Notations 68

4.3.2 Centroid Decomposition 69

4.3.3 Application of CD 70

4.4 Scalable Sign Vector 71

4.4.1 Overview and Data Structures 71

4.4.2 Transformation of the Maximization Problem 72

4.4.3 Computation of Maximizing Sign Vectors 78

4.5 Properties of the Algorithm 80

4.5.1 Monotonicity 80

4.5.2 Termination 82
4.5.3 Greedy Strategy 83
4.5.4 Correctness 85
4.5.5 Complexity Analysis 86
4.6 Empirical Evaluation 87
4.6.1 Setup 87
4.6.2 Experiments 87
4.7 Conclusion and Future Work 93

5 Conclusion and Future Work **95**

Bibliography **97**

List of Figures

1.1 Example of Irregular Time Series 2
1.2 Example of Three Hydrological Time Series. TS1 contains a missing block for $ts \in [115, 170]$. 5
1.3 Recovery of the Missing Block using REBOM. 5
1.4 Display of Time Series using REBOM. 6
1.5 Recovery of a Missing Block using REBOM. 7
1.6 Illustration of the State-of-the-art Solution. 9
1.7 Illustration of our Solution. 10

2.1 Recovery Performed by REBOM 15
2.2 Recovery using Different Techniques 17
2.3 Original Time Series X_1^0, X_2^0; Initialized Time Series X_1^1, X_2^1; Multidimensional Points M. 20
2.4 Example of Singular Value Decomposition 22

2.5 Illustration of Dimensionality Reduction 22
2.6 Example of $\mathrm{M}^{top\text{-}3}$ 24
2.7 Example of a matrix and its $SVD_r()$ transformation 25
2.8 Iterative Recovery of REBOM 28
2.9 Recovery using Low Rank SVD and DynaMMo 31
2.10 Recovery Using REBOM and DynaMMo 32
2.11 Recovery of Blocks of Different Lengths 33
2.12 Impact of n in the Recovery MSE of REBOM 35
2.13 Recovery MSE using Different Number of Time Series in the Top-k Ranking Matrix 36
2.14 Number of Iterations using Different Initialization Techniques 37
2.15 Impact of Extremely Irregular Time Series in the Recovery of REBOM 37
2.16 Recovery of DynaMMO and REBOM for Time Series of Different Amplitudes 38
2.17 Recovery of DynaMMO and REBOM for Shifted Time Series 39

3.1 Example of Singular Value Decomposition. 47
3.2 Example of Centroid Decomposition. 47
3.3 MSE relative reduction of CD and SVD using highly and lowly correlated time series: case 1. 54
3.4 MSE relative reduction of CD and SVD using highly and lowly correlated time series: case 2. 54
3.5 Recovery using highly and lowly correlated hydrological TS. 56
3.6 MSE for Successive Removed Blocks. 57
3.7 Impact of varying time shift 57
3.8 Recovery using highly and lowly correlated synthetic TS. 58

3.9 Recovery accuracy using varying number of input TS. 59

3.10 Recovery accuracy of CD against GROUSE. 59

3.11 Approximation error. 60

3.12 number of iterations performed by CD. 61

4.1 Illustration of the main data structures used by the QSV algorithm. 66

4.2 Example of Centroid Decomposition. 71

4.3 Illustration of SSV. 72

4.4 Illustration of Centroid Decomposition using SSV. 72

4.5 Runtime by Varying n. 88

4.6 Runtime by Varying m in Hydrological Time Series. 89

4.7 Memory Usage. 89

4.8 Monotonicity Property of SSV. 90

4.9 Correctness of SSV. 90

4.10 Number of Iterations in SSV. 91

4.11 Impact of the Sign of Input Rows. 92

4.12 Run Time with Varying Percentage of Negative Rows. 92

4.13 Impact of the Length of the Time Series and the Correlation Between Them on the Recovery of Missing Values. 93

List of Tables

2.1 Average Running Time Comparison (sec). 36

4.1 Description of the First Set of Time Series. 88

4.2 Description of the Second Set of Time Series. 91

CHAPTER 1

Introduction

1.1 Recovery of Missing Values

Nowadays, time series are ubiquitous and are particularly important in real world applications that manage and analyze the history of data, e.g., environment, finance, web traffic, medicine, astronomy, etc.

In the last decade, commercial systems that handle time series data have been proposed. Examples are time series databases such as Prometheus [PRO12], InfluxDB [INF13], OpenTSDB [TSD10], etc., and time series management tools such as SAP Time Series Management [SAP10], IBM Informix TimeSeries [IBM13], etc. These systems are designed to efficiently handle time series data. They offer many data management operations such as storage, display, compression and prediction. The support of the recovery of missing values is however limited, i.e., missing values are either ignored or replaced by standard statistical methods such as mean of values, linear interpolation, regression, etc.. The accuracy of these standard statistical recovery techniques deteriorates in the case where the missing values in the time series are blocks that include peaks and/or valleys.

The goal of this thesis is to propose: (i) an accurate recovery technique of blocks of missing values in time series; (ii) a recovery technique that is able to learn from similar but lowly correlated time series; and (iii) a scalable recovery technique that does not require any prior segmentation.

Recovery techniques that use only the time series that contains the missing blocks are suitable for regular time series, i.e., time series having peaks and valleys that follow one or more periodic models such as a cosine wave. However, in real world applications, e.g., the hydrological field, time series are irregular and using only the time series that contains the missing blocks yields bad block recovery results.

Example 1. Figure 1.1 graphically illustrates a humidity time series measured during the year 2001 for a period of three consecutive days in the area of Lisser located in the region of South Tyrol in Italy. The values of the observations are recorded every 15 min. The x-axis represents the timestamps of the observations and the y-axis represents the normalized values of each observation recorded. The humidity time series is irregular, i.e., the shape, the amplitude and the width of the peaks and valleys do not follow any periodic model.

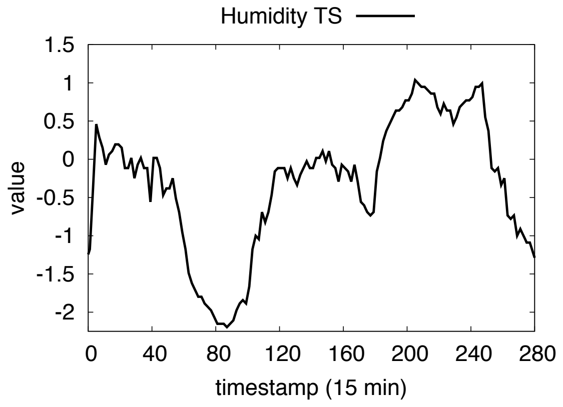


Figure 1.1: Example of Irregular Time Series

Thus, instead of using only the time series that contains the missing blocks, we propose a recovery technique that learns from different time series. This learning will make it possible to accurately recover blocks of missing values in irregular time series.

The solutions introduced in this thesis are based on the following observations:

Observation 1 An accurate recovery of blocks of missing values requires the use of the time series that contains the missing blocks and other similar time series. The correlation is an effective measure to evaluate the similarity between time series.

Observation 2 The Centroid Decomposition (CD) technique takes correlated input time series to produce correlated output vectors that can be used to perform an accurate recovery using time series with mixed correlations.

Observation 3 The space complexity of the CD technique can be reduced by avoiding the construction of the correlation matrix.

Observation 1 yields a recovery solution based on a matrix decomposition technique. The latter takes the top-k most correlated time series to the time series that contains the missing blocks as columns of an input matrix. The columns of the matrix are used to iteratively refine the initialized missing values. If the set of the most correlated time series to the time series that contains the missing blocks changes after updating the missing values, then we choose a different time series in the recovery process.

Observation 2 yields a CD based recovery technique that learns from highly and lowly correlated time series at a time. In fact, the CD technique performs a matrix decomposition that generates correlated output vectors. The impact of this property on the block recovery is empirically evaluated.

Observation 3 yields a space complexity reduction of the CD technique. We first derive a new optimization problem from the original one. Then, we solve the new optimization problem using a sequence of weight vectors instead of the correlation matrix. We iteratively choose the weight vector that yields the local optimum. At the end of this process, the weight vector that solves the optimization problem is computed.

1.2 Contributions

In this thesis we make three main contributions to the field of time series:

- We introduce a parameter-free algorithm that uses the correlation between time series in order to perform an accurate recovery of blocks of missing values in irregular time series.
- We introduce a CD based recovery technique that learns from highly and lowly correlated time series to recover blocks of missing values.
- We reduce the space complexity of the CD technique to linear to make it usable without segmenting the time series.

The research solutions proposed in this thesis are triggered by real world problems. The research methodology is based on a formal description of the problem and its properties, an implementation of the solution and an empirical comparison against the state-of-the-art solutions using real world data sets. Prototypes of the proposed algorithms have been implemented and source code is made available[1].

The remainder of this section describes in details the main contributions of this thesis.

1.2.1 REBOM Recovery of Missing Values

In the first part of this thesis we propose to recover blocks of missing values in irregular time series. The recovery process is as follows. First, the missing values are initialized. Then, we apply our technique to iteratively refine the initialized missing values. The desired recovery should use the correlation between time series to perform a block recovery that satisfies the following properties: (1) *accurate reconstruction* of the amplitude, shape and type of the missing blocks; (2) *parameter-free* recovery; (3) *independent recovery* from the initialization method.

In order to achieve this goal, we present a Singular Value Decomposition (SVD) based greedy approach, called REBOM, that uses two ranking vectors to learn from correlated time series to recover missing blocks in time series.

Example 2. In Figure. 1.2 we consider the humidity time series displayed in Figure 1.1 denoted as TS1 to which we add two other humidity time series measured in two different areas of the region of South Tyrol during the same time period denoted respectively as TS2 and TS3. We assume that TS1 contains a missing block for $ts \in [115, 170]$. The block recovery we want to achieve should learn from the history of the time series that contains the missing block, i.e., TS1, and the history of the correlated time series TS2 and TS3.

Figure. 1.3 shows the result of the application of REBOM to recover the missing block in TS1. Our proposed solution uses the history of TS1 and exploits its correlation to TS2 and TS3 to perform an accurate recovery of the missing block. In fact, REBOM detects that the first peak of TS1 is more correlated to the first peak of TS2 than the first peak of TS3. It also detects that the third peak of TS1 is more correlated to the third peak of TS3 than the third peak of TS2. The shape of the missing peak is thus recovered by learning the second peak of TS2 and TS3 at the same time. The amplitude of the missing block is recovered using the first and the third peaks of

[1]http://www.ifi.uzh.ch/dbtg/research/amv/proto.html

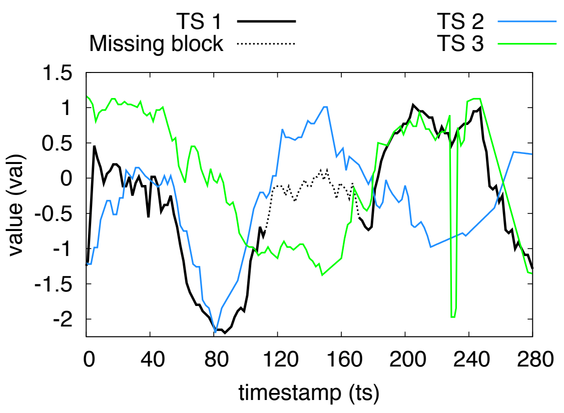


Figure 1.2: Example of Three Hydrological Time Series. TS1 contains a missing block for $ts \in [115, 170]$.

TS1.

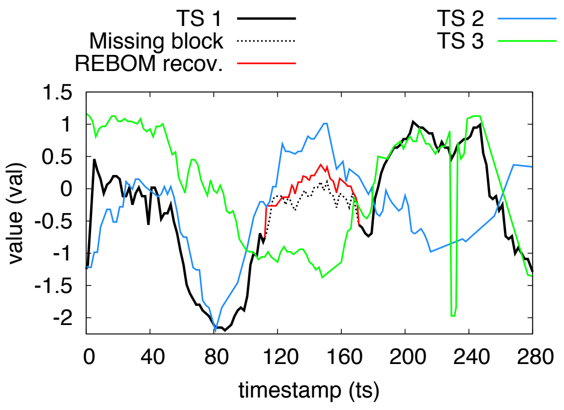


Figure 1.3: Recovery of the Missing Block using REBOM.

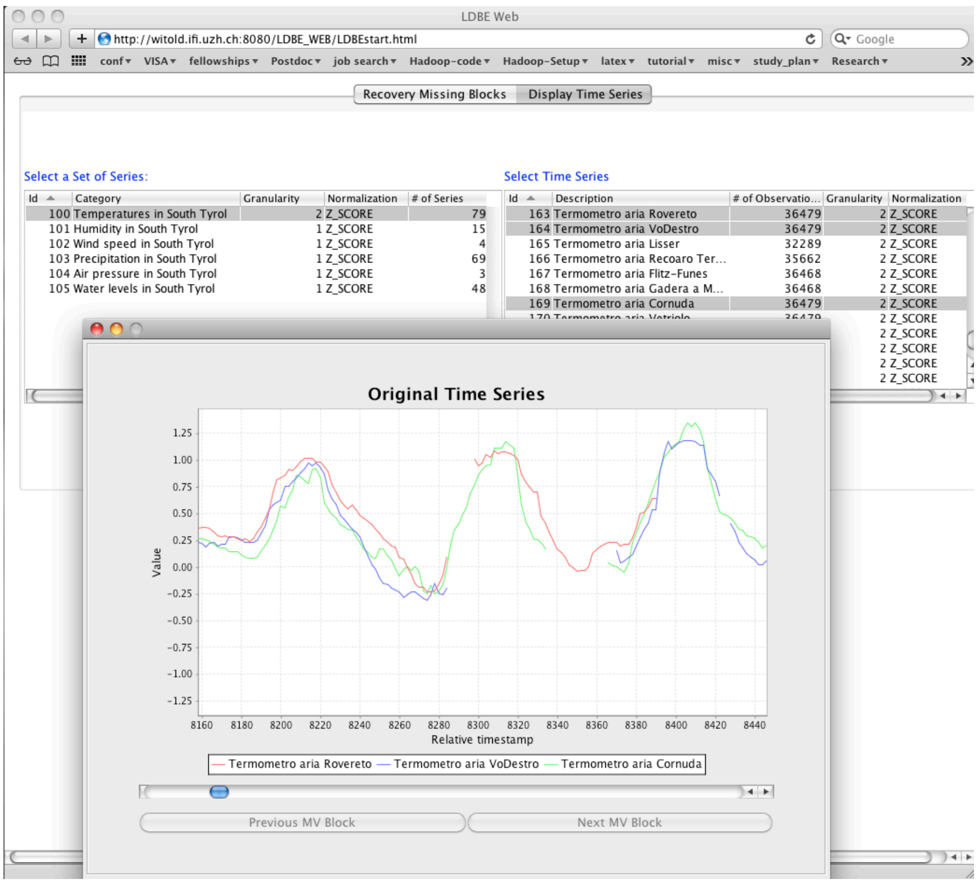
The greedy approach of REBOM is based on a global ranking vector, which ranks the top-k correlated time series considering all observations of the time series, and a partial ranking vector, which ranks the top-k correlated time series by considering only observations with timestamps of missing values. The two ranking vectors are iteratively used to select the set of time series that maximally reduce the recovery error. They are also used to terminate the iterative process, i.e., in case the two ranking vectors are equal.

The proposed recovery technique has been implemented as an online graphical tool[2]. In what follows we present REBOM (REcovery of BlOcks of Missing values), a tool for visualizing time

[2]http://www.ifi.uzh.ch/dbtg/research/amv/proto/rebom.html

series and recovering their missing values.

REBOM performs the recovery of missing values on time series data preloaded into a database server. The raw data can be normalized using two normalization techniques, i.e., z-score normalization and the min-max normalization. The proposed tool offers the possibility to display the time series before and after the recovery process. The recovery is performed by considering time series from the same set, e.g., a temperature time series is recovered using only temperature time series. REBOM offers also the possibility to navigate through missing blocks.



Select a Set of Series:

Id	Category	Granularity	Normalization	# of Series
100	Temperatures in South Tyrol	2	Z_SCORE	79
101	Humidity in South Tyrol	1	Z_SCORE	15
102	Wind speed in South Tyrol	1	Z_SCORE	4
103	Precipitation in South Tyrol	1	Z_SCORE	69
104	Air pressure in South Tyrol	1	Z_SCORE	3
105	Water levels in South Tyrol	1	Z_SCORE	48

Select Time Series

Id	Description	# of Observatio...	Granularity	Normalization
163	Termometro aria Rovereto	36479	2	Z_SCORE
164	Termometro aria VoDestro	36479	2	Z_SCORE
165	Termometro aria Lisser	32289	2	Z_SCORE
166	Termometro aria Recoaro Ter...	35662	2	Z_SCORE
167	Termometro aria Flitz-Funes	36468	2	Z_SCORE
168	Termometro aria Gadera a M...	36468	2	Z_SCORE
169	Termometro aria Cornuda	36479	2	Z_SCORE
170	Termometro aria Vetriolo	36479	2	Z_SCORE
			2	Z_SCORE
			2	Z_SCORE
			2	Z_SCORE
			2	Z_SCORE

Figure 1.4: Display of Time Series using REBOM.

REBOM offers the following key features:

Display of time series REBOM supports the display of the entire history of multiple time series at a time. Time series are graphically displayed with different colors using line charts where the x-axis represents the timestamps and the y-axis represents the values. Figure 1.4 illustrates the graphical representation of three real world temperature time series measured in the region of South Tyrol in Italy. The values are measured every 30 minutes. The display feature helps to highlight the trend similarities between time series.

Recovery of missing blocks The main goal of REBOM is to recover missing blocks in real world irregular time series. First, the time series are aligned to have the same starting timestamp. Then, the recovery is performed using the time series that contains the missing blocks and its most correlated time series. At the end of the recovery, all the time series used during the recovery process are displayed in the same window. The right hand side picture of Figure 1.5 illustrates the recovery of a missing block in the original black time series (middle picture of Figure 1.5) using REBOM. The latter uses the black time series together with its 4 most correlated time series to produce the block recovery.

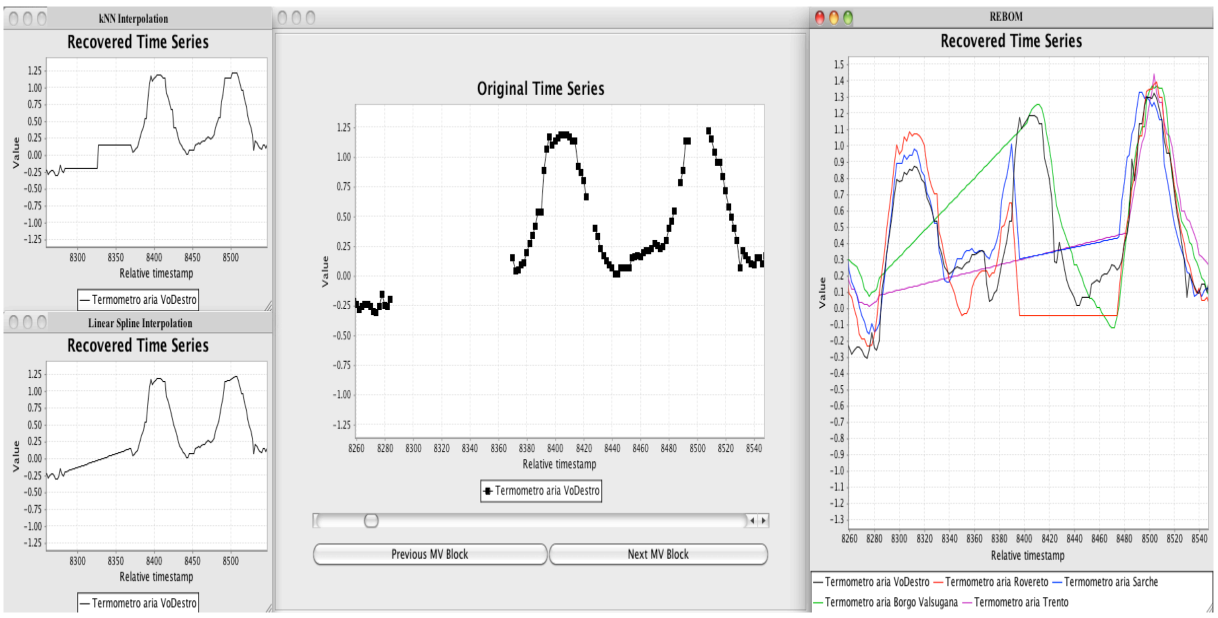


Figure 1.5: Recovery of a Missing Block using REBOM.

Browsing of missing blocks REBOM provides the user with navigational controls to browse the missing blocks (see the middle picture of Figure 1.5). These controls allow browsing forward and backward missing blocks in one time series. This feature helps the user (1) to directly access the missing blocks and the recovered ones (2) to highlight the local trend similarities between the time series that contains the missing blocks and its most correlated time series.

Comparison with statistical recovery techniques REBOM displays also the block recovery result of other statistical recovery techniques, e.g., linear spline interpolation, k nearest neighbors (kNN) interpolation, etc. Each recovery technique is displayed in a separate window. This feature helps to graphically compare the block recovery produced by our proposed approach against some of the state-of-the-art recovery techniques. The left hand side picture of Figure 1.5 illustrates the block recovery produced respectively by the kNN interpolation and the linear spline interpolation. These two techniques use only the black time series in the recovery.

1.2.2 Recovery using Mixed Correlated Time Series

In the second part of this thesis we introduce a recovery solution that learns from highly and lowly correlated time series to recover missing blocks in irregular time series. Including lowly correlated time series that exhibit trend and/or shape similarities in the recovery process could increase the recovery accuracy. For example, Foehn is a warm wind that yields lowly correlated temperature time series that exhibit shape similarities. These lowly correlated time series could be exploited in the recovery process. However, lowly correlated time series should be given less weight than the highly correlated ones in the recovery process. The Mean Squared Error (MSE) is used to quantify the weight assigned to each of the input time series during the recovery process.

The proposed recovery solution is based on the Centroid Decomposition (CD) technique. Unlike many other matrix decomposition techniques such as SVD, CD uses correlated input time series to produce correlated output vectors. This key property follows from the fact that CD technique does not perform a vector orthogonalization during the decomposition process. The correlated output vectors are used to recover the missing blocks.

The proposed solution produces an MSE relative reduction of the recovered block proportionally to the correlation of the columns of the input matrix. As a result, the proposed solution exploits all the input time series while taking into account the initial correlation value of each of them. This makes our solution suitable for learning from time series of different types, e.g., a humidity time series measured at one place can be used to recover missing blocks in temperature time series measured at the same place in case both time series exhibit some similarities.

1.2.3 Scalable Centroid Decomposition

In the third part of this thesis we reduce the space complexity of CD technique. The original algorithm to compute CD constructs a square correlation matrix that requires quadratic space complexity. We propose an algorithm that reduces the space complexity from quadratic to linear. This space reduction is important to be able to apply the CD based recovery using the entire history of the time series and without the need to apply any prior segmentation. The space reduction makes it possible to perform a block recovery on time series with a long history.

The key idea behind reducing the space complexity is to transform the optimization problem that CD solves into a new and equivalent optimization problem that uses a sequence of weight vectors instead of the correlation matrix. The weight vectors are iteratively computed using a greedy process. At each iteration, the weight vector that produces the local optimum is chosen. A key property of the proposed solution is that the local optimum is monotonically increasing after each iteration. At the end of the process, the solution returns the global optimum.

Example 3. Consider a matrix $\mathbf{X}$ where each column represents the set of values of observations of each time series. The values are ordered with respect to their timestamps. Figure 1.6 illustrates the state-of-the-art solution, called *QSV*, to compute the Centroid Decomposition of X. In Step 1, *QSV* takes the input matrix $\mathbf{X}$ and constructs a square correlation matrix $\mathbf{X}{\cdot}\mathbf{X}^T$. Therefore, the space complexity of *QSV* is quadratic with the number of rows of X. In Step 2, *QSV* returns the vector Z that solves the optimization problem.

$$\underbrace{\underbrace{\begin{bmatrix}2&-2\\0&3\\-4&2\\1&5\end{bmatrix}}_{\mathbf{X}}}_{\text{Input}} \overset{Step1}{\overbrace{\rightarrow}} \underbrace{\underbrace{\begin{bmatrix}8&-6&-12&-8\\-6&9&6&15\\-12&6&20&6\\-8&15&6&26\end{bmatrix}}_{\mathbf{X}\cdot\mathbf{X}^T}}_{\text{Intermediate computation}} \overset{Step2}{\overbrace{\rightarrow}} \underbrace{\underbrace{\begin{bmatrix}-1\\1\\1\\1\end{bmatrix}}_{Z}}_{\text{QSV output}}$$

Figure 1.6: Illustration of the State-of-the-art Solution.

Figure 1.7 illustrates our approach, called SSV, to compute the Centroid Decomposition of $\mathbf{X}$. In step 1, SSV takes the input matrix $\mathbf{X}$ and constructs a sequence of weight vectors $V^{(i)}$. Note that only one weight vector is stored in memory at a time. In Step 2, SSV returns the vector Z that solves the optimization problem. The returned Z is the same as the one produced by QSV. The space complexity of our solution is linear with the number of rows of X.

$$\underbrace{\underbrace{\begin{bmatrix}2 & -2\\ 0 & 3\\ -4 & 2\\ 1 & 5\end{bmatrix}}_{\mathbf{X}}}_{\text{Input}} \overbrace{\rightarrow}^{Step1} \underbrace{\underbrace{\begin{bmatrix}-26\\ 15\\ 0\\ 13\end{bmatrix}}_{V^{(1)}}\underbrace{\begin{bmatrix}-42\\ 27\\ 24\\ 29\end{bmatrix}}_{V^{(2)}}}_{\text{Intermediate computation}} \overbrace{\rightarrow}^{Step2} \underbrace{\underbrace{\begin{bmatrix}-1\\ 1\\ 1\\ 1\end{bmatrix}}_{Z}}_{\text{SSV output}}$$

Figure 1.7: Illustration of our Solution.

To prove the correctness of the proposed solution, we first divide the optimization problem into two sub-problems: the sub-problem that involves the initial solution and the sub-problem that involves the remaining choices. Then, we prove that the initial choice of our solution is part of the global optimal solution. Finally, we prove that by omitting the initial choice, the remaining choices lead to the global optimum of the remaining sub-problem. As a result, our solution computes the global optimal solution and thus, produces a correct result.

The greedy process used by our algorithm terminates. In fact, the local optima computed at each iteration by our solution are increasingly monotonic. Thus, our solution terminates in all possible cases.

1.3 Organization of the Thesis

This thesis is based on a collection of papers. A bibliography for all chapters is given at the end of the thesis.

In Chapter 2 and in Chapter 4, SVD and CD techniques are respectively defined. In Chapter 3, the connection between the input matrix and the output vectors, produced respectively by SVD and CD techniques, is defined and described more in details. In Chapter 2 time series are defined using their timestamps and their values whereas in Chapter 3 time series are defined using only their values.

The organization of the thesis is as follows:

Chapter 2 REBOM

Mourad Khayati and Michael H. Böhlen. REBOM: Recovery of Blocks of Missing Values in Time Series. In *Proceedings of the 18th International Conference on Management of Data*, COMAD '12, pages 44–55, 2012.
`DOI:` http://dx.doi.org/10.5167/uzh-72323

Chapter 3 Using Lowly Correlated Time Series to Recover Missing Values

Mourad Khayati, Michael H. Böhlen, and Philippe Cudré-Mauroux. Using Lowly Correlated Time Series to Recover Missing Values in Time Series: a Comparison between SVD and CD. Accepted at the *14th International Symposium on Spatial and Temporal Databases*, SSTD '15, August 2015.

Chapter 3 Memory-efficient Centroid Decomposition

Mourad Khayati, Michael H. Böhlen, and Johann Gamper. Memory-efficient Centroid Decomposition for Long Time Series. In *Proceedings of the 30th IEEE International Conference on Data Engineering*, ICDE '14, pages 100–111, IEEE Computer Society, 2014.
`DOI:` http://dx.doi.org/10.1109/ICDE.2014.6816643

CHAPTER 2

REBOM

Abstract

The recovery of blocks of missing values in regular time series has been addressed by model-based techniques. Such techniques are not suitable to recover blocks of missing values in irregular time series and restore peaks and valleys. We propose REBOM (REcovery of BlOcks of Missing values): a new technique that reconstructs shape, amplitude and width of missing peaks and valleys in irregular time series. REBOM successfully reconstructs peaks and valleys by iteratively considering the time series itself and its correlation to multiple other time series. We provide an iterative algorithm to recover blocks of missing values and analytically investigate its monotonicity and termination. Our experiments with real world hydrological and synthetic data confirm that the recovery of blocks of missing values in irregular time series of REBOM is more accurate than existing methods.

2.1 Introduction

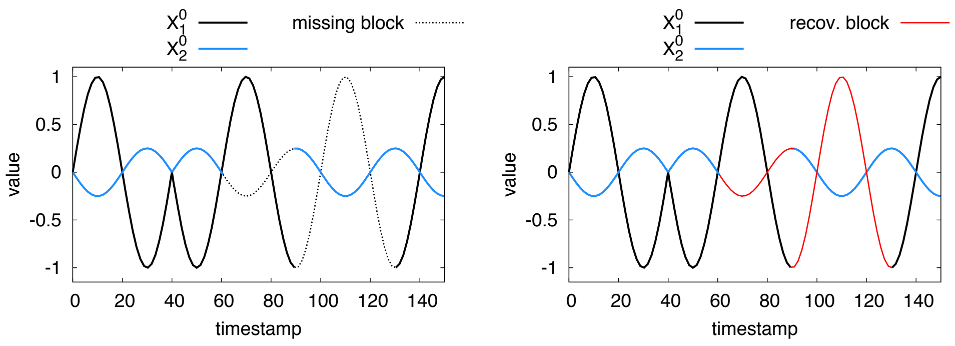
Time series data arise in a variety of domains, such as environmental, telecommunication, financial, and medical data. For example, in the field of hydrology, sensors are used to capture environmental phenomena including temperature, air pressure, and humidity at different points in time. For such data, it is not uncommon that more than 20% of the data is missing as blocks, i.e., multiple consecutive measurements are missing.

Existing techniques effectively recover blocks of missing values in regular time series, i.e., time series containing peaks and valleys with a possibly varying frequency or amplitude that follow one or more periodic models, e.g., the sinus model where the frequency varies over time. The recovery accuracy of these techniques decreases for irregular time series, i.e., time series containing peaks and valleys that do not follow any model. In this work, we address the problem of finding the optimal recovery of blocks of missing values in irregular time series. We propose REBOM (REcovery of BlOcks of Missing values), a new data driven recovery technique for blocks of missing values that is able to restore missing peaks and valleys. We use the correlation [MNL10] between time series to recover blocks of missing values. Intuitively, time series that tend to change their peaks and valleys simultaneously are correlated and we use the Pearson coefficient to quantify this correlation.

REBOM is an iterated low rank Singular Value Decomposition (SVD). We decompose a matrix $\mathbf{M}$ of correlated time series, where missing values have been initialized through linear interpolation combined with nearest neighbor imputation, into the product of three matrices $\mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^T$. By nullifying the smallest singular value of $\mathbf{\Sigma}$, we emphasize the impact of the correlation between time series. The subsequent matrix multiplication yields an approximation of $\mathbf{M}$ that better approximates the missing values. After each iteration, the ranking of the most correlated time series with respect to the time series to recover, is updated. The iterative recovery terminates if the total ranking, which is determined by considering all observations of the time series, is identical to the partial ranking, which is determined by considering only observations with timestamps of missing values. If the total and the partial ranking are equal, the correlation can no longer be used to improve the recovery of missing values.

Problem definition: Assume a set of n irregular correlated time series $\mathbf{X}^0 = \{X_1^0, X_2^0, \ldots, X_n^0\}$ where $X_1^0, X_2^0, \ldots, X_n^0$ contain blocks of missing values. We propose a recovery method that determines, in j iterations, a set of time series $\widetilde{\mathbf{X}}^j = \{\widetilde{X}_1^j, \widetilde{X}_2^j, \ldots, \widetilde{X}_n^j\}$ where the missing blocks of $X_1^0, X_2^0, \ldots, X_n^0$ have been restored.

The result of REBOM for the recovery of peaks and valleys for two correlated time series is illustrated in Figure 2.1. Each time series is displayed as a $2d$ plot where the x-axis shows the timestamp t and the y-axis shows the value v for a given t. X_1^0 represents an air pressure time series and contains a missing block for the time range $]90, 130[$. X_2^0 represents a temperature time series that contains a missing block for the time range $]60, 90[$. REBOM can be used to restore the missing blocks of X_1^0 and X_2^0.



(a) Original Time Series

(b) Restoration of Missing Blocks of X_1^0 and X_2^0

Figure 2.1: Recovery Performed by REBOM

Figure 2.1 illustrates that REBOM accurately recovers *shape*, *amplitude* and *width* of the missing blocks. REBOM detects that the peaks and valleys of X_1^0 and X_2^0 are correlated (high pressure corresponds to low temperature and vice versa). The shape and the width of the missing block are recovered from the position of the local extrema of X_1^0 with respect to the local extrema of the correlated time series X_2^0. The amplitude of the missing block of X_1^0 is recovered based on the two preceding peaks of X_1^0.

At the technical level, we show how to iterate the low rank SVD and we analytically investigate the main properties of the method. The main contributions are as follows:

- We propose REBOM: an iterated low rank SVD that iteratively refines the initial recovery of missing values.
- We propose a greedy algorithm that repeatedly selects a time series with missing values that have been initialized and uses the k most correlated time series to iteratively refine the recovery of the missing values.
- We prove that our greedy algorithm is stepwise monotonic, i.e., the accuracy of the recovery increases by choosing, at each step, the most correlated time series. The algorithm

terminates when the set of the most correlated time series does not change anymore.

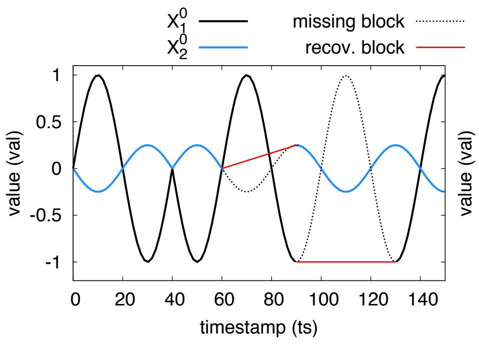
- We empirically show that the recovery accuracy of REBOM is invariant to the initial recovery. Different initialization methods lead to the same recovery accuracy but with different number of iterations.
- We present an experimental evaluation of the accuracy of our technique that compares REBOM to the state-of-the-art techniques for the recovery of blocks of missing values. The results show the superiority of our algorithm for the restoration of peaks and valleys.

The rest of the chapter is organized as follows. Section 2.2 reviews related work on reduction methods and existing techniques for imputing missing values. Section 2.3 defines the initialization method and describes the basics of the low rank SVD. Section 2.4 introduces and discusses REBOM and its properties. Section 2.5 empirically compares the results of REBOM to other techniques proposed in the literature for the recovery of blocks of missing values.

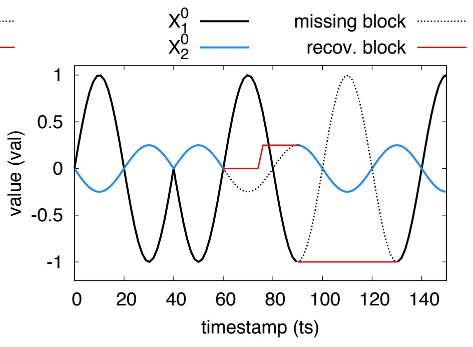
2.2 Related Work

Prediction models such as Maximum Likelihood Estimation (MLE) [STP07], Bayesian Networks (BN) [RS04, HCRC11] and Expectation Maximization (EM) [SJ03] were used to estimate single missing values or small blocks of missing values in time series. These techniques are parametric and require a specific type of data distribution, e.g., Gaussian distribution. Therefore, they only perform well for the recovery of blocks of missing values in regular time series where peaks and valleys follow a periodic model of constant frequency and amplitude.

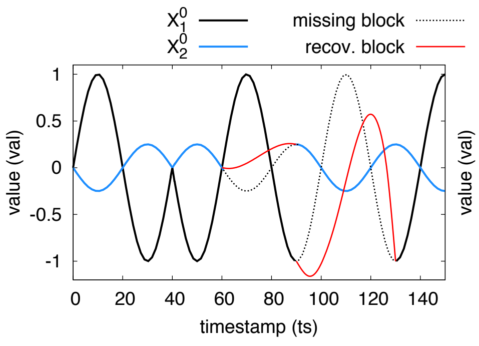
Li et al. [LMPF09] presented an approach called DynaMMo that is based on Expectation Maximization (EM) and Kalman Filter [JCW04]. This technique is intended to recover missing blocks in non linear time series that contain peaks and valleys. DynaMMo allows to use one reference time series in addition to the time series that contains the missing block. The Kalman Filter uses the data of the time series that contains missing blocks together with a reference time series, to estimate the current state of the missing blocks. This estimation is performed as a multi step process that uses two different estimators. The first estimator represents the current state and the second estimator represents the initial state and the error of the estimation. For every step of the process, an EM method predicts the value of the current state and then the two estimators are used to refine the predicted values of the current state and to maximize their likelihood. DynaMMo



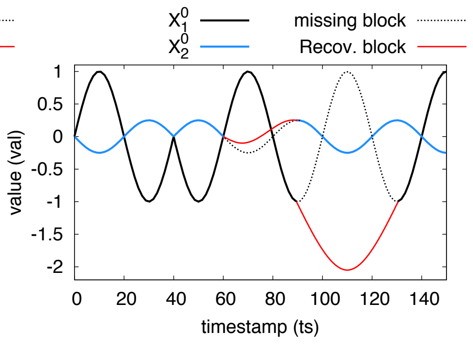
(a) Linear Spline Recovery



(b) k Nearest Neighbor Recovery



(c) Polynomial Regression Recovery



(d) Cubic Spline Recovery

Figure 2.2: Recovery using Different Techniques

does not allow to use more than one reference time series for the block recovery. DynaMMO performs an accurate block recovery for any type of regular time series. The accuracy of the block recovery decreases for irregular time series (cf. Section 2.5).

Techniques that rely on basic statistical methods such as mean imputation, piecewise approximation (linear spline, cubic spline, ...) [DTS+08, CCL+07], regression [GH07, YSJ+00] and k Nearest Neighbors [SK98] have been proposed for the recovery of blocks of missing values. Figures 2.2(a) and 2.2(b) illustrate the block recovery performed respectively by linear spline and k nearest neighbor using values at $ts = 60$ and $ts = 90$. The two preceding techniques are not able to accurately recover any of the two missing blocks in X_1^0 and X_2^0. In Figure 2.2(c) we choose the best order of polynomial regression for the the recovery of each missing block in X_1^0, i.e., order three for the first block and order four for the second one. The polynomial regression replaces missing values by points lying on the line that minimizes the regression error of points. The polynomial regression performs a bad recover of the first missing block and partially recovers the second missing block of X_1^0. The cubic spline technique finds a third order polynomial that connects three successive values. Figure 2.2(d) shows that the cubic spline replaces the missing block by a block opposite to the one that precedes the missing block. Cubic spline partially recovers the first missing block of X_1^0 and performs a bad recovery for the second block. All basic methods are not suitable techniques for block recovery in regular time series where peaks and valleys follow a periodic model of varying amplitude or frequency, or in irregular time series.

Kurucz et al. [MKT07] proposed a technique based on EM and Singular Value Decomposition (SVD) [Mey00, Kal96, Bra02, ABB00] for comparing recommender systems where one of them contains missing values. A recovery of the missing values is performed before the comparison process. Each recommender system is represented by one column of values in a rating matrix which is decomposed using SVD. The result of the decomposition is modified using a method called gradient boosting [SZB$^+$11]. The EM algorithm is then applied to refine the result of gradient boosting. The proposed solution dynamically discovers data dependencies from coordinate axes that represent the recommender systems and used more than one recommender system. However, the application of gradient boosting on different recommender systems looses the dependencies among the original values of recommender systems. Therefore, this technique yields bad results for block recovery in case where more than one recommender system contains missing blocks.

Tree-based methods were proposed to impute missing values. Ding and Simonoff [DS10], and He [He06] present an overview of tree classification methods that are able to replace missing values in time series. These trees find the optimal way to classify missing values using a regression approach and are called Classification and Regression Trees (CART). These techniques are designed to create a classification of the missing values. Missing values that belong to the same class will be recovered with the same value. Therefore, these methods are not able to effectively restore missing peaks and valleys in regular and irregular time series.

2.3 Preliminaries and Background

2.3.1 Notation

We use the following notation: sets and vectors are upper-case, matrices are upper-case bold, and elements of sets and matrices are lower-case. A time series $X_1 = \{x_1, x_2, \ldots, x_n\}$ is a set of n observations. Each observation x_j from X_1 is a pair (t_j, v_j) where t_j and v_j are respectively the timestamp and the value of the observation. $T_1 = \{t \mid (t, \_) \in X_1)\}$ denotes the set of all timestamps from X_1; $M_1 = \{v \mid (\_, v) \in X_1)\}$ denotes the vector of all values from the time series X_1. A time series X_1 with missing values that have not been recovered yet, is denoted as X_1^0.

2.3.2 Preprocessing of Time Series

The first preprocessing step uses basic statistical methods to initialize all missing values. After the initialization, the timestamps of all time series are aligned.

Definition 1 (Missing timestamps)**.** Given a set of n time series $\{X_1^0, \dots, X_n^0\}$, the set of missing timestamps of time series X_i^0 with respect to the timestamps of the other time series is $T_i^0 = \{t \mid ((t, \_) \in X_1^0 \vee \dots \vee (t, \_) \in X_n^0) \ \wedge (t, \_) \notin X_i^0\}$.

Note that missing timestamps of one time series have to be present in at least another time series. Timestamps missing in all time series are not considered. An additional preprocessing step can be added if such timestamps shall be recovered as well.

$X_1^1 = \{(t_1, v_1), ..., (t_n, v_n)\}$ is the initial recovery of X_1^0 iff $\forall i \in \{1, \dots, n\}$

$$(t_i, v_i) = \begin{cases} (t_i, v_i) \textbf{ if } (t_i, v_i) \in X_1^0 \\ \textbf{Else} \begin{cases} (t_i, v) \textbf{ if } (s(t_i), \_) \notin X_1^0, \\ \qquad\qquad (p(t_i), v) \in X_1^0 \\ (t_i, v) \textbf{ if } (p(t_i), \_) \notin X_1^0, \\ \qquad\qquad (s(t_i), v) \in X_1^0 \\ (t_i, \frac{(t_i - p(t_i))(s(v_i) - p(v_i))}{s(t_i) - p(t_i)} + s(v_i)) \\ \qquad\qquad \textbf{otherwise} \end{cases} \end{cases}$$

$p(t_i) = max\{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j < t_i\}$ is the predecessor of timestamp t_i in X_1^0 and $s(t_i) = min\{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j > t_i\}$ is the successor timestamp of t_i in X_1^0. Similarly, $p(v_i) = \{v_j \mid (t_j, \_) \in X_1^0 \wedge t_j = p(t_i)\}$ is the predecessor of value v_i in X_1^0 and $s(v_i) = \{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j = s(t_i)\}$ is the successor value of v_i in X_1^0. Thus, the initial recovery of the missing values is a linear interpolation. If the missing values occur as the first or the last elements of X_1^0, we use the nearest neighbor imputation.

Two time series X_1^1 and X_2^1 with initialized missing values define a set of multidimensional points: $\{(v, v') \mid (t, v) \in X_1 \wedge (t, v') \in X_2\}$. The second preprocessing step constructs a matrix with n m-dimensional points from m time series with n observation each.

Example 4. Figure 2.3 shows two time series X_1^0 and X_2^0 with missing values, the initialized time series X_1^1 and X_2^1, and the set of multidimensional points $\mathbf{M}$. The initialized missing values are highlighted in gray.

X_1^0		X_2^0	
t	v	t	v
0	0	0	0
10	1	10	-0.25
20	0	20	0
30	-1	30	0.25
40	0	40	0
50	-1	50	0.25
60	0	60	0
70	1	90	0.25
80	-1	100	0
90	-1	110	-0.25
130	-1	120	0
140	0	130	0.25
150	1	140	0
		150	-0.25

$\Rightarrow$

X_1^1		X_2^1	
t	v	t	v
0	0	0	0
10	1	10	-0.25
20	0	20	0
30	-1	30	0.25
40	0	40	0
50	-1	50	0.25
60	0	60	0
70	1	70	0.08
80	-1	80	0.16
90	-1	90	0.25
100	-1	100	0
110	-1	110	-0.25
120	-1	120	0
130	-1	130	0.25
140	0	140	0
150	1	150	-0.25

$\Rightarrow$

$\mathbf{M}$	
M_1	M_2
0	0
1	-0.25
0	0
-1	0.25
0	0
-1	0.25
0	0
1	0.08
-1	0.16
-1	0.25
-1	0
-1	-0.25
-1	0
-1	0.25
0	0
1	-0.25

Figure 2.3: Original Time Series X_1^0, X_2^0; Initialized Time Series X_1^1, X_2^1; Multidimensional Points $\mathbf{M}$.

From Definition 1 we get $T_1^0 = \{100, 110, 120\}$ and $T_2^0 = \{70, 80\}$.

2.3.3 Low Rank Matrix Decomposition

Singular Value Decomposition

The Singular Value Decomposition (SVD) is a matrix decomposition method that decomposes a matrix $\mathbf{M}$ into three matrices $\mathbf{L}$, $\mathbf{\Sigma}$ and $\mathbf{R}^T$. The product of the three matrices is equal to $\mathbf{M}$.

Definition 2 (SVD). A matrix $\mathbf{M} = [M_1|M_2|\ldots|M_n] \in \mathcal{R}^{m \times n}$ can be decomposed into a product of three matrices:

$$
\begin{aligned}
SVD(\mathbf{M}) &= \mathbf{L} \times \mathbf{\Sigma} \times \mathbf{R}^T \\
&= \underbrace{\begin{bmatrix} U_1 \Big| \dots \Big| U_p \end{bmatrix}}_{\mathbf{U}(m \times p)} \times \underbrace{\begin{bmatrix} \begin{array}{c|c|c} \sigma_1 & \dots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \dots & \sigma_n \end{array} \end{bmatrix}}_{\mathbf{\Sigma}(p \times n)} \times \underbrace{\begin{bmatrix} \begin{array}{c} V_1^T \\ \hline \vdots \\ \hline V_n^T \end{array} \end{bmatrix}}_{\mathbf{V}^T(n \times n)}
\end{aligned}
$$

Where:

- $p = \min(m, n)$.
- $\mathbf{U}$: is an $m \times p$ orthogonal matrix whose columns are the orthonormal eigen vectors of $\mathbf{MM}^T$ ($\mathbf{U}^T\mathbf{U} = I$, where I is the identity matrix). The eigen vectors of $\mathbf{MM}^T$ are computed by solving $Det(\sigma\mathbf{I} - \mathbf{MM}^T) = 0$ where $Det(\mathbf{M})$ is the determinant of matrix $\mathbf{M}$.
- $\mathbf{\Sigma}$: is a $p \times m$ diagonal matrix that contains positive singular values of $\mathbf{M}$. The diagonal entries σ_i of $\mathbf{\Sigma}$ are the square roots of the eigen values of $\mathbf{M}^T\mathbf{M}$ and are ranked in decreasing order such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$.
- $\mathbf{V}$: is an $n \times n$ orthogonal matrix having as columns orthonormal eigen vectors of $\mathbf{M}^T\mathbf{M}$. The latter vectors are computed by solving $Det(\sigma\mathbf{I} - \mathbf{M}^T\mathbf{M}) = 0$.
- A singular value σ_i defines the variance of vector U_i along dimension V_i^T, i.e., $var(U_i) = \sigma_i$.

Example 5. Consider time series X_1^1 and X_2^1 from Figure 2.3. Figure 3.1 illustrates the SVD of $\mathbf{M}$.

Dimensionality Reduction

SVD allows to perform a dimensionality reduction from a dimension n to a lower dimension r. The dimensionality reduction is performed by nullifying the $n - r$ smallest singular values from matrix $\mathbf{\Sigma}$, where $0 < \sigma_r < \sigma_n$. Figure 2.5 illustrates the dimensionality reduction for $r = n - 1$, i.e., the smallest singular value of $\mathbf{\Sigma}$ is nullified. We write $SVD_r(\mathbf{M})$ for the result of a low rank

$$SVD(\mathbf{M}) = \underbrace{\begin{bmatrix} 0 & 0 \\ 0.31 & -0.22 \\ 0 & 0 \\ -0.31 & 0.22 \\ 0 & 0 \\ -0.31 & 0.22 \\ 0 & 0 \\ -0.30 & -0.11 \\ -0.30 & 0.04 \\ -0.31 & 0.22 \\ -0.30 & -0.27 \\ -0.30 & -0.75 \\ -0.30 & -0.27 \\ -0.31 & 0.22 \\ 0.00 & 0.00 \\ 0.31 & -0.22 \end{bmatrix}}_{\mathbf{U}} \times \underbrace{\begin{bmatrix} 3.35 & 0 \\ 0 & 0.51 \end{bmatrix}}_{\mathbf{\Sigma}} \times \underbrace{\begin{bmatrix} 0.99 & -0.14 \\ 0.14 & 0.99 \end{bmatrix}}_{\mathbf{V}^T}$$

Figure 2.4: Example of Singular Value Decomposition

SVD of a matrix M. REBOM uses the low rank SVD for improving the initial imputation of the missing values as described in the next section.

$$SVD_r(\mathbf{M}) = \underbrace{\left[U_1 \middle| \dots \middle| U_p \right]}_{\mathbf{U}(m \times p)} \times \underbrace{\left[\begin{array}{c|c|c|c} \sigma_1 & \dots & 0 & 0 \\ \hline \vdots & \ddots & \vdots & 0 \\ \hline 0 & \dots & \sigma_r & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right]}_{\mathbf{\Sigma}_r(p \times n)} \times \underbrace{\left[\begin{array}{c} V_1^T \\ \hline \vdots \\ \hline V_n^T \end{array}\right]}_{\mathbf{V}^T(n \times n)}$$

Figure 2.5: Illustration of Dimensionality Reduction

2.4 REBOM

REBOM combines the characteristics of a time series with missing values with the characteristics of its most correlated time series to recover blocks of missing values in irregular time series.

2.4.1 Correlation Ranking Matrix

We define the top-k ranking matrix to capture the correlation between different time series. The correlation is defined over all values of the first vector of the matrix with respect to all values of another vector. The Pearson coefficient is used as a correlation metric. Given two vectors $M_i = [m_{i_1}, m_{i_2}, \ldots, m_{i_n}]$ and $M_j = [m_{j_1}, m_{j_2}, \ldots, m_{j_n}]$ of the same length n, the Pearson correlation coefficient ρ of M_i with respect to M_j is defined as follows:

$$\begin{aligned}\rho(M_i, M_j) &= \frac{cov(M_i, M_j)}{\sqrt{var(M_i)var(M_j)}} \\ &= \frac{\sum_{p=1}^{n}(m_{i_p} - \bar{m}_i)(m_{j_p} - \bar{m}_j)}{\sqrt{\sum_{p=1}^{n}(m_{i_p} - \bar{m}_i)^2 \sum_{p=1}^{n}(m_{j_p} - \bar{m}_j)^2}} \\ & \quad with\ \bar{m}_i = \frac{1}{n}\sum_{p=1}^{n} m_{i_p},\ \ \bar{m}_j = \frac{1}{n}\sum_{p=1}^{n} m_{j_p}\end{aligned}$$

$\rho(M_i, M_j)$ is undefined if all values of M_i and/or M_j are equal. The vectors of the correlation ranking matrix are ranked in decreasing order of the Pearson coefficient between the first vector and the remaining vectors.

Definition 3 (Top-k ranking matrix)**.** Let $\mathbf{M} = [M_1, M_2, \ldots, M_n]$ be a matrix of n vectors. $\mathbf{M}^{\textit{top-k}} = [M'_1, M'_2, \ldots, M'_k]$ is defined as the top-k ranking matrix of $\mathbf{M}$ with respect to a given vector that contains initialized missing values $M^1_p \in \mathbf{M}$ iff:

- $\mathbf{M}^{\textit{top-k}}$ contains the k vectors that are most correlated to M^1_p: $\forall M'_i \in \mathbf{M}^{\textit{top-k}}\, \forall M_j \in \mathbf{M} \setminus \mathbf{M}^{\textit{top-k}} : |\rho(M'_i, M^1_p)| \geq |\rho(M_j, M^1_p)|$
- The elements of $\mathbf{M}^{\textit{top-k}}$ are sorted by their correlation coefficient to M^1_p : $\forall 1 \leq i < k$: $|\rho(M'_i, M^1_p)| \geq |\rho(M'_{i+1}, M^1_p)|$

For each matrix $\mathbf{M}^{\textit{top-k}}$ we define a corresponding top-k ranking vector $\rho_{\mathbf{M}^{\textit{top-k}}} = [\rho(M^1_p, M^{\textit{top-k}}_1), \rho(M^1_p, M^{\textit{top-k}}_2), \ldots, \rho(M^1_p, M^{\textit{top-k}}_k)]$ for M^1_p with the l_1-norm $||\rho_{\mathbf{M}^{\textit{top-k}}}|| = \sum_{i=1}^{k}(|\rho(M^1_p, M^{\textit{top-k}}_i)|)$.

Example 6. Consider Figure 2.6 with $\mathbf{M} = [M_1, M_2, M_3, M_4]$ and top-3 ranking $\mathbf{M}^{\textit{top-3}} = [M_4, M_3, M_1]$ for M_4.

We get $\rho_{\mathbf{M}^{\textit{top-3}}} = [\rho(M_4, M_4), \rho(M_4, M_3), \rho(M_4, M_1)] = [1, 0.93, 0.87]$ and $||\rho_{\mathbf{M}^{\textit{top-3}}}|| = 2.8$.

$$\mathbf{M}=\begin{bmatrix} 4 & 6 & 3 & 2 \\ 5 & 7 & 1 & 3 \\ 6 & 7 & 9 & 8 \\ 7 & 6 & 8 & 7 \end{bmatrix}, \mathbf{M}^{top\text{-}3}=\begin{bmatrix} 2 & 3 & 4 \\ 3 & 1 & 5 \\ 8 & 9 & 6 \\ 7 & 8 & 7 \end{bmatrix}$$

Figure 2.6: Example of $\mathbf{M}^{top\text{-}3}$

2.4.2 Stepwise Correlation Monotonicity

We prove that REBOM is stepwise monotonic, i.e, choosing a bigger correlation value in the same iteration implies a bigger sum of variances. Lemma 1 states that the l_1-norm of a ranking vector $\rho_{\mathbf{M}}$ is proportional to the sum of the variance of vectors obtained by the application of the low rank SVD. In what follows a submatrix $\mathbf{M}_i = [M_{i_1}, M_{i_2}, \ldots, M_{i_k}]$ that contains k different columns of $\mathbf{M}$ is denoted as $\mathbf{M}_i \in \mathbf{M}$.

Lemma 1. *Let* $\mathbf{M}_i = [M_{i_1}, M_{i_2}, \ldots, M_{i_k}]$ *and* $\mathbf{M}_j = [M_{j_1}, M_{j_2}, \ldots, M_{j_k}]$ *be two different* $m \times k$ *matrices and let* $\mathbf{M}$ *be* $m \times n$ *matrix such that* $k \leq n$ *and* $\mathbf{M}_i, \mathbf{M}_j \in \mathbf{M}$. *Let* $\mathbf{N}_i = [N_{i_1}, N_{i_2}, \ldots, N_{i_k}] = SVD_r(\mathbf{M}_i)$ *and* $\mathbf{N}_j = [N_{j_1}, N_{j_2}, \ldots, N_{j_k}] = SVD_r(\mathbf{M}_j)$ *such that* $M_{i_1} = M_{j_1}$. *The* l_1*-norm of* $\rho_{\mathbf{M}_i}$ *and* $\rho_{\mathbf{M}_j}$ *is proportional to the sum of the variances of* $\mathbf{N}_i$ *and* $\mathbf{N}_j$*:*

$$||\rho_{\mathbf{M}_i}|| > ||\rho_{\mathbf{M}_j}|| \Rightarrow \sum_{p=1}^{k} var(N_{i_p}) > \sum_{p=1}^{k} var(N_{j_p})$$

Lemma 1 states that choosing a matrix with a bigger l_1-norm of the ranking vector implies a higher sum of variances over the vectors obtained by the application of SVD. Therefore, more correlated vectors of the input matrix yields a higher sum of the variances after the application of $SVD_r()$. Thus, by considering the top-k ranking matrix, the result of $SVD_r()$ maximizes the following objective function:

$$\sum_{N_i \in SVD_r(\mathbf{M})} var(N_i)$$

Proof. We prove that our algorithm is stepwise monotonic. We perform this proof by showing that the correlation matrix used is monotonic at every step of the algorithm. i) From Def. 2 (SVD) we know that the singular values define the variances along the vectors. ii) From the definition of the top-k ranking matrix we know that at every step of SVD, we take the matrix with the biggest l-1 norm of correlation. iii) From the definition of $SVD_r()$ we know that only the smallest variance will be nullified and the biggest ones will be kept. Using i), ii) and iii) we can deduce that our algorithm takes the biggest $||\rho_{\mathbf{M}_i}||$ in order to compute the biggest $\sum_{N_{i_j} \in \mathbf{N}_i} var(N_{i_j})$ where $\mathbf{N}_i = SVD_r(\mathbf{M}_i)$. Therefore, the bigger the correlation is, the bigger sum of variances we will obtain. This implies that the correlation used by the algorithm is stepwise monotonic.

□

Example 7. Consider matrix $\mathbf{M} = M_1, M_2, M_3, M_4$ from example 6 and the result matrix of the application of $SVD_r(\mathbf{M})$ as shown in Figure 2.7.

$$\mathbf{M}=\begin{bmatrix} 4 & 6 & 3 & 2 \\ 5 & 7 & 1 & 3 \\ 6 & 7 & 9 & 8 \\ 7 & 6 & 8 & 7 \end{bmatrix}, \mathbf{N}=\begin{bmatrix} 3.73 & 6.12 & 2.9 & 2.21 \\ 5.19 & 6.9 & 1 & 2.83 \\ 6.48 & 6.77 & 9.15 & 7.6 \\ 6.49 & 6.23 & 7.83 & 7.41 \end{bmatrix}$$

Figure 2.7: Example of a matrix and its $SVD_r()$ transformation

Let's take the example of $\mathbf{M}_1, \mathbf{M}_2 \in \mathbf{M}$ where $\mathbf{M}_1 = \{M_4, M_3, M_1\}$ and $\mathbf{M}_2 = \{M_4, M_2, M_1\}$, and let $\mathbf{N}_1 = SVD_r(\mathbf{M}_1)$ and $\mathbf{N}_2 = SVD_r(\mathbf{M}_2)$.

If we apply the computation with respect to vector M_4, we get $||\mathbf{M}_1|| = 2.8, ||\mathbf{M}_2|| = 2.06, \sum_{p=3}^{k} var(N_{1_p}) = 18.5$ and $\sum_{p=1}^{3} var(N_{2_p}) = 7.6$.

Lemma 1 holds for any other matrices $\mathbf{M}_i, \mathbf{M}_j \in \mathbf{M}$.

2.4.3 Iterative Recovery of REBOM

This section proves that REBOM terminates. In each step we compute the partial correlation ranking for the time series based on the missing values. If this partial ranking is the same as the global ranking, the recovery stops. For all missing values $t \in T_i^0$ (cf. Definition 1) the partial correlation $\widetilde{\rho}(M_i, M_j)$ is defined as follows:

$$\widetilde{\rho}(M_i, M_j) = \frac{\sum_{t=1}^{|T_i^0|}(m_{i_t} - \bar{m}_i)(m_{j_t} - \bar{m}_j)}{\sqrt{\sum_{t=1}^{|T_i^0|}(m_{i_t} - \bar{m}_i)^2 \sum_{t=1}^{|T_i^0|}(m_{j_t} - \bar{m}_j)^2}}$$

Where $|T_i^0|$ is the length of T_i^0. $\widetilde{\rho}(M_i, M_j)$ is undefined if all missing values of M_i or M_j are equal. The partial ranking matrix contains the partially most correlated vectors to the vector that contains the missing blocks to recover.

Definition 4 (Partial ranking matrix). Given a matrix $\mathbf{M} = [M_1, M_2, \ldots, M_n]$ of n vectors, $\widetilde{\mathbf{M}}^{\textit{top-k}} = [M'_1, M'_2, \ldots, M'_k]$ is defined as the top-k partial ranking matrix of $\mathbf{M}$ with respect to a given vector $M_p^1 \in \mathbf{M}$ iff:

- $\widetilde{\mathbf{M}}^{\textit{top-k}}$ contains the k vectors that are partially most correlated to M_p^1: $\forall M'_i \in \widetilde{\mathbf{M}}^{\textit{top-k}} \forall M_j \in \mathbf{M} \setminus \widetilde{\mathbf{M}}^{\textit{top-k}} : \widetilde{\rho}(M'_i, M_p^1) \geq \widetilde{\rho}(M_j, M_p^1)$
- The elements of $\widetilde{\mathbf{M}}^{\textit{top-k}}$ are sorted by their partial correlation coefficient to $M_p^1 : \forall 1 \leq i < k : \widetilde{\rho}(M'_i, M_p^1) \geq \widetilde{\rho}(M'_{i+1}, M_p^1)$

The top-k ranking and the top-k partial ranking are used to terminate the iterative recovery process.

Lemma 2 (Termination Condition). *Let* $\mathbf{N}_i^{\textit{top-k}} = [N_{i_1}, N_{i_2}, \ldots, N_{i_k}]$ *and let Ranking() be the ranking of vectors inside a matrix. If* $\mathbf{N}_i^{\textit{top-k}}$ *and its partial correlation matrix have the same ranking then the algorithm can not anymore create a matrix* $\mathbf{N}_{i+1}$ *with bigger sum of variances along its vectors. Formally:*

$$Ranking(\mathbf{N}_i^{\textit{top-k}}) = Ranking(\widetilde{\mathbf{N}}_i^{\textit{top-k}}) \Rightarrow$$

$$\sum_{N_{i_j} \in \mathbf{N}_i} var(N_{i_j}) > \sum_{N_{(i+1)_j} \in \mathbf{N}_{i+1}} var(N_{(i+1)_j})$$

Proof. We prove that our algorithm terminates after finding the matrix that has the maximum sum of variances along its vectors. We perform this proof by showing that a) REBOM computes a finite number of ranking; b) the matrix ranking determines the termination of the algorithm.

a) Finite number of rankings i) From Def. 2 (SVD) we know that the variance values obtained by SVD are ranked in increasing order in matrix Σ. ii) From [Lag91] we have that the singular values obtained by SVD are monotonic. Using i) and ii) it follows that the variance obtained by the decomposition is monotonic and thus: $N_{1_j} \in \mathbf{N}_1^{top\text{-}k} \wedge N_{2_j} \notin \mathbf{N}_2^{top\text{-}k} \Rightarrow N_{3_j} \notin \mathbf{N}_3^{top\text{-}k}$ where $\mathbf{N}_2^{top\text{-}k} = SVD_r(\mathbf{N}_1^{top\text{-}k})$ and $\mathbf{N}_3^{top\text{-}k} = SVD_r(\mathbf{N}_2^{top\text{-}k})$. Therefore, the number of rankings generated by REBOM is finite.

b) Ranking determines termination Let R_i be the ranking of matrix $\mathbf{N}_i$, $\widetilde{R}_i$ be the partial ranking of matrix $\mathbf{N}_i$ and R_{i+1} be the ranking of matrix $\mathbf{N}_{i+1}$ where $\mathbf{N}_{i+1} = SVD_r(\mathbf{N}_i)$. i) We have from Def. 3 that the correlation value determines the ranking inside a matrix. Thus, $||\rho_{\mathbf{N}_i}|| = ||\rho_{\mathbf{N}_{i+1}}|| \Rightarrow R_i = R_{i+1}$. ii) Since UMV algorithm (cf. Subsection 2.4.4) is updating only the missing values of the matrix, then $\widetilde{R}_i$ determines $||\rho_{\mathbf{N}_{i+1}}||$. It follows that $R_i = \widetilde{R}_i \Rightarrow ||\rho_{\mathbf{N}_i}|| = ||\rho_{\mathbf{N}_{i+1}}||$. Using i) and ii) we deduce by transitivity that $R_i = \widetilde{R}_i \Rightarrow R_i = R_{i+1}$ and the iterative process terminates.

Based on properties a) and b), it follows the proof for this lemma. □

After each iteration, REBOM compares the ranking of vectors in the top-k ranking with the ranking of vectors in the top-k partial ranking. If the two rankings are equal, the recovery process terminates. As long as the two rankings are different or one of the two rankings is undefined, the most correlated time series can be used to further improve the accuracy of the recovery.

Example 8. Consider $\mathbf{M}_1 = [M_{1_1}, M_{1_2}, M_{1_3}, M_{1_4}, M_{1_5}]$ and $k = 3$. After each iteration we create matrix $\mathbf{N}_i$ with recovered values and compare $Ranking(\mathbf{N}^{top\text{-}3})$ with $Ranking(\widetilde{\mathbf{N}}^{top\text{-}3})$. Initially, $\widetilde{\rho}(M_1, M_i)$ and $Ranking(\widetilde{\mathbf{M}}^{top\text{-}3})$ are undefined and thus, REBOM iterates. REBOM terminates after two steps since $Ranking(\mathbf{N}_2^{top\text{-}3}) = Ranking(\widetilde{\mathbf{N}}_2^{top\text{-}3}) = \{M_1, M_2, M_3\}$. The vectors of the top-k ranking and top-k partial ranking are highlighted in gray and the recovered values are displayed in bold.

	$\mathbf{M}_1$				
	M_{1_1}	M_{1_2}	M_{1_3}	M_{1_4}	M_{1_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-1	0.2	0	0	0.7
	-1	0.4	-0.25	0	0.4
	-1	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(M_{1_1}, M_{1_i})$	1	-0.69	-0.33	-0.43	-0.46
$\widetilde{\rho}(M_{1_1}, M_{1_i})$	-	-	-	-	-

	$\mathbf{N}_1$				
	N_{1_1}	N_{1_2}	N_{1_3}	N_{1_4}	N_{1_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-0.5	0.2	0	0	0.7
	-0.8	0.4	-0.25	0	0.4
	-0.5	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(N_{1_1}, N_{1_i})$	1	-0.78	-0.45	-0.47	-0.41
$\widetilde{\rho}(N_{1_1}, N_{1_i})$	1	-1	1	0.5	0.97

	$\mathbf{N}_2$				
	N_{2_1}	N_{2_2}	N_{2_3}	N_{2_4}	N_{2_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-0.2	0.2	0	0	0.7
	-0.8	0.4	-0.25	0	0.4
	-0.2	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(N_{2_1}, N_{2_i})$	1	-0.8	-0.48	-0.46	-0.36
$\widetilde{\rho}(N_{2_1}, N_{2_i})$	1	-1	1	0.5	0.97

Figure 2.8: Iterative Recovery of REBOM

2.4.4 Algorithm

Algorithm 1 implements the block recovery of REBOM. First, using the method described in subsection 2.3.2, $\mathbf{X}^1$ is created by initializing the missing values of $\mathbf{X}^0$. Then, the vectors representing each time series of $\mathbf{X}^1$ are inserted as columns in the matrix of vectors $\mathbf{N}_1$.

Algorithm 1: REBOM's Block Recovery

```
Input: A set of $n$ time series $\mathbf{X}^0 = \{X_1^0, X_2^0, \ldots, X_n^0\}$
Output: A set of recovered time series $\widetilde{\mathbf{X}} = \{\widetilde{X}_1^{j_1}, \widetilde{X}_2^{j_2}, \ldots, \widetilde{X}_n^{j_n}\}$
begin
    $\mathbf{X}^1 = Init(\mathbf{X}^0)$;
    for each $X_i^1 \in \mathbf{X}^1$ do
        $X_i^1 = Extract\_val(X_i^1)$;
        $j = 1$;
        $\mathbf{N}_j = [V_i^1]$;
        for each $\widetilde{X}_p^1 \in \widetilde{\mathbf{X}}^1 \setminus \widetilde{X}_i^1$ do
            $\mathbf{V}_p^1 = Extract\_val(\widetilde{X}_p^1)$;
            $\mathbf{N}_j = [\mathbf{N}_j, V_p^1]$;
        while $Ranking(\mathbf{N}_j^{top\text{-}k}) <> Ranking(\widetilde{\mathbf{N}}_j^{top\text{-}k})$ or $Ranking(\mathbf{N}_j^{top\text{-}k})$=NAN or $Ranking(\widetilde{\mathbf{N}}_j^{top\text{-}k}) = NAN$ do
            $\mathbf{U\Sigma V}^T = SVD(\mathbf{N}_j^{top\text{-}k})$;
            $\mathbf{\Sigma}_r = Reduce\_Dim(\mathbf{\Sigma}, n, r)$;
            $\mathbf{P} = \mathbf{U} \times \mathbf{\Sigma}_r \times \mathbf{V}^T$;
            $\mathbf{N}_j = UMV(\mathbf{N}_j^{top\text{-}k}, \mathbf{P})$;
            $j+ = 1$;
        $\widetilde{X}_i^j = Add\_ts(N_{j_i})$;
        $\widetilde{\mathbf{X}}^j = \{\widetilde{\mathbf{X}}^j\} \cup \{\widetilde{X}_i^j\}$;
        $i+ = 1$;
    return $\widetilde{\mathbf{X}}^j$;
```

The vector to recover is inserted as the first column of $\mathbf{N}_1$. The order of the selected vector to recover has no impact on the result of the recovery since only the original vectors are used in the recovery process. Therefore, the proposed recovery is deterministic and does not depend on the order of time series to recover. Next, if the ranking of the top-k ranking matrix is different from the ranking of the top-k partial matrix or one of the rankings is undefined (NAN), the recovery is performed. If $Ranking(\mathbf{N}_j^{top\text{-}k})$ is equal to $Ranking(\widetilde{\mathbf{N}}_j^{top\text{-}k})$ the recovered time series is inserted into the set of recovered time series, i.e, $\widetilde{\mathbf{X}}^j$. Once all time series have been recovered, $\widetilde{\mathbf{X}}^j$ will be returned as the result of REBOM's block recovery.

$Extract\_val()$ and $Add\_ts()$ are used respectively to extract values from a time series and to add time stamps to a vector.

The UMV algorithm (cf. Algorithm 2) updates missing values. It accesses the database and uses procedural SQL to determine the indexes of missing values (load_mv_indexes()). The code of this function is described in the appendix.

Algorithm 2: Updating Initialized Missing Values

```
1 begin
2     for each $V_i \in \mathbf{V}_1$ do
3         $T_i^0$=load_mv_indexes($i$);
4         for each $v_{i_j} \in V_i$ do
5             if $position(v_{i_j}) \in T_i^0$ then
6                 Insert_element($\mathbf{V}_3, v'_{ij}$);
                  // Insert $v'_{ij} \in \mathbf{V}_2$ in row $i$ and column $j$ of $\mathbf{V}_3$
7             else
8                 Insert_element($\mathbf{V}_3, v_{i_j}$);
9     return $\mathbf{V}_3$;
```

2.5 Experiments

2.5.1 Experimental Setup

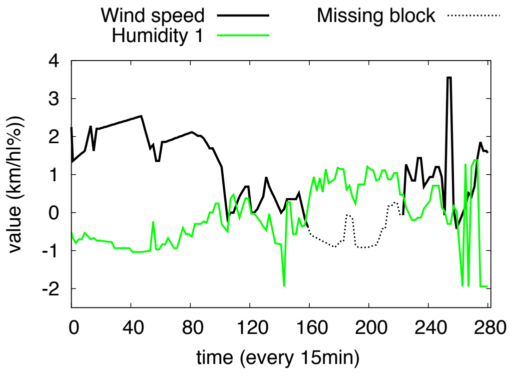
For the evaluation we use real world datasets and synthetic data sets that describe hydrological phenomena of up to 15 million observations produced by sensors in 242 mountain stations. Our hydrological database contains 79 temperature time series, 69 precipitation time series, 48 water level time series, 15 humidity time series, 4 wind speed time series and 3 air pressure time series. The data was provided by an environmental engineering company [Hyd12].

We ran experiments to compare the recovery accuracy of REBOM against state-of-the-art techniques.

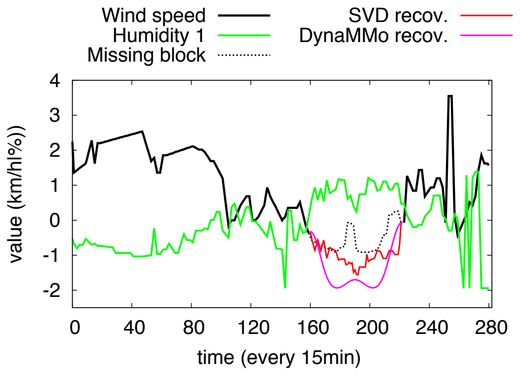
2.5.2 Experiments with Hydrological Time Series

Restoration of Peaks and Valleys

In the first set of experiments, we compare the accuracy of REBOM for the restoration of missing blocks against a parameter-free recovery technique that is the (non-iterated) low rank SVD and a parametric recovery technique that is DynaMMo [LMPF09]. These two techniques are the most accurate techniques for the recovery of blocks of missing values in time series. We ran our experiment on a wind speed time series measured in the area of Adige a Lasa of the region of South Tyrol during year 2001 and a humidity time series measured in the area of Col dei Baldi in the same region during the same time period. Figure 2.9(a) graphically displays the two time series. We drop a block of values for $t \in]160, 220[$ and restore it using the low rank SVD and DynaMMo. The dropped block includes a valley with a small peak.



(a) Time Series Measured in Two Different Areas



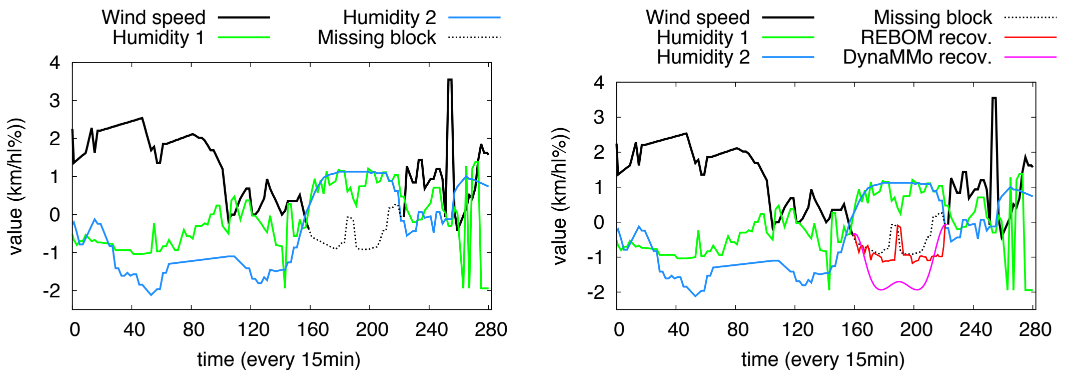
(b) Recovery of a Removed Block

Figure 2.9: Recovery using Low Rank SVD and DynaMMo

The recovery of the two techniques is shown in Figure 2.9(b). The low rank SVD is only able to detect part of the trend of the missing block, i.e., only a valley is recovered. The shape of the recovered valley resembles the shape of the block that belongs to the same time interval of the missing block in the other time series. DynaMMo is able to detect the entire trend of the missing block, i.e., a valley containing a small peak. However the shape of the original block is not accurately restored. The recovered block looks similar to a smooth spline that contains a small peak. Since we use only two time series REBOM will not iterate. Therefore, the recovery of REBOM is similar to the recovery of the low rank SVD.

We add a second humidity time series to the experiment to compare the block recovery of RE-

BOM against DynaMMo (see Figure 2.10). The second humidity time series, denoted as Humidity 2, is measured in the area of Monte Piana in the region of South Tyrol during year 2001. The result of Figure 2.10(b) shows that the recovery of DynaMMo does not change by the addition of a third time series because DynaMMo cannot use more than one reference time series in the recovery process. REBOM exploits the two humidity time series in the recovery process. It uses the history of the wind speed time series together with the correlation with respect to the two humidity time series to recover the missing block. Both the trend and the shape of the missing block are accurately recovered. Adding more correlated time series will further improve the block recovery of REBOM (see Figure 2.12).



(a) Time Series Measured in Three Different Areas (b) Recovery of a Removed Block

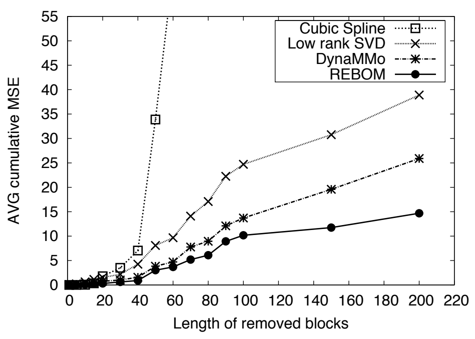
Figure 2.10: Recovery Using REBOM and DynaMMo

We run a second set of experiments in which we compare the block recovery error using the Mean Squared Error (MSE):

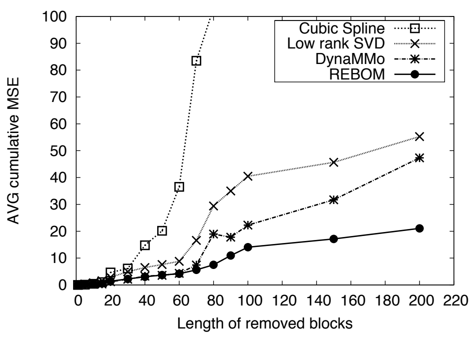
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(w_i - {v_i}^{+})^2$$

where w is the recovered value, v^{+} is the original value and n is the number of deleted observations.

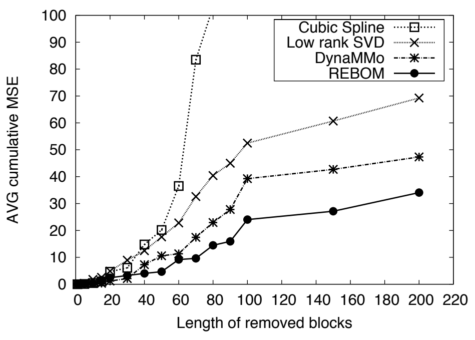
Figure 2.11 shows the cumulative recovery error for removed blocks of values of increasing length: we set a starting timestamp, we vary the length of the removed block and we compute the cumulative recovery MSE of each block. The x-axis represents the length (number of values) of the removed block to recover and the y-axis represents the average cumulative MSE. The experiments in Figures 2.11(a) and 2.11(b) are executed respectively on six different temperature time series with 1000 values each measured in region of Alto Adige and four different humidity time series with 1000 values each measured in the region of Vipetino. For these two experiments, we remove a block from one time series only while the other time series are complete. The results in both experiments show that REBOM outperforms the low rank SVD and DynaMMo for the recovery of successive blocks of missing values and cubic spline is off the scale. For blocks of up to 100 removed values, the recovery error of REBOM slightly increases with the number of removed values. For blocks of more than 100 removed values, the error becomes almost stable and is not anymore affected by the number of removed values. In contrast, the recovery error



(a) Average Cumulative Error for Successive Removed Blocks in One Temperature Time Series. The correlated temperature time series used in the recovery are complete.



(b) Average Cumulative Error for Successive Removed Blocks in One Humidity Time Series. The correlated humidity time series used in the recovery are complete.



(c) Average Cumulative Error for Successive Removed Blocks in One Humidity Time Series. The correlated humidity time series used in the recovery contain missing values.

Figure 2.11: Recovery of Blocks of Different Lengths

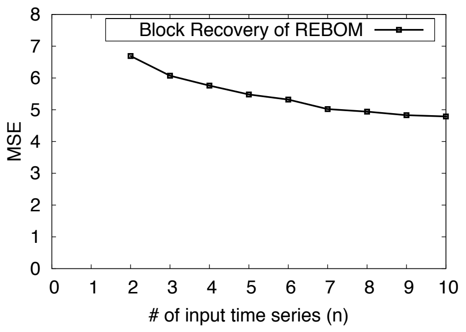
of DynaMMo and the low rank SVD increases with the length of removed blocks. The small cumulative recovery error of REBOM is due to the use of different correlated time series at every iteration of the algorithm. The experiment of Figure 2.11(c) is executed on four humidity time series of 1000 values each. The first time series is complete, the second time series contains a missing block in the time range $[0, 100]$, the third time series contains a missing block in the time range $[100, 200]$ and the fourth time series contains a missing block in the time range $[200, 300]$. We execute the same process performed in the experiment of Figure 2.11(b) for the complete correlated humidity time series. The experiment shows that, compared to the result of Figure 2.11(b), the recovery accuracy of REBOM, DynaMMo and low rank SVD gets worse when using multiple time series with missing values. The recovery accuracy of REBOM is still better than the one of the other techniques.

In the experiment of Figure 2.12, we use different correlation values and number of input time series (n) to evaluate the impact on the recovery MSE. We vary n and we compute the MSE of REBOM for the same block containing 90 missing values. Figure 2.12(a) shows that in the case of time series of high correlation ($1 \geq |\rho| > 0.7$), the MSE of REBOM decreases only slightly as n grows. REBOM is able to restore the missing block using a small number of highly correlated input time series. This result is explained by the fact that, for highly correlated time series, the starting top-k ranking matrix is similar to the partial ranking matrix. Therefore, the recovery of REBOM converges quickly. Figure 2.12(b) shows that, using more time series of moderate correlation ($0.7 \geq |\rho| > 0.4$), the MSE of REBOM decreases linearly. REBOM uses all the time series to perform the most accurate recovery. Figure 2.12(c) illustrates that, the MSE increases for input time series with low correlated time series ($0.4 \geq |\rho| > 0$).

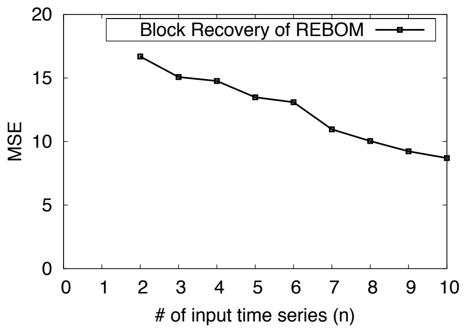
In the experiment of Figure 2.13 we set n to 10 and we vary the number of time series in the top-k ranking matrix. In Figure 2.13(a) the minimum MSE is reached for $k \in [2, 4]$. In Figures 2.13(b) and 2.13(c), the minimum recovery MSE is reached for a single value that is respectively $k = 4$ and $k = 2$. Again, the recovery accuracy of REBOM decreases for time series with low correlation, i.e., $0.4 \geq |\rho| > 0$, in the top-k ranking matrix.

Invariance to Initialization Method

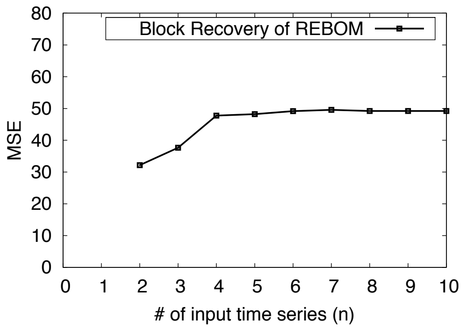
We run this experiment to test the impact of the initialization method on the block recovery of REBOM. Figure 2.14 shows that with different initialization techniques, REBOM needs more iterations to reach the minimum recovery error. Compared to our initialization method, a linear spline initialization needs twice the number of iterations to reach the minimum recovery error. Using a k Nearest Neighbor initialization, REBOM needs 2.5 times more iterations than our initialization technique to reach the same recovery error. Thus, the accuracy of REBOM is independent from the initialization method. However, our initialization initialization method provides a faster recovery of blocks of missing values.



(a) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $1 \geq |\rho| > 0.7$



(b) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $0.7 \geq |\rho| > 0.4$

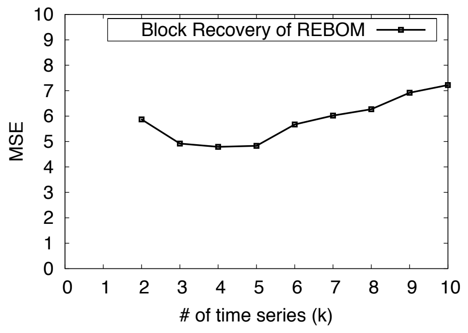


(c) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $0.4 \geq |\rho| > 0$

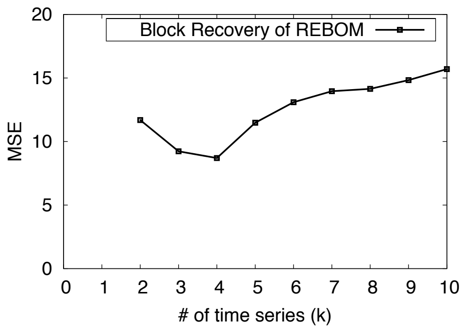
Figure 2.12: Impact of n in the Recovery MSE of REBOM

Running Time Performance

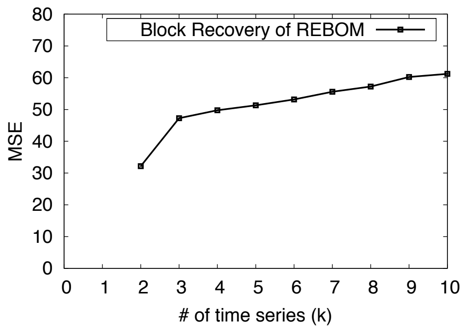
The REBOM implementation uses the Golub/Kahan decomposition algorithm [Kon05] and has a run time complexity of $\mathcal{O}(\#iterations \times (4n^2k + 8nk^2 + 9k^3))$, where n is the length of the longest time series and k is the number of vectors of $\mathrm{M}^{top\text{-}k}$. The complexity of building $\mathrm{M}^{top\text{-}k}$ is the cost of computing k times ρ between two time series and that is $\mathcal{O}(kn^2)$. Therefore, the total complexity of using REBOM is $\mathcal{O}(\#iterations \times (5n^2k + 8nk^2 + 9k^3))$. Table 2.1 compares the average running time of REBOM against DynaMMo that has a complexity of $\mathcal{O}(\#iterations \times (kn^3))$. 3000 different time series were created by extracting 1000 observations from 15 different temperature time series. We drop from one time series a block of 200 values and we recover it back using REBOM (containing 1000, 2000 and 3000 time series respectively) and DynaMMo. We set the value of k to four, since we reached the optimal recovery accuracy with this value. The result of this experiment shows that up to 1500 time series, REBOM is faster



(a) Impact of k in the Recovery MSE. Each time series has a correlation value: $1 \geq |\rho| > 0.7$



(b) Impact of k in the Recovery MSE. Each time series has a correlation value: $0.7 \geq |\rho| > 0.4$



(c) Impact of k in the Recovery MSE. Each time series has a correlation value: $0.4 \geq |\rho| > 0$

Figure 2.13: Recovery MSE using Different Number of Time Series in the Top-k Ranking Matrix

than DynaMMo. With a higher number of input time series, the performance of REBOM starts to be slower than DynaMMo.

Table 2.1: Average Running Time Comparison (sec).

$REBOM^{3000}$	$REBOM^{2000}$	DynaMMo	$REBOM^{1000}$
158	49	32	8

Recovery Using Linear Time Series

In the experiment of Figure 2.15, we show the impact of using extremely irregular time series. We take as input a humidity time series measured in spring 2001 from which we drop a block for $ts \in]120, 160[$, a constant time series, and a monotonic time series. The result of Figure 2.15(a)

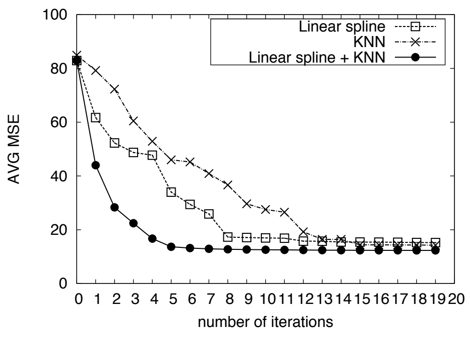
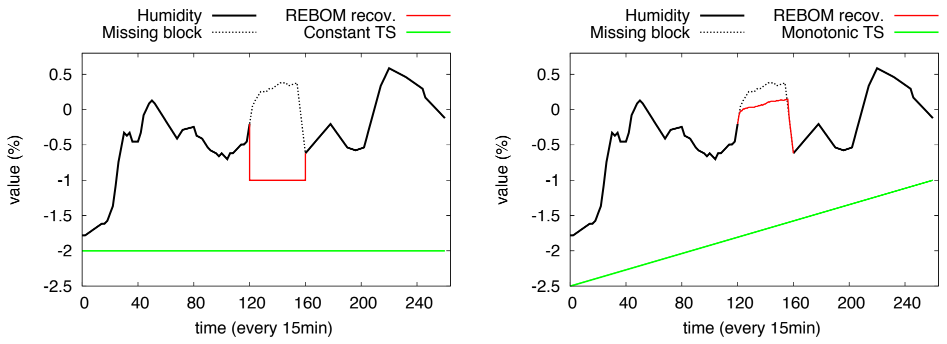


Figure 2.14: Number of Iterations using Different Initialization Techniques

shows that, since the correlation between the humidity time series and the constant time series is undefined (all values are equal), REBOM performs a bad recovery. In Figure 2.15(b), the humidity time series and the monotonic time series are correlated. Therefore, both time series are used to recover the type of the missing block. The recovered block has an increasing monotonic shape that looks similar to the monotonic time series. In the experiment of Figure 2.15(b), REBOM produces a good recovery of the type and the shape of the missing block. The application of DynaMMo in the experiment would set all the recovered values to 0.



(a) Recovery of REBOM using lines of function $v = c$, where c is a constant. The result of the recovery is the same for any given value of c

(b) Recovery of REBOM using lines of function $v = at + b$, where $a = 0.5$ and $b = -2.5$

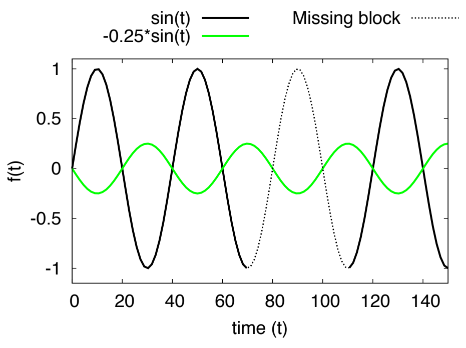
Figure 2.15: Impact of Extremely Irregular Time Series in the Recovery of REBOM

2.5.3 Experiments with Synthetic Regular Time Series

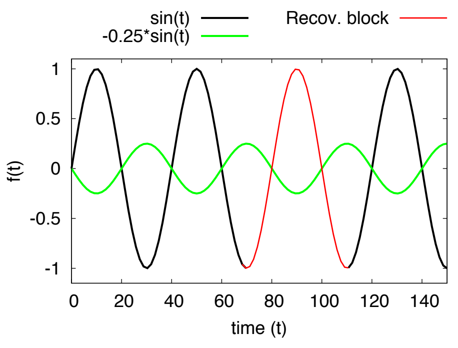
This subsection describes a set of experiments conducted with synthetic data. We compare the block recovery of REBOM against DynaMMo.

Different Amplitudes

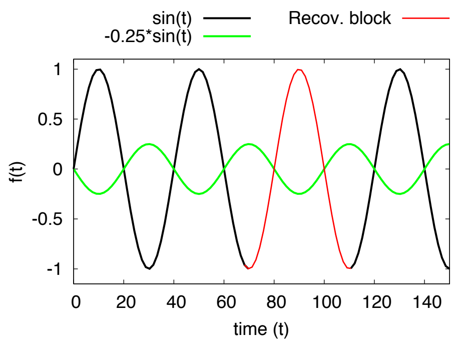
Figure 2.16 compares the recovery of the two techniques for two regular time series having different amplitudes. The first time series is a $sin(t)$ wave and the second time series is a sine wave multiplied by a negative scaling factor, i.e., $-0.25 * sin(t)$. For $t \in]70, 110[$, we drop a block from $sin(t)$ and we recover it using REBOM and DynaMMo. Both techniques are able to accurately recover the missing block. REBOM uses the correlation between the two time series in order to determine the shape of the missing block, i.e, a peak. The amplitude of the missing peak is determined using the amplitude of the existing peaks from $sin(t)$. The two techniques perform an accurate recovery in the case of multiplying the second wave by any scaling factor.



(a) Original Waves



(b) Recovery of REBOM

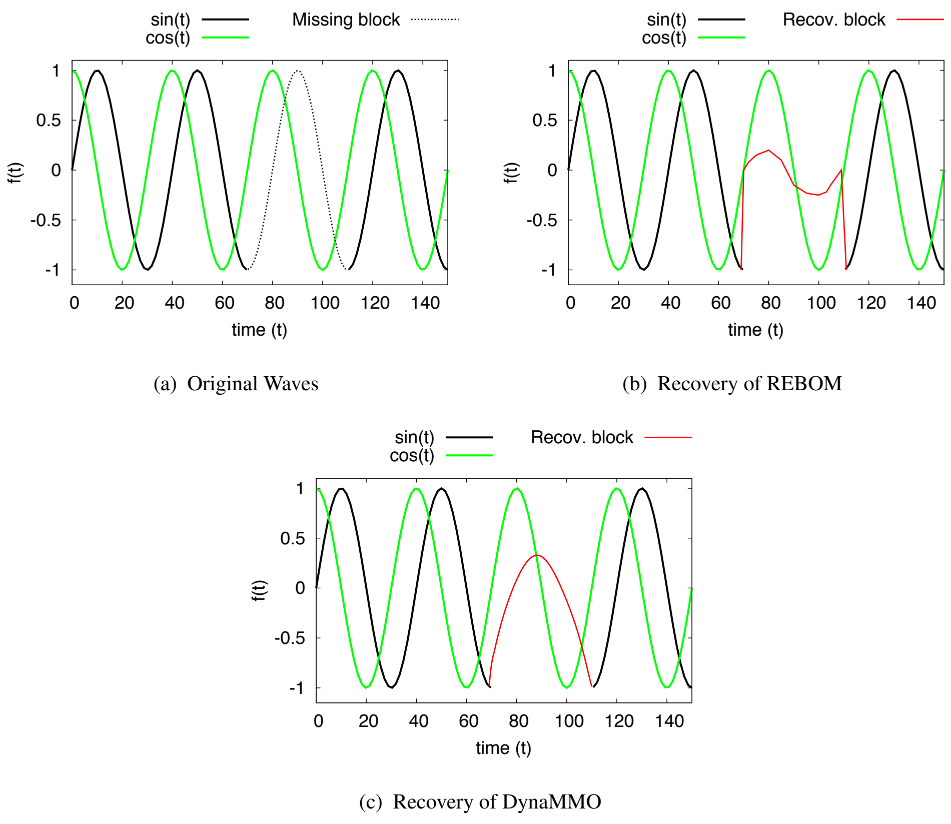


(c) Recovery of DynaMMO

Figure 2.16: Recovery of DynaMMO and REBOM for Time Series of Different Amplitudes

Shifted Peaks

Figure 2.17 shows two regular time series shifted in time, i.e., $sin(t)$ and $cos(t)$. For $t \in]70, 110[$, we drop a block from $sin(t)$ and we recover it using REBOM and DynaMMo. REBOM is applied without initial alignment of the two time series. As expected, DynaMMo outperforms REBOM in recovering the missing block. DynaMMo is able to compute the periodicity model and performs a good block recovery. However, REBOM recovers a block that is only influenced by the shape of the block in $cos(t)$ for $t \in]70, 110[$, i.e., a peak followed by a valley. For shifted time series, REBOM is not able to use the history of $sin(t)$ in the recovery process. The decomposition performed by our technique is sensitive to the row position of values inside the $\mathrm{M}^{top\text{-}k}$ matrix. In order to overcome this problem, an initial alignment between the two time series must be performed in a preprocessing step (cf. Subsection 2.3.2).



(a) Original Waves (b) Recovery of REBOM (c) Recovery of DynaMMO

Figure 2.17: Recovery of DynaMMO and REBOM for Shifted Time Series

2.6 Conclusion

This chapter studies the recovery of blocks of missing values in irregular time series. We develop an iterative greedy algorithm called REBOM, that uses at every iteration the most correlated time series to the time series that contains the missing blocks to reconstruct missing peaks and valleys. Empirical studies on real hydrological data sets demonstrate that our algorithm has the most accurate block recovery among existing techniques. In future work, it is of interest to examine the impact of using the recovered time series in the recovery process instead of the original ones. It is also foreseen to investigate the impact of the global correlation on the recovery accuracy together with the local correlation. Another promising direction, is to progress the interaction with the database and develop an SQL based recovery solution that reduces the number of I/O's.

CHAPTER 3

Using Lowly Correlated Time Series to Recover Missing Values

Abstract

The Singular Value Decomposition (SVD) is a matrix decomposition technique that has been successfully applied for the recovery of blocks of missing values in time series. In order to perform an accurate block recovery, SVD requires the use of highly correlated time series. However, using lowly correlated time series that exhibit shape and/or trend similarities could increase the recovery accuracy. Thus, the latter time series could also be exploited by including them in the recovery process.

In this chapter, we compare the accuracy of the Centroid Decomposition (CD) against SVD for the recovery of blocks of missing values using highly and lowly correlated time series. We show that the CD technique better exploits the trend and shape similarity to lowly correlated time series and yields a better recovery accuracy. We run experiments on real world hydrological and synthetic time series to validate our results.

3.1 Introduction

In real world applications sensors are used to measure time series data of different types, which are then collected, processed and stored in central stations. In the hydrological field, for instance, weather stations collect measurements that describe meteorological phenomena, e.g., temperature, humidity, air pressure, precipitation, etc. These time series contain blocks of missing values due to many reasons, e.g., sensor failure, power outage, sensor to central server transmission problem, etc. In order to recover these missing values, existing recovery techniques use the (base) time series that contains the missing values in addition to highly correlated (reference) time series. However, these recovery techniques can not learn from the trend and shape similarity of lowly correlated reference time series which are consequently not considered in the recovery process.

The Foehn, for instance, is a warm wind that reaches weather stations at different time points. This environmental phenomenon yields time series with shape and trend similarities, but shifted in time. For example, the Foehn yields shifted temperature time series with similar shapes, i.e., peaks that contain similar spikes. These shifted time series are lowly correlated. It is of interest to benefit from Foehn based time series and include them, in addition to the highly correlated time series, in the recovery process. In this chapter, we consider the category of lowly correlated reference time series, e.g., Fohen based time series, that exhibit shape and/or trend similarities to the base time series.

Matrix decomposition techniques decompose an input matrix into the product of k matrices where $k \in [2, 3]$. The truncated Singular Value Decomposition (SVD) has been successfully applied to recover missing values in time series [KB12]. The truncated SVD performs a decorrelation of vectors and subsequently an unweighted relative reduction of the Mean Squared Error (MSE) to the reference time series. The unweighted MSE reduction yields a recovery that ignores the correlation difference between the input time series. Thus, this recovery technique is not suitable to apply in case of using highly and lowly correlated reference time series (cf. Section 3.5). To the best of our knowledge, there does not exist any technique that introduces different weights in the decomposition process of SVD. In [LBKL15, HMT11, AM07] fast approximations of the truncated SVD have been proposed. Similarly to SVD, the latter approximations perform a decorrelation of vectors and thus, produce an unweighted MSE relative reduction.

In this work, we are interested in the case of using highly and lowly correlated time series for the recovery of blocks of missing values. Intuitively, in such cases, an accurate recovery technique should give different weights to the used time series. In contrast to the truncated SVD, the truncated Centroid Decomposition (CD) technique gives a weight proportional to the correlation between the base and the reference time series (cf. Section 3.5). Consequently, the obtained recovery produces a relative reduction of the MSE to the highly correlated reference time series more than to the lowly correlated one yielding a block recovery better than the one produced by the truncated SVD. We assume that the lowly correlated time series that exhibit trend and/or shape similarity are given as input. Searching for these time series is beyond the scope of this chapter.

The main contributions of this chapter are:

- We prove that CD technique produces correlated output vectors while SVD technique produces uncorrelated output vectors.
- We empirically show that CD performs a weighted MSE relative reduction that is proportional to the correlation of the input time series. The resulting recovery of missing values uses the correlation difference between the input time series.
- We empirically show that SVD performs an unweighted MSE relative reduction. The resulting recovery of missing values ignores the correlation difference between the input time series.
- We present the results of an experimental evaluation of the recovery accuracy of the CD and SVD techniques. The iterated truncated CD produces a better recovery accuracy in case of using a similar number of highly and lowly correlated time series.

The rest of the chapter is organized as follows. Section 3.2 discusses related work. Section 3.3 describes the recovery process using SVD and CD techniques. Section 3.4 defines the unweighted recovery and the correlation based recovery respectively performed by SVD and CD techniques. Section 3.5 reports the evaluation results. Section 3.6 concludes the chapter and points to future work.

3.2 Related Work

The Singular Value Decomposition (SVD) is a commonly used matrix decomposition technique. It computes the singular values with their corresponding right and left singular vectors. The truncated SVD, which is computed out of SVD by nullifying the smallest singular values, has been extensively used in many fields, e.g., compression, noise reduction, etc. Khayati et al. [KB12] applied the truncated SVD for the recovery of missing values in time series. The basic idea is as follows: the truncated SVD is iteratively applied to a matrix that has as columns the time series for which the missing values have been initialized through linear interpolation. The iterative process refines only the initialized missing values and terminates when the difference between the updated values before and after the refinement is smaller than a small threshold value, e.g., 10^{-5}. The Mean Squared Error (MSE), between the real values and the recovered ones, is used to evaluate the recovery accuracy [LMPF09].

The Centroid Decomposition (CD) is a matrix decomposition technique that decomposes an input matrix into the product of two matrices. Chu et al. [CF01] introduce an algorithm that computes the CD of an input matrix in quadratic run time, but requires the construction of a correlation square matrix that has a quadratic space complexity. Khayati et al. [KBG14] propose an algorithm to compute the CD out of the input matrix using a weight vector instead of the construction of the correlation matrix. They prove the correctness of the proposed solution. The space complexity is thus reduced from quadratic to linear while keeping the same run time complexity.

The Semi Discrete Decomposition (SDD) [KO98] is a matrix decomposition technique that decomposes an input matrix into three matrices such that their product approximates the input matrix, i.e., $\mathbf{X} \approx \mathbf{X}' \cdot \mathbf{D} \cdot \mathbf{Y}^T$. The resulting $\mathbf{D}$ is a diagonal matrix and the values of $\mathbf{X}'$ and $\mathbf{Y}$ are restricted to belong to the set $\{-1, 0, 1\}$. The truncated SDD has been used as clustering method [KO00]. The non-zero elements of the matrix obtained from the product $d_{ii} \times X'_{*i} \cdot Y^T_{*i}$ are the elements of the input matrix $\mathbf{X}$ which have the closest values and thus can be clustered together. Due to the set restriction of the elements of $\mathbf{X}'$ and $\mathbf{Y}$, the application of SDD for the recovery of blocks of missing values does not produce accurate results.

In addition to matrix decomposition techniques, matrix factorization techniques have been also applied for the recovery of missing values. The latter techniques start from k random matrices in order to approximate the input matrix. Stochastic Gradient Descent (SGD) [YHSD12] is a matrix factorization technique that approximates an input matrix $\mathbf{X}$ by the product of two matrices $\mathbf{P}$ and $\mathbf{Q}$, i.e., $\mathbf{X} \approx \mathbf{P} \cdot \mathbf{Q}$. SGD iteratively minimizes an error function by computing the gradient. At each iteration, the gradient is computed using random sample square blocks of the input matrix. The accuracy of the gradient increases with the size and the number of the used blocks [GNHS11]. Thus, using an input matrix with high number of rows and columns yields an accurate gradient's computation and subsequently a good approximation of the input matrix. In [KBV09], SGD has been successfully applied to predict ratings in recommender systems for a matrix of items as rows and users as columns. Balzano et al [BNR10] propose an SGD-based solution, called GROUSE, for the recovery of blocks of missing values in an input matrix. GROUSE performs an accurate recovery for matrices of a high number of rows and columns. The recovery accuracy of the proposed solution deteriorates if the number of columns is much smaller than the number of rows such as in the hydrology field where the number of time series is much smaller than the number of observations.

3.3 Preliminaries

3.3.1 Notation

Bold upper-case letters refer to matrices, regular font upper-case letters to vectors (rows and columns of matrices) and lower-case letters to elements of vectors/matrices. For example, $\mathbf{X}$ is a matrix, $\mathbf{X}^T$ is the transpose of $\mathbf{X}$, X_{i*} is the i-th row of $\mathbf{X}$, X_{*i} is the i-th column of $\mathbf{X}$ and x_{ij} is the j-th element of X_{i*}.

In multiplication operations we use the sign $\times$ for scalar multiplication and the sign $\cdot$ otherwise. The symbol $\|\|$ refers to the l-2 norm of a vector. Assume $X = [x_1, \dots, x_n]$, then $\|X\| = \sqrt{\sum_i^n (x_i)^2}$.

3.3.2 Background

Time Series

A time series $X_{i*} = \{(t_1, v_1), (t_2, v_2), \dots, (t_n, v_n)\}$ is a set of n temporal values v_i ordered with respect to their timestamps t_i. We consider time series that have the same granularity of values. Thus, we omit the timestamps and we write time series using only their ordered values, e.g., time series $X_{1*} = \{(1, 4), (2, 5), (3, 1)\}$ is written as $X_{1*} = \{4, 5, 1\}$. Time series are inserted as columns of the input matrix $\mathbf{X}$.

Pearson Correlation Coefficient

The definition of the Pearson correlation coefficient $\rho(X, Y)$ between two vectors X and Y of equal length n is introduced in Section 2.4.1. The absolute value of ρ ranges between 0 and 1 where $\rho \in [0.7, 1]$ stands for highly correlated vectors. The value of ρ is undefined if all x_i (and/or y_i) are equal.

Initialization Strategy

The missing values of each time series are initialized as a preprocessing step before the application of the recovery process. A missing value is initialized with a linear interpolation between the predecessor and the successor values. If the missing value occurs as the first or the last elements of the time series, we use the nearest neighbor initialization (cf. Section 2.3.2).

3.3.3 Matrix Decomposition

Singular Value Decomposition

The *Singular Value Decomposition (SVD)* is a matrix decomposition technique that decomposes an $n \times m$ matrix, $\mathbf{X} = [X_{*1}| \ldots |X_{*m}]$, into an $n \times p$ matrix, $\mathbf{U}$, a $p \times m$ matrix, $\mathbf{\Sigma}$, and an $m \times m$ matrix $\mathbf{V}$, i.e.,

$$\begin{aligned}\mathbf{X} &= \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T \\ &= \sum_{i=1}^{p} \sigma_i \times U_{*i} \cdot (V_{*i})^T,\end{aligned}$$

where $p = \min(n, m)$, the columns of $\mathbf{U}$ and $\mathbf{V}$ are respectively called left and right singular vectors, and $\mathbf{\Sigma}$ is a matrix whose diagonal elements, σ_i, are called singular values and are arranged in decreasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$. The obtained columns of $\mathbf{U}$ are orthogonal to each other, i.e., $U_{*1} \perp U_{*2} \perp \ldots \perp U_{*p}$. Similarly, the columns of $\mathbf{V}^T$ are orthogonal to each other. In order to guarantee the orthogonality of columns of respectively $\mathbf{U}$ and $\mathbf{V}$, SVD requires the use of the same input matrix [GVL96]. Since SVD decomposition is performed based on the same input matrix $\mathbf{X}$, we refer to SVD as a *flat* decomposition method.

Figure 3.1 illustrates the SVD decomposition of an input matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \text{-4} & 0 \\ 2 & 1 \\ 3 & \text{-2} \end{bmatrix}; \text{ SVD}(\mathbf{X}) = \underbrace{\begin{bmatrix} \text{-0.725} & \text{-0.307} \\ 0.333 & 0.627 \\ 0.603 & \text{-0.716} \end{bmatrix}}_{\mathbf{U}}, \underbrace{\begin{bmatrix} 5.445 & 0 \\ 0 & 2.086 \end{bmatrix}}_{\mathbf{\Sigma}}, \underbrace{\begin{bmatrix} 0.987 & 0.16 \\ \text{-0.16} & 0.987 \end{bmatrix}}_{\mathbf{V}}$$

such that

$$\mathbf{X} = \underbrace{\begin{bmatrix} \text{-0.725} & \text{-0.307} \\ 0.333 & 0.627 \\ 0.603 & \text{-0.716} \end{bmatrix}}_{\mathbf{U}} \times \underbrace{\begin{bmatrix} 5.445 & 0 \\ 0 & 2.086 \end{bmatrix}}_{\mathbf{\Sigma}} \times \underbrace{\begin{bmatrix} 0.987 & \text{-0.16} \\ 0.16 & 0.987 \end{bmatrix}}_{\mathbf{V}^T}$$

Figure 3.1: Example of Singular Value Decomposition.

Centroid Decomposition

The *Centroid Decomposition (CD)* is a matrix decomposition technique that decomposes an $n \times m$ matrix, $\mathbf{X} = [X_{*1}| \ldots |X_{*m}]$, into an $n \times m$ loading matrix, $\mathbf{L}$, and an $m \times m$ relevance matrix, $\mathbf{R}$, i.e.,

$$\mathbf{X} = \mathbf{L} \cdot \mathbf{R}^T = \sum_{i=1}^{m} L_{*i} \cdot (R_{*i})^T,$$

where $\|L_{*1}\| > \|L_{*2}\| > \ldots > \|L_{*m}\| \geq 0$. Figure 4.2 illustrates the CD of matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \text{-4} & 0 \\ 2 & 1 \\ 3 & \text{-2} \end{bmatrix}; \text{ CD}(\mathbf{X}) = \underbrace{\begin{bmatrix} \text{-3.977} & -0.43 \\ 1.878 & 1.214 \\ 3.202 & \text{-1.658} \end{bmatrix}}_{\mathbf{L}}, \underbrace{\begin{bmatrix} 0.994 & 0.11 \\ \text{-0.11} & 0.994 \end{bmatrix}}_{\mathbf{R}}$$

such that

$$\mathbf{X} = \begin{bmatrix} \text{-4} & 0 \\ 2 & 1 \\ 3 & \text{-2} \end{bmatrix} = \underbrace{\begin{bmatrix} \text{-3.977} & -0.43 \\ 1.878 & 1.214 \\ 3.202 & \text{-1.658} \end{bmatrix}}_{\mathbf{L}} \times \underbrace{\begin{bmatrix} 0.994 & \text{-0.11} \\ 0.11 & 0.994 \end{bmatrix}}_{\mathbf{R}^T}$$

Figure 3.2: Example of Centroid Decomposition.

The CD technique applies an iterative process to compute matrices $\mathbf{L}$ and $\mathbf{R}$. At each iteration i, the input matrix $\mathbf{X}$ is updated by subtracting the product $L_{*i} \cdot R_{*i}^T$ from it. The columns of $\mathbf{L}$

(and $\mathbf{L}$) are not orthogonal to each other. Since CD decomposition is performed by hierarchically updating $\mathbf{X}$, we refer to CD as a *hierarchical* decomposition method.

Chu et al. [CF01] prove that the decomposition performed by CD best approximates the one produced by SVD, i.e., $\mathbf{L}$ approximates the product $\mathbf{U} \cdot \mathbf{\Sigma}$ and $\mathbf{R}$ approximates $\mathbf{V}$.

Truncation

The *truncated SVD* computes a matrix $\mathbf{X}_k$ out of the SVD of $\mathbf{X}$. It takes only the k first columns of $\mathbf{U}$ and $\mathbf{V}$ and the k largest elements of $\mathbf{\Sigma}$ such that $k < p$, i.e.,

$$\mathbf{X}_k = \sum_{i=1}^{k} \sigma_i \times U_{*i} \cdot (V_{*i})^T. \tag{3.1}$$

Eq. (3.1) is equivalent to $\mathbf{X}_k = \mathbf{U} \cdot \mathbf{\Sigma}_k \cdot \mathbf{V}^T$ where $\mathbf{\Sigma}_k$ is obtained by setting the $r - k$ smallest (non zero) singular values of $\mathbf{\Sigma}$ to 0. Let rank p be the maximal number of linearly independent rows or columns of $\mathbf{X}$. Then, among all matrices with rank $k < p$, $\mathbf{X}_k$ is proven to be the optimal approximation to the input matrix $\mathbf{X}$ in the Frobenius norm [Bjö96].

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

3.4 Decomposition Comparison

In this section, we compare the decomposition produced by the truncated SVD against the one produced by the truncated CD using the Mean Squared Error ($MSE = \frac{1}{k}\sum_{i=1}^{k}(\tilde{x}_i - x_i)^2$; initialized value x_i; recovered value $\tilde{x}_i$; number of observations k) between the initialized values and the recovered ones.

3.4.1 Recovery Process

Algorithm 3 describes the pseudo code of function *RecM()* that applies truncated SVD and truncated CD to recover missing values. The algorithm takes an input matrix $\mathbf{X}$ where the missing values have been initialized, and returns a matrix with recovered values $\widetilde{\mathbf{X}}$. Different initialization techniques would lead to the same result but with a higher number of iteration [KBG14]. *RecM()* iteratively replaces the initialized missing values by the result of the truncation of a given matrix decomposition technique. The algorithm terminates if the difference in Frobenius norm ($\|\mathbf{X} - \widetilde{\mathbf{X}}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij} - \tilde{x}_{ij})^2}$; x_{ij}: element of $\mathbf{X}$; $\tilde{x}_{ij}$: element of $\widetilde{\mathbf{X}}$) between the matrix before the update of missing values, $\mathbf{X}$, and the one after, $\widetilde{\mathbf{X}}$, is less than a small threshold value, e.g., $\epsilon = 10^{-5}$.

Algorithm 3: RecM($\mathbf{X}$, n, m, T_j^m)

```
Input: $n \times m$ matrix $\mathbf{X}$; set of missing time stamps $T_j^m$ in $X_{*j}$
Output: $n \times m$ matrix $\widetilde{\mathbf{X}}$ of recovered values
1 repeat
2     $\widetilde{\mathbf{X}} = \mathbf{X}$ ;
      // Apply truncated SVD or truncated CD
3     $\mathbf{X}_k$ =Truncate($\widetilde{\mathbf{X}}$);
      // Update missing values
4     foreach $t \in T_j^m$ do
5         $x_{tj} = w_{tj}$;
          // $w_{tj}$ element of $\mathbf{X}_k$
6 until $\|\mathbf{X} - \widetilde{\mathbf{X}}\|_F < \epsilon$;
7 return $\widetilde{\mathbf{X}}$
```

In what follows we describe the recovery properties using respectively SVD and CD. We assume the case where the correlation ranking between time series does not change over the entire history, i.e., the most correlated reference time series has the highest correlation value to the base time series all over the entire history. In case where the correlation ranking changes over the history, then a segmentation of the time series has to be applied.

3.4.2 SVD recovery

Lemma 3. *Given an input matrix $\mathbf{X}$ of m correlated columns. SVD($\mathbf{X}$) produces non correlated vectors.*

Proof. By definition of SVD, we have that $U_{*1} \perp U_{*2} \perp \ldots \perp U_{*p}$. This implies that the pairwise dot product of columns of $\mathbf{U}$ is equal to 0 and thus, $\forall a, b \in [1, p] \wedge a \neq b$, we get $(U_{*a})^T \cdot U_{*b} = 0$. Using the fact that all input time series have been normalized to have mean equal to 0, we assume u and u' to be the i-th elements of respectively U_{*a} and U_{*b}, and get from Equation (2.1) the following

$$\begin{aligned}\rho(U_{*a}, U_{*b}) &= \frac{\sum_{i=1}^{n}(u_i \times u'_i)}{\sqrt{\sum_{i=1}^{n}(u_i - \bar{u})^2}\sqrt{\sum_{i=1}^{n}(u'_i - \bar{u'})^2}} \\ &= \frac{(U_{*a})^T \cdot U_{*b}}{\sqrt{\sum_{i=1}^{n}(u_i - \bar{u})^2}\sqrt{\sum_{i=1}^{n}(u'_i - \bar{u'})^2}} = 0\end{aligned}$$

As a result, the pairwise correlation between all columns of $\mathbf{U}$ is equal to 0. The previous property holds also for the columns of $\mathbf{V}^T$. □

Definition 5 (Unweighted Recovery)**.** Let $\mathbf{X}$ be an input matrix that contains a base time series B and $k > 2$ reference time series each with a correlation ρ_i to B. An *unweighted recovery* of B produces a similar relative reduction of the MSE between B and the reference time series.

Proposition 1. *Assume an $n \times m$ matrix $\mathbf{X} = [B, R_1, \ldots, R_{m-1}]$. A truncated matrix decomposition of $\mathbf{X}$ that produces uncorrelated vectors performs an unweighted recovery of B.*

Based on Lemma 3 and Proposition 1, we get that the truncated SVD performs an unweighted recovery.

Example 9. Let's take the example of a matrix $\mathbf{X} = [B, R_1, R_2]$ where initialized missing values are marked in bold.

$$\mathbf{X} = \begin{bmatrix} -4 & 1 & 3 \\ \mathbf{-1} & 3 & -1 \\ \mathbf{2} & 6 & 6 \\ 5 & 5 & 3 \end{bmatrix}$$

R_1 is a highly correlated reference time series to B with $\rho(B, R_1)$= 0.88 and R_2 is a lowly correlated reference time series to B with $\rho(B, R_2) = 0.32$. The computation of the MSE before the recovery gives $MSE(B, R_1) = 16$ and $MSE(B, R_2) = 8$.

The following matrix $\widetilde{\mathbf{X}} = [\widetilde{B}, R_1, R_2]$ is an example of an SVD based recovery of B.

$$\widetilde{\mathbf{X}} = \begin{bmatrix} -4 & 1 & 3 \\ 0 & 3 & -1 \\ 4 & 6 & 6 \\ 5 & 5 & 3 \end{bmatrix}$$

The computation of the MSE after the recovery gives $MSE(\widetilde{B}, R_1) = 6.5$ and $MSE(\widetilde{B}, R_2) = 2.5$. The percentage of the MSE relative reduction between B and R_1 is $red(R_1) = \frac{16-6.5}{16} \times 100 = 60\%$. Similarly, the percentage of the MSE relative reduction between B and R_2 is $red(R_2) = 69\%$. As a result, we have $red(R_1) \approx red(R_2)$.

3.4.3 CD recovery

Lemma 4. *Given an input matrix $\mathbf{X}$ of m correlated columns. CD($\mathbf{X}$) produces correlated vectors.*

Proof. This proof follows directly from the proof of Lemma 3. On the contrary of SVD, the columns of $\mathbf{L}$ and $\mathbf{R}^T$ computed by the truncated CD are not orthogonal and thus, the pairwise dot product and consequently the pairwise correlation values are different from 0. □

Definition 6 (Correlation Weighted Recovery)**.** Let $\mathbf{X}$ be an input matrix that contains a base time series B and $k > 2$ reference time series each with a correlation ρ_i to B. A *correlation weighted recovery* of B performs a relative reduction of the MSE between B and the reference time series proportionally to $|\rho_i|$.

Proposition 2. *Assume an $n \times m$ matrix $\mathbf{X} = [B, R_1, \ldots, R_{m-1}]$. A truncated matrix decomposition of $\mathbf{X}$ that produces correlated vectors performs a correlation weighted recovery of B.*

Based on Lemma 4 and Proposition 2 we get that the truncated CD performs a correlation weighted recovery.

Example 10. Let's take the example of a matrix $\mathbf{X} = [B, R_1, R_2]$ used in Example 9. The following matrix $\widetilde{\mathbf{X}} = [\widetilde{B}, R_1, R_2]$ is an example of a CD based recovery of B.

$$\widetilde{\mathbf{X}} = \begin{bmatrix} -4 & 1 & 3 \\ 2 & 3 & -1 \\ 5 & 6 & 6 \\ 5 & 5 & 3 \end{bmatrix}$$

The computation of the MSE after the recovery gives $MSE(\widetilde{B}, R_1) = 1$ and $MSE(\widetilde{B}, R_2) = 5$. The percentage of the MSE relative reduction between B and R_1 is $red(R_1) = 94\%$. The percentage of the MSE relative reduction between B and R_2 is $red(R_2) = 37.5\%$. As a result, we have $red(R_1) \gg red(R_2)$.

3.4.4 Complexity

We compare the runtime and space complexity of CD based recovery against SVD based recovery. We use the algorithm that computes the exact decomposition for each technique.

Run time

Consider an input matrix $\mathbf{X}$ with n rows and m columns. The number of arithmetic operations to compute SVD of $\mathbf{X}$, using Golub and Reinsch algorithm [GVL96], is $4n^2m + 8mn^2 + 9m^3$. The number of arithmetic operations to compute CD of $\mathbf{X}$ is $2pnm$ where p is the number of iterations [KBG14]. At each iteration of CD, the input matrix is subtracted yielding an updated matrix that contains negative elements. Thus, the value of p depends on the distribution of the minus sign across the updated matrix. In practice, the value of p ranges between $\frac{n}{2}$ and $\frac{n}{3}$ (cf. Section 3.5.5).

Space

CD technique requires the storage of nm values of $\mathbf{X}$, nm values of $\mathbf{L}$ and m^2 values of $\mathbf{R}$. No data structure other than the input and the two output matrices is stored. Thus, the total number values stored by CD is equal to $m(2n + m)$ values. SVD requires the storage of nm values of $\mathbf{X}$, nm values of $\mathbf{U}$, m values of Σ and m^2 values of $\mathbf{V}$. Additionally, SVD has to transform $\mathbf{X}$

to a bidiagonal matrix using Householder reduction [GS06] which requires the storage of three additional matrices, i.e., the first matrix contains nm values and the two others contain m^2 values each. The total number values stored by SVD is thus equal to $m(3n + 3m + 1)$ values.

3.5 Experiments

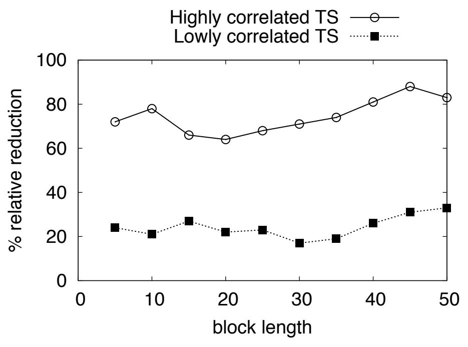
The experiments are performed using real world datasets that describe hydrological time series where each tuple records a timestamp and a value of a specific observation. Hydrological time series with shifted peaks and/or valleys are lowly correlated. Our first set of time series, *HYD*[1], contains 200 time series of six years length each, where measurements are recorded every five minutes. The second set of time series we refer to, *SBR*[2], contains 120 time series of twelve years length each, where measurements are recorded every 30 minutes. The hydrological time series have been normalized with the z-score normalization technique [JNR05]. We consider hydrological time series where the correlation ranking does not change all over the history. We use also synthetic time series, where the correlation is constant all over the entire history. To measure the recovery accuracy, we compute the Mean Squared Error ($MSE = \frac{1}{k}\sum_{i=1}^{k}(\tilde{x}_i - x_i)^2$; original value x_i; initialized value $\tilde{x}_i$; number of observations k) between the original and the recovered blocks.

3.5.1 Recovery using real world TS

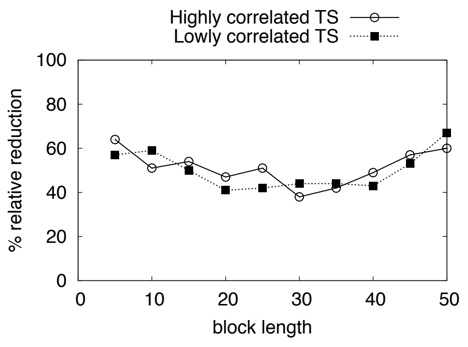
MSE relative reduction

In this experiment we compute the MSE relative reduction between a base time series B and two reference time series. In Figure 3.3 we choose one highly and one lowly correlated reference time series with the respective correlation values $\rho(B, R_1) = 0.83$ and $\rho(B, R_2) = 0.18$. The result of this experiment shows that the iterated truncated CD produces a correlation weighted recovery that reduces the relative MSE more to the highly correlated time series than the lowly correlated time series. The iterated truncated SVD performs an unweighted recovery that produces an almost equal reduction of the relative MSE to both reference time series.

[1]The data was kindly provided by HydroloGIS (http://www.hydrologis.edu).
[2]The data was kindly provided by Südtiroler Beratungsring (http://www.beratungsring.org).



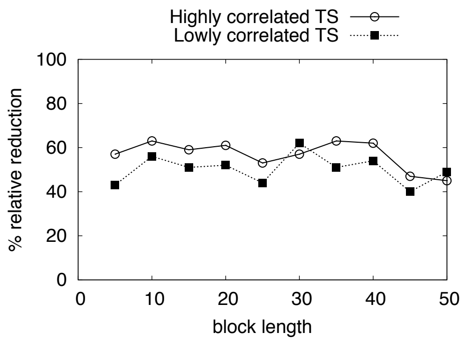
(a) Iterated truncated CD.



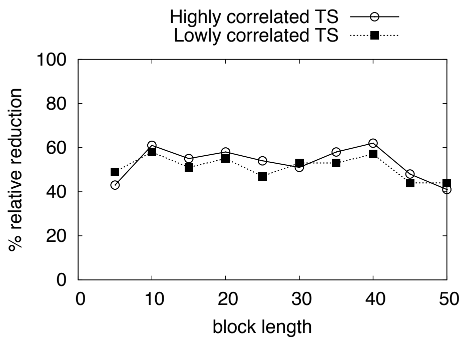
(b) Iterated truncated SVD.

Figure 3.3: MSE relative reduction of CD and SVD using highly and lowly correlated time series: case 1.

In Figure 3.4 we consider one highly and one lowly correlated reference time series with a correlation value $\rho(B, R_1) = 0.76$. We add also a lowly correlated time series with a correlation value $\rho(B, R_2) = 0.62$ that is higher than the one used in the experiment of Figure 3.3. As expected, the MSE relative reduction of the iterated truncated CD is slightly higher to R_1 than to R_2. The MSE relative reduction of the iterated truncated SVD remains similar to both reference time series.



(a) Iterated truncated CD.



(b) Iterated truncated SVD.

Figure 3.4: MSE relative reduction of CD and SVD using highly and lowly correlated time series: case 2.

Recovery accuracy

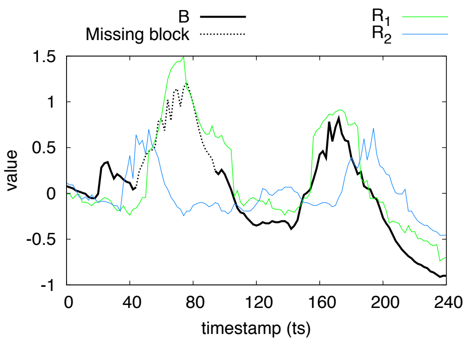
In this section we compare the recovery accuracy of the iterated truncated SVD against the iterated truncated CD using highly and lowly correlated time series.

In the experiment of Figure 3.5, we use three temperature time series from ***HYD*** measured respectively in Aria Borgo (B), Ponte Adige (R_1) and Aria La Villa (R_2) in the region of South Tyrol, Italy. B is highly correlated to R_1 with $\rho(B, R_1) = 0.75$. B is lowly correlated to R_2 with $\rho(B, R_2) = 0.32$. However, the peaks of B and R_2 exhibit shape similarity, i.e., the peaks contain similar spikes. The time shift is caused by the Foehn phenomenon (cf. Section 3.1). We drop from the base time series, B, a block for ts $\in [45, 95]$ and recover it using two reference time series, R_1 and R_2. The result of this experiment shows that the iterated truncated CD gives a weight to the reference time series proportional to their correlation with B, yielding a good block recovery accuracy, i.e., the amplitude and the shape of the missing block are accurately recovered. On the contrary, the iterated truncated SVD performs a block recovery that gives the same weight to both time series R_1 and R_2 at a time yielding a bad block recovery accuracy.

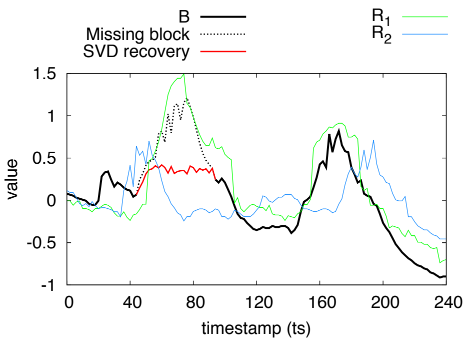
Figure 3.6 shows the MSE for removed blocks of values of increasing length from a base time series: starting from the middle of a block we increase the length of the removed block in both directions and we compute the MSE for each block. We run the experiment on five different base time series from *HYD* and we take the average of the MSE. For each run we use, in addition to the base time series, one highly correlated and one lowly correlated time series. As expected, the iterative truncated CD learns from the highly and lowly correlated time series at a time and thus, produces a small recovery error that slightly increases with the length of the missing block to recover. However, the recovery accuracy of the iterated truncated SVD considerably deteriorates with the length of the missing block to recover.

Impact of the time shift

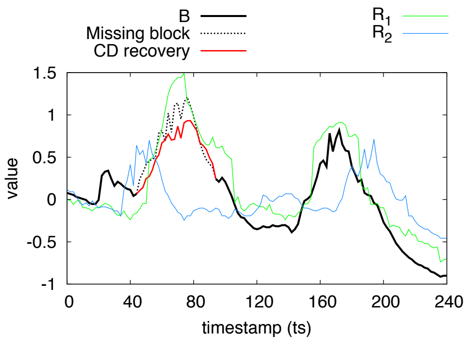
In Figure 3.7 we evaluate the impact of a varying time shift, denoted as s, on the recovery accuracy of the iterated truncated CD and the iterated truncated SVD. We show that for a high value of time shift, the two techniques produce similar block recoveries. In Figure 3.7(a) we take three time series from *SBR* measured respectively in Kaltern (B), Kollman (R_1) and Ritten (R_2) in the region of South Tyrol, Italy. The peaks of B and R_2 have a similar shape, but with a time shift. We drop one peak from B, we shift backwards R_2 with a value s and we compute



(a) B with 1 highly and 1 lowly correlated TS.



(b) Recovery of iterated truncated SVD.



(c) Recovery of iterated truncated CD.

Figure 3.5: Recovery using highly and lowly correlated hydrological TS.

the MSE recovery accuracy. The result of the experiment shows that starting from $s = 30$, the iterated truncated CD is not able anymore to exploit the lowly correlated time series and produces a block recovery similar to the one produced by the iterated truncated SVD.

3.5.2 Recovery using synthetic TS

For the following experiments, we consider a time series $sin(t)$ that has a small valley at each of the peaks, denoted as B, from which we drop a block of values for $t \in [70, 110]$ that we recover using both techniques.

Recovery accuracy

In Figure 3.8 we add to B one highly correlated time series $-0.5 * sin(t)$ denoted as R_1 such that $\rho(B, R_1) = 0.84$. We add also a lowly correlated time series by shifting B and we denote

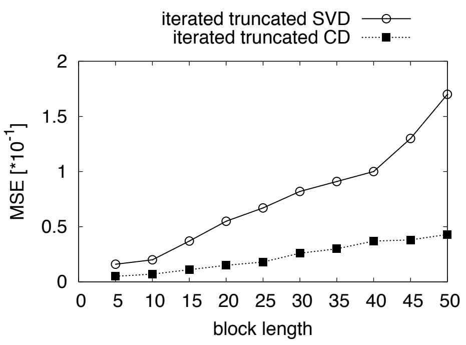
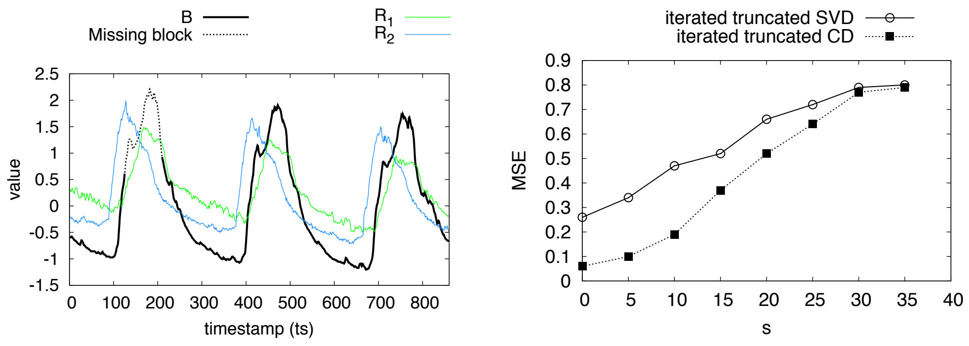


Figure 3.6: MSE for Successive Removed Blocks.



(a) Original time series.

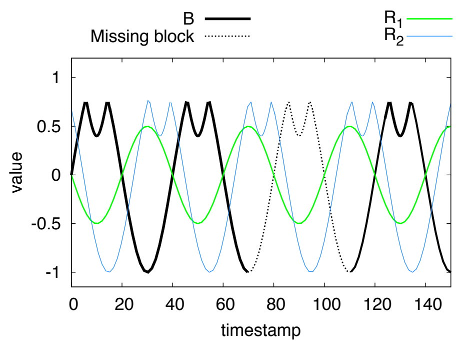
(b) MSE with varying time shift.

Figure 3.7: Impact of varying time shift

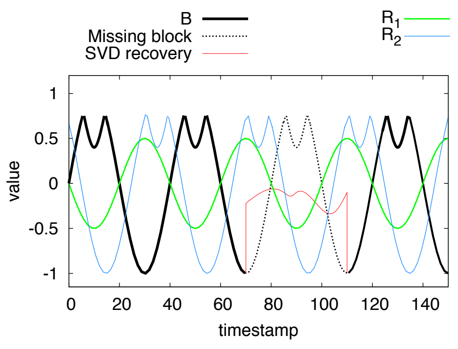
it as R_2 such that $\rho(B, R_2) = 0.16$. As expected, by giving a higher weight to R_2, the iterated truncated CD is able to perform a good recovery of the shape and the amplitude of the missing block. The iterated truncated SVD fails to recover the shape and the amplitude of the missing block.

Impact of Number of Input Time Series

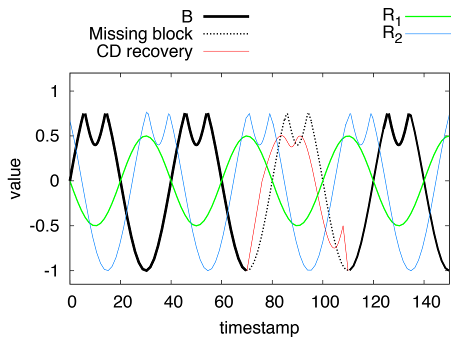
In Figure 3.9 we evaluate the robustness of the recovery produced by both techniques using a varying number of highly and lowly correlated time series. In Figure 3.9(a) we take B from the experiment of Figure 3.8 and one highly correlated time series with $\rho = 0.9$ to which we add a varying number of lowly correlated time series, by shifting $sin(t)$, such that $\rho \in [0.2, 0.6]$. The latter time series are added in the decreasing order of their correlation. This experiment shows that for $p_1 < 4$, the iterated truncated CD is able to use the most correlated time series yielding a smaller MSE than the iterated truncated SVD. For $p_1 \geq 4$, the MSE of both techniques



(a) R with 1 highly and 1 lowly correlated time series



(b) Recovery of iterated truncated SVD.

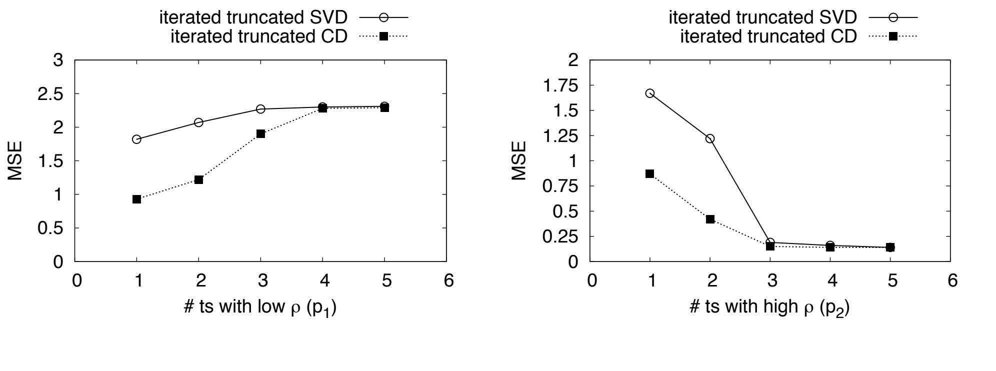


(c) Recovery of iterated truncated CD.

Figure 3.8: Recovery using highly and lowly correlated synthetic TS.

converges towards the same value. In the experiment of Figure 3.9(b) we take $sin(t)$ and one lowly correlated time series with $\rho = 0.2$ to which we add a varying number of highly correlated time series such that $\rho \in [0.7, 0.9]$. The latter time series are added in the increasing order of their correlation. In the presence of one lowly correlated time series, the iterated truncated SVD requires at least three additional highly correlated time series in order to reach the same MSE as one of the iterated truncated CD.

The experiment of Figure 3.9 shows that, for a close number of highly and lowly correlated time series, the correlation weighted recovery helps the iterated truncated CD to produce a better recovery than the one produced by the iterated truncated SVD. Otherwise, the two techniques produce similar recovery of missing values. However, the iterated truncated CD technique is computationally more efficient than the iterated truncated SVD, i.e., CD is linear with the number of input time series while SVD is cubic with the number of input time series.



(a) MSE using varying # of lowly correlated time series. (b) MSE using varying # of highly correlated time series

Figure 3.9: Recovery accuracy using varying number of input TS.

3.5.3 Comparison to SGD based recovery

In the experiment of Figure 3.10 we compare the accuracy recovery of the iterated truncated CD against GROUSE [KBV09] for the recovery of 20 missing values using an increasing number of time series where each contains 200 values. We omit the iterated truncated SVD from this experiment because of the high computational time. The result of this experiment shows that the iterated truncated CD produces a more accurate block recovery for low number of input time series. However, the recovery accuracy produced by GROUSE outperforms the one produced by the iterated truncated CD as the number of time series approaches the number of observations (cf. Section 3.2). In real world applications such as hydrology, the length of time series is much bigger than their number and thus, CD based recovery outperforms GROUSE recovery.

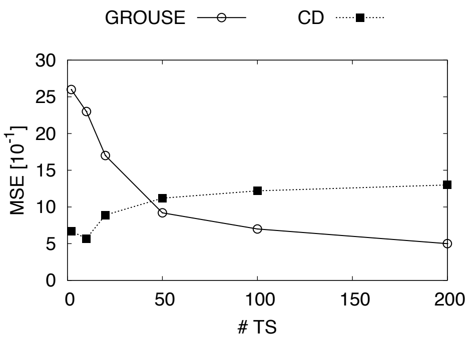


Figure 3.10: Recovery accuracy of CD against GROUSE.

3.5.4 Approximation accuracy

Figure 3.11 compares the approximation accuracy of the iterated truncated CD and the iterated truncated SVD to the input matrix. We use the Frobenius norm between the input matrix and the one obtained after the decomposition as an approximation error (cf. Section 3.4.1). The input matrix contains 10 columns where each one is a time series from *HYD*. This experiment shows that by updating all values of the input matrix at a time (and not only the missing ones), the two techniques perform similar approximation accuracy. The same result holds for different values of the truncation parameter k.

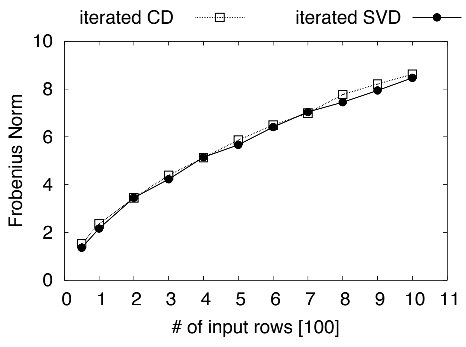


Figure 3.11: Approximation error.

3.5.5 Number of iterations of CD

In the experiment of Figure 3.12 we consider three temperature time series from *HYD*: a base time series, one highly correlated reference time series and one lowly correlated time series. We compute the number of iterations p required by the CD technique with an increasing number of rows n. The result of this experiment shows that p ranges between $\frac{n}{2}$ and $\frac{n}{3}$.

3.6 Conclusion

In this chapter, we compare the CD and SVD techniques for the recovery of missing values using time series with mixed correlation values. We empirically show that CD produces a weighted relative reduction of MSE that is proportional to the correlation of the input time series, while SVD produces an unweighted relative reduction of MSE. Our experiments on real world hydrological

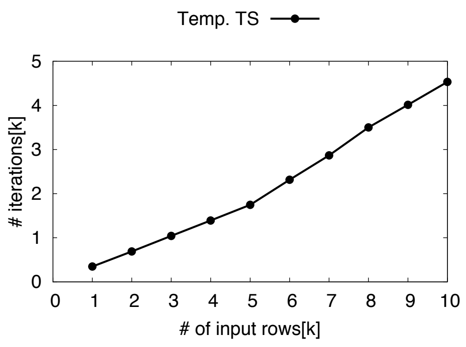


Figure 3.12: number of iterations performed by CD.

and synthetic time series also show that the iterated truncated CD performs a better recovery in case of similar number of highly and lowly correlated time series.

In future work, it would be of interest to compare the segmentation techniques that are applied in cases where the correlation ranking varies along the time series history. Another promising direction is to refine the definition of highly and lowly correlated time series.

CHAPTER 4

Memory-efficient Centroid Decomposition

Abstract

Real world applications that deal with time series data often rely on matrix decomposition techniques, such as the Singular Value Decomposition (SVD). The Centroid Decomposition (CD) approximates the Singular Value Decomposition, but does not scale to long time series because of the quadratic space complexity of the sign vector computation.

In this paper, we propose a greedy algorithm, termed Scalable Sign Vector (SSV), to efficiently determine sign vectors for CD applications with long time series, i.e., where the number of rows (observations) is much larger than the number of columns (time series). The SSV algorithm starts with a sign vector consisting of only 1s and iteratively changes the sign of the element that maximizes the benefit. The space complexity of the SSV algorithm is linear in the length of the time series. We provide proofs for the scalability, the termination and the correctness of the SSV algorithm. Experiments with real world hydrological time series and data sets from the UCR repository validate the analytical results and show the scalability of SSV.

4.1 Introduction

The Centroid Decomposition (CD) has been introduced as an approximation of the Singular Value Decomposition (SVD). It decomposes an input matrix, $\mathbf{X}$, into the product of two matrices, $\mathbf{X} = \mathbf{L} \cdot \mathbf{R}^T$, where $\mathbf{L}$ is the loading matrix and $\mathbf{R}$ is the relevance matrix ($\mathbf{R}^T$ denotes the transpose of $\mathbf{R}$). Each loading and relevance vector is determined based on a maximal centroid value, $\max \|\mathbf{X}^T \cdot Z\|$, which is equal to the norm of the product between the transpose of the input matrix and the sign vector Z consisting of 1s and -1s. Finding the *maximizing sign vector* Z that maximizes the centroid value is therefore at the core of the CD method. The classical approach [KT98] enumerates all possible sign vectors and chooses the one that maximizes the centroid value. This approach has linear space complexity since no data structures other than the input matrix are needed. However, its runtime is exponential. A more efficient solution to determine the maximizing sign vector has been introduced by Chu and Funderlic [CF01] and has quadratic runtime. The drawback of this solution is a quadratic space complexity since a correlation/covariance matrix is needed in addition to the input matrix.

In this work, we address the scalability of the Centroid Decomposition technique for an $n \times m$ matrix, $\mathbf{X}$, that represents m time series with n observations each, where n is much larger than m (i.e., $n \gg m$). We propose a greedy algorithm, termed *Scalable Sign Vector (SSV)*, to compute the maximizing sign vector. The basic idea is as follows: instead of searching for the maximizing sign vector using all elements of $\mathbf{X}$, we search for it by rows of $\mathbf{X}$. First, a sign vector Z is initialized to contain only 1s as elements. Then, the algorithm iteratively updates the sign of the element in Z that increases $\|\mathbf{X}^T \cdot Z\|$ most. The relevant element can be determined efficiently by checking all elements of a weight vector V, which is derived from $\mathbf{X}$. Instead of enumerating all possible sign vectors, our strategy generates only the vectors that most increase $Z^T \cdot V$. At the end of this iterative process, the sign vector Z that yields the maximal centroid value is found. The SSV algorithm has quadratic time (worst case) and linear space complexity. Compared to the classical approach, SSV reduces the runtime of the CD method from exponential to quadratic while keeping the same linear space complexity. Compared to the most efficient algorithm [CF01], SSV keeps the same quadratic runtime complexity, but reduces the space complexity from quadratic to linear.

Matrix decomposition techniques are widely used for time series data in a variety of real world applications, such as data prediction, recommender systems, image compression, recovery of missing values, stocks, etc. In most of these applications, only very few and short time series can be considered for the analysis due to the computational complexity of current solutions for matrix decomposition. As a consequence, not all relevant information of the original set of time series is considered for the decomposition. This is an unfortunate limitation since it can be imperative to use long time series to improve data analysis [LMP10]. For instance, in the recovery of missing values the most correlated time series are used to capture similar trends, and the use of longer time series improves the accuracy of the recovered values (as we will show in Section 4.6). Thus, scalable solutions that avoid an a priori segmentation of long time series are needed.

At the technical level, we provide an analysis and proofs of the correctness, the termination and the scalability of the SSV algorithm. The analytical results are confirmed by an in-depth empirical evaluation. In summary, the main contributions of this paper are the following:

- We propose a sign vector computation algorithm, called Scalable Sign Vector (SSV), that reduces the space complexity of the Centroid Decomposition technique from quadratic to linear, while keeping the same runtime complexity as the state-of-the-art solution.
- We prove that the space complexity of the SSV algorithm increases linearly with the length of the time series.
- We prove that the computation performed by the SSV algorithm is strictly monotonic. We use the monotonicity property to prove the correctness of the proposed solution.
- We prove that the SSV algorithm terminates and performs at most n iterations.
- We present the results of an experimental evaluation of the efficiency and scalability of the SSV algorithm on real world hydrological data and on datasets from the UCR repository.

The remainder of the paper is organized as follows. Section 4.2 reviews related work. Preliminary concepts and definitions are provided in Section 4.3. In Section 4.4, we present the SSV algorithm. Section 4.5 describes the main properties of the SSV algorithm. Section 4.6 reports the results of our experiments. Section 4.7 concludes the paper and discusses future work.

4.2 Related Work

The Centroid Decomposition (CD) has been introduced as an approximation of the Singular Value Decomposition [CF01]. It computes the centroid values, the loading vectors and the relevance vectors to approximate, respectively, the eigen values, the right singular vectors and the left singular vectors of SVD. Chu et al. [CF01, CFG95] prove that the CD approximation of SVD is the one that best minimizes the variance between corresponding elements. Thus, the variance between the centroid values computed by CD and the eigen values computed by SVD is minimal.

The most challenging part of the CD of a matrix $\mathbf{X}$ is the computation of the sign vector Z, consisting of 1s and -1s, that maximizes $||\mathbf{X}^T{\cdot}Z||$, where $\mathbf{X}^T$ is the transpose of $\mathbf{X}$ and $||{\cdot}||$ denotes the norm of a vector. The classical approach is based on the centroid method [DM01]. The centroid method uses a brute force search through an exponential number of sign vectors. Thus, the algorithm has exponential time and linear space complexity. This method has been used in various fields such as dimensionality reduction [PJR03], peak shift detection [RPBA09], etc.

The most efficient algorithm to find the maximizing sign vector was introduced by Chu and Funderlic [CF01], which we refer to as *Quadratic Sign Vector* (QSV). It transforms the maximization problem from $\max ||\mathbf{X}^T{\cdot}Z||$ to $\max (Z^T{\cdot}(\mathbf{X}{\cdot}\mathbf{X}^T){\cdot}Z)$ and achieves a quadratic runtime complexity. The space complexity is quadratic as well due to the construction of $\mathbf{X}{\cdot}\mathbf{X}^T$. Figure 4.1 illustrates the main data structures of the algorithm for an $n \times 3$ input matrix $\mathbf{X}$. Step 1 applies the transformation of the maximization problem, and Step 2 determines the maximizing sign vector. The set of all possible sign vectors can be considered as an n-dimensional hypercube, where each node represents a sign vector and is connected with all nodes representing a sign vector that differs in exactly one element. The QSV algorithm performs a traversal along the nodes of the hypercube, starting from the node that represents the sign vector $Z = [\text{-}1, \ldots, \text{-}1]^T$ until finding the node (and corresponding sign vector) that maximizes $Z^T{\cdot}(\mathbf{X}{\cdot}\mathbf{X}^T){\cdot}Z$.

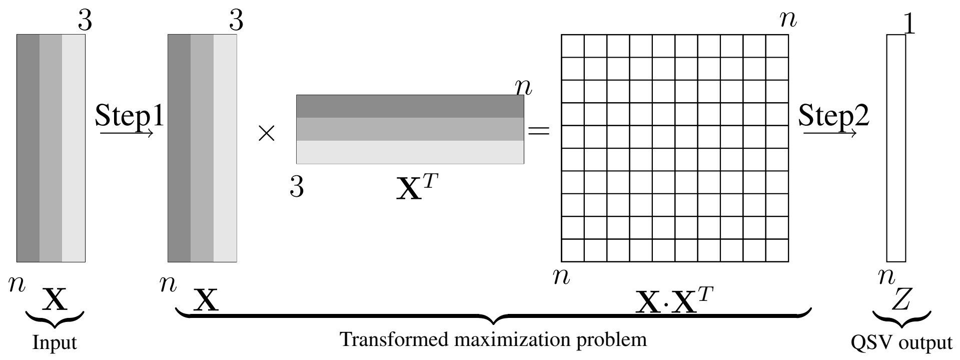


Figure 4.1: Illustration of the main data structures used by the QSV algorithm.

The Singular Value Decomposition (SVD) [GVL96, Mey00, Ski07] is a widely used matrix decomposition technique. SVD performs the decomposition by finding the eigen values with their corresponding left and right singular vectors. SVD has been used for dimensionality reduction [KAES99, HKP11], image compression [YJL04, WR97], missing values recovery [KB12, Bra03], etc. As most matrix decomposition techniques, SVD constructs a correlation/covariance matrix to find the eigen values and their corresponding singular vectors. Several algorithms have been proposed to make SVD for an $n \times m$ matrix faster for special cases of n and m. For instance, for $m > \frac{5}{3}n$, the runtime of SVD is reduced from $4n^2m + 8nm^2 + 9m^3$ to $2n^2m + 2m^3$ [TB97]. Less attention has been given to reduce the space complexity while keeping the same runtime complexity, which is the goal of this paper.

Rendle [Ren10, Ren13] introduces Factorization Machines (FM) that perform a decomposition of large input matrices. Factorization Machines perform learning and prediction with input matrices of millions of values. For specific input matrices, the result obtained by the application of FM contains the decomposition matrices produced by SVD. The approach assumes the existence of repeating patterns in the input matrix. Each repeating pattern is represented by a block of data, and the decomposition is computed using these blocks of data.

Papalexakis et al. [PFS12] present a scalable solution to compute tensor ($3d$ matrix in this work) decompositions [KB09]. More specifically, a parallel approach to compute the PARAFAC decomposition is proposed, which is a multidimensional generalization of SVD. The proposed solution works in three steps: create random samples of the input tensor, apply a parallel decomposition on each of them, and merge the result of each decomposition. This gives an approximation of the decomposition of the entire input tensor. The method scales linearly to millions of values. However, it is applicable only to sparse tensors, where more than 90% of the elements of the input tensor are equal to zero. Only the non-zero elements are used in the decomposition. In contrast, our solution performs the matrix decomposition for any type of input matrices.

Gemulla et al. [GNHS11] propose a large-scale matrix decomposition technique that, similar to CD, decomposes an input matrix into the product of two matrices. The proposed technique implements a scalable distributed Stochastic Gradient Descent (SGD) method [ZWSL10]. The latter computes a loss function that minimizes an error value between the input matrix and the product of the two matrices. The method works as follows. First, the input matrix is partitioned into blocks that are distributed across a MapReduce cluster. Then, the loss function is computed as the sum of local loss functions, each of which is computed in parallel over a single data block. This technique exploits the fact that the computation of local loss functions can be swapped without changing the final result of the decomposition. This is possible since each local loss function is computed over a different row and/or column of the input matrix. This technique scales to large input matrices consisting of millions of rows and columns. The technique we propose in this paper cannot be distributed since it computes a sequence of vectors, where each vector relies on the previous one.

Li et al. [LTS13] follow the same idea to distribute the computation of a matrix decomposition across a MapReduce cluster. They propose a parallelizable computation of the SGD method using Resilient Distributed Datasets (RDDs) [XRZ$^+$13], but investigate the case when the individual blocks do not fit into the main memory of a node. The proposed solution is based on a hash table that stores partitions of data blocks in memory. It scales to large data sets containing millions of values. The work uses the fact that the decomposition can be performed in parallel over separate rows and/or columns of data. Our solution cannot be computed over separate rows and/or columns of the input matrix since the sequence of computed vectors requires the use of the entire matrix.

The methods described in Gemulla et al. [GNHS11] and Li et al. [LTS13] implement a scalable SGD method. They do not assume any constraints about the input matrix, but compute a decomposition that is different from the one produced by SVD. The SGD method minimizes a given error value, which is different from CD. This approach makes SGD suitable for applications where an error value needs to be minimized, such as in recommender systems [KBV09] used in Netflix [AT05, MM09, BL07] or MovieLens [MAL$^+$03].

The solution proposed in this paper describes a scalable implementation of the CD matrix decomposition technique that approximates SVD. Our solution uses the entire input matrix for the computation and can therefore not be computed over separate blocks of rows and/or columns in parallel. Instead of distributing the decomposition across clusters, we propose to reduce the space complexity of the decomposition method.

4.3 Preliminaries

4.3.1 Notations

Bold upper-case letters refer to matrices, regular font upper-case letters to vectors (rows and columns of matrices) and lower-case letters to elements of vectors/matrices. For example, $\mathbf{X}$ is a matrix, X_{i*} is the i-th row of $\mathbf{X}$, X_{*i} is the i-th column of $\mathbf{X}$, $(X_{i*})^T$ is the transpose of X_{i*} and x_{ij} is the j-th element of X_{i*}.

An $n \times m$ matrix $\mathbf{X} = [X_{*1}|\ldots|X_{*m}]$ contains as columns m time series X_{*j} and as rows n values for each time series (cf. Section 3.3). Our database contains up to 200 real world hydrological time series with each of them containing up to 120k values. Thus, we consider matrices where the length of time series n is much larger than the number of time series m, i.e., $n \gg m$.

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

4.3.2 Centroid Decomposition

The *Centroid Decomposition (CD)* is a decomposition technique that 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 $\mathbf{R}$.

Algorithm 4: CD($\mathbf{X}$, n, m)

```
Input: $n \times m$ matrix $\mathbf{X}$
Output: $\mathbf{L}$, $\mathbf{R}$
1  $L = R = []$;
2  for $i = 1$ to $m$ do
3      $Z = FindSignVector(\mathbf{X}, n, m)$;
4      $C_{*i} = \mathbf{X}^T \cdot Z$;
5      $R_{*i} = \frac{C_{*i}}{||C_{*i}||}$;
6      $\mathbf{R} = Append(\mathbf{R}, R_{*i})$;
7      $L_{*i} = \mathbf{X} \cdot R_{*i}$;
8      $\mathbf{L} = Append(\mathbf{L}, L_{*i})$;
9      $\mathbf{X} := \mathbf{X} - L_{*i} \cdot R_{*i}^T$;
10 return $\mathbf{L}$, $\mathbf{R}$
```

Algorithm 4 computes the CD of an input matrix $\mathbf{X}$ into matrices $\mathbf{L}$ and $\mathbf{R}$. At each iteration i, function FindSignVector() determines the sign vector Z that yields the maximal centroid value $\|\mathbf{X}^T \cdot Z\|$ (where $\|\mathbf{X}^T \cdot Z\|$ is equal to the square root of the squared elements of $\mathbf{X}^T \cdot Z$). We call Z the *maximizing sign vector*. Next, the centroid vector, C_{*i}, and the centroid value, $\|C_{*i}\|$, are computed. Finally, the 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 next 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 are found.

Example 11. Consider a matrix $\mathbf{X}$ and two sign vectors:

$$\mathbf{X}= \begin{bmatrix} 2 & \text{-}2 \\ 0 & 3 \\ \text{-}4 & 2 \end{bmatrix} \quad \mathbf{X}^T= \begin{bmatrix} 2 & 0 & \text{-}4 \\ \text{-}2 & 3 & 2 \end{bmatrix} \quad Z_1= \begin{bmatrix} 1 \\ \text{-}1 \\ 1 \end{bmatrix} \quad Z_2= \begin{bmatrix} \text{-}1 \\ 1 \\ 1 \end{bmatrix}$$

The centroid values for the two sign vectors are computed as $\|\mathbf{X}^T \cdot Z_1\| = \sqrt{(\text{-}2)^2 + (\text{-}3)^2} = \sqrt{13}$ and $\|\mathbf{X}^T \cdot Z_2\| = \sqrt{(\text{-}6)^2 + 7^2} = \sqrt{85}$. Since $\|\mathbf{X}^T \cdot Z_2\| > \|\mathbf{X}^T \cdot Z_1\|$, Z_2 is the maximizing sign vector (among the two sign vectors).

4.3.3 Application of CD

The following example illustrates how to interpret the Centroid Decomposition of a matrix of m time series. Let $F = \{f_1, \ldots, f_m\}$ be the set of m factors that (most) influence the values in the time series.

Example 12. Consider a 2×3 matrix $\mathbf{X} = \{X_1, X_2\}$ that consists of two time series $X_1 = \{2, 0, \text{-}4\}$ and $X_2 = \{\text{-}2, 3, 2\}$. X_1 is the temperature in Zurich, and X_2 is the temperature in Basel. The CD method decomposes $\mathbf{X}$ by finding the loading and the relevance vectors with respect to each time series as shown in Figure 4.2.

If $F = \{nbrSunnyHours, amntRain\}$ and if each temperature time series is mainly influenced by the two factors of F, the Centroid Decomposition shows how to obtain the temperature values in a specific city using these two factors. For instance, the first value of the temperature in Zurich shown in gray color (i.e., 2) is obtained using a loading value of -2.820 for $nbrSunnyHours$ with a relevance value of -0.651 to which we add a loading value of 0.217 for $amntRain$ with a relevance value of 0.759.

The result of the decomposition in Example 12 can be used to recover missing values. Let's

$$\mathbf{X} = \begin{bmatrix} 2 & \text{-}2 \\ 0 & 3 \\ \text{-}4 & 2 \end{bmatrix}; \ \mathrm{CD}(\mathbf{X}) = \underbrace{\begin{bmatrix} \text{-}2.820 & 0.217 \\ 2.278 & 1.952 \\ 4.122 & \text{-}1.735 \end{bmatrix}}_{\mathbf{L}}, \underbrace{\begin{bmatrix} \text{-}0.651 & 0.759 \\ 0.759 & 0.651 \end{bmatrix}}_{\mathbf{R}}$$

$$\text{such that } \mathbf{X} = \begin{bmatrix} 2 & \text{-}2 \\ 0 & 3 \\ \text{-}4 & 2 \end{bmatrix} = \underbrace{\begin{bmatrix} \text{-}2.820 & 0.217 \\ 2.278 & 1.952 \\ 4.122 & \text{-}1.735 \end{bmatrix}}_{\mathbf{L}} \times \underbrace{\begin{bmatrix} \text{-}0.651 & 0.759 \\ 0.759 & 0.651 \end{bmatrix}}_{\mathbf{R}^T}$$

Figure 4.2: Example of Centroid Decomposition.

assume the second value of the temperature in Basel is missing. The first step of the recovery process is to initialize the missing value using a classical imputation technique, e.g., linear interpolation. Then, we apply the Centroid Decomposition to learn the loading and relevance values of the two factors to refine the initialized value. The refined values better approximate the missing value. We show the result of the recovery based on Centroid Decomposition in Section 4.6.2.

4.4 Scalable Sign Vector

This section presents a scalable sign vector (SSV) computation technique, which has the same quadratic runtime complexity as the QSV algorithm [CF01], but requires only linear space. The core of the solution is the transformation of the maximization of the centroid values $||\mathbf{X}^T \cdot Z||$ into a new and equivalent maximization problem that can be computed efficiently.

4.4.1 Overview and Data Structures

Figure 4.3 illustrates the SSV computation method. We transform the maximization of the centroid values to a new maximization problem that is based on a sign vector Z and a weight vector V that is derived from $\mathbf{X}$. More specifically, a sequence of vector pairs $V^{(k)}$ and $Z^{(k)}$ is iteratively computed, beginning with $Z^{(1)} = [1, \dots, 1]^T$. In each iteration, the sign vector is changed at the position that maximizes the product of the two vectors. The last sign vector $Z^{(k)}$ $(1 \leq k \leq n)$ is the maximizing sign vector.

Figure 4.4 illustrates the Centroid Decomposition of an input matrix $\mathbf{X}$ using SSV. In the first

$$\underbrace{\underbrace{\begin{bmatrix} 2 & -2 \\ 0 & 3 \\ -4 & 2 \end{bmatrix}}_{\mathbf{X}}}_{\text{Input}} \rightarrow \underbrace{\underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{Z^{(1)}} \underbrace{\begin{bmatrix} -18 \\ 0 \\ -6 \end{bmatrix}}_{V^{(1)}} \underbrace{\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}}_{Z^{(2)}} \underbrace{\begin{bmatrix} -18 \\ 12 \\ 18 \end{bmatrix}}_{V^{(2)}}}_{\text{SSV computation}} \rightarrow \underbrace{\underbrace{\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}}_{Z^{(2)}}}_{\text{SSV output}}$$

Figure 4.3: Illustration of SSV.

iteration of Algorithm 4, we use $\mathbf{X}$ and SSV to derive the first maximizing sign vector. We then compute vector L_{*1} and R_{*1} according to Algorithm 4. In the second iteration, we update matrix $\mathbf{X}$ to $\mathbf{X}' = \mathbf{X} - L_{*1} \cdot R_{*1}^T$ and repeat the process, i.e., derive the second maximizing sign vector Z' and compute vectors L_{*2} and R_{*2}.

$$\underbrace{\begin{bmatrix} 2 & -2 \\ 0 & 3 \\ -4 & 2 \end{bmatrix}}_{\mathbf{X}} \rightarrow \underbrace{\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}}_{Z} \rightarrow \underbrace{\begin{bmatrix} -2.820 \\ 2.278 \\ 4.122 \end{bmatrix}}_{L_{*1}} \underbrace{\begin{bmatrix} -0.651 \\ 0.759 \end{bmatrix}}_{R_{*1}} \rightarrow$$

(a) Iteration 1

$$\rightarrow \underbrace{\begin{bmatrix} 0.164 & 0.141 \\ 1.482 & 1.270 \\ -1.317 & -1.129 \end{bmatrix}}_{\mathbf{X}'} \rightarrow \underbrace{\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}}_{Z'} \rightarrow \underbrace{\begin{bmatrix} 0.217 \\ 1.952 \\ -1.735 \end{bmatrix}}_{L_{*2}} \underbrace{\begin{bmatrix} 0.759 \\ 0.651 \end{bmatrix}}_{R_{*2}}$$

(b) Iteration 2

Figure 4.4: Illustration of Centroid Decomposition using SSV.

4.4.2 Transformation of the Maximization Problem

In this section, we present a transformation of the maximization of $||\mathbf{X}^T \cdot Z||$ into a new and equivalent maximization problem and we show that the new maximization problem can be efficiently computed with linear space complexity.

The following auxiliary function is used: $diag^{=0}(\mathbf{X})$ sets the diagonal values of an $n \times n$ matrix $\mathbf{X}$ to 0.

The following lemma introduces a maximization equivalence, which states that maximizing $||\mathbf{X}^T \cdot Z||$ over all possible sign vectors Z is equivalent to maximizing the product of Z^T and the vector $V = diag^{\bar{=}0}(\mathbf{X}{\cdot}\mathbf{X}^T) \cdot Z$. This maximization equivalence will be used to define the new maximization problem.

Lemma 5 (Maximization Equivalence). *Let matrix* $\mathbf{X} = [X_{*1}| \dots |X_{*m}]$ *be an* $n \times m$ *matrix and* V *be the vector* $V = diag^{\bar{=}0}(\mathbf{X}{\cdot}\mathbf{X}^T){\cdot}Z$. *The following equivalence holds:*

$$\underset{Z\in\{1,-1\}^n}{\arg\max}\ \|\mathbf{X}^T{\cdot}Z\| \equiv \underset{Z\in\{1,-1\}^n}{\arg\max}\ Z^T{\cdot}V.$$

Proof. We expand both sides of the equivalence and show that the expanded expressions are equivalent.

The transformation of the expression on the left-hand side yields

$$\begin{aligned}
&\underset{Z\in\{-1,1\}^n}{\arg\max}\ \|\mathbf{X}^T{\cdot}Z\| \equiv \\
&\equiv \underset{Z\in\{-1,1\}^n}{\arg\max}\ \|\mathbf{X}^T{\cdot}Z\|^2 \\
&\equiv \underset{Z\in\{-1,1\}^n}{\arg\max}\ ((\sum_{i=1}^{n} x_{i1}{\times}z_i)^2 + \cdots + (\sum_{i=1}^{n} x_{im}{\times}z_i)^2) \\
&\equiv \underset{Z\in\{-1,1\}^n}{\arg\max}\ ((\sum_{i=1}^{n} \tilde{x}_{i1})^2 + \cdots + (\sum_{i=1}^{n} \tilde{x}_{im})^2),
\end{aligned}$$

where $\tilde{x}_{ij} = x_{ij} \times z_i$ for $j = 1, \dots, m$. Notice that the first step takes the square of the norm, which has no impact on the vector Z that maximizes the norm. Next, we use the square of sum rule

$$(\sum_{i=1}^{n} x_i)^2 = \sum_{i=1}^{n} x_i^2 + 2{\times} \sum_{j=2}^{n}\sum_{i=1}^{j-1} x_i{\times}x_j$$

and transform the above expression into

$$\begin{aligned}
&\underset{Z\in\{-1,1\}^n}{\text{arg max}}\ \|\mathbf{X}^T{\cdot}Z\| \equiv \\
&\quad\equiv \underset{Z\in\{-1,1\}^n}{\text{arg max}}\ (\sum_{i=1}^{n}\tilde{x}_{i1}^2 + 2\times\sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{i1}\times\tilde{x}_{j1} + \\
&\qquad\qquad\qquad\vdots \\
&\qquad\qquad \sum_{i=1}^{n}\tilde{x}_{im}^2 + 2\times\sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{im}\times\tilde{x}_{jm}).
\end{aligned}$$

Since $\tilde{x}_{ij} = x_{ij} \times z_i$ with $z_i \in \{-1, 1\}$, we have $\tilde{x}_{ij}^2 = x_{ij}^2$. That is, the terms $\sum_{i=1}^{n}\tilde{x}_{ij}^2$ for $j = 1, \ldots, m$ in the above expression are constant and independent of the sign vector Z. Therefore, they can be removed from the maximization problem, which yields

$$\begin{aligned}
&\underset{Z\in\{-1,1\}^n}{\text{arg max}}\ \|\mathbf{X}^T{\cdot}Z\| \equiv \\
&\quad\equiv \underset{Z\in\{-1,1\}^n}{\text{arg max}}\ (2\times\underbrace{(\sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{i1}\times\tilde{x}_{j1} + \cdots + \sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{im}\times\tilde{x}_{jm})}_{\sum_{k=1}^{m}\sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{ik}\times\tilde{x}_{jk}}) \\
&\quad\equiv \underset{Z\in\{-1,1\}^n}{\text{arg max}}\ (2\times\sum_{k=1}^{m}\sum_{j=2}^{n}\sum_{i=1}^{j-1}\tilde{x}_{ik}\times\tilde{x}_{jk}).
\end{aligned} \tag{4.1}$$

Next, we transform the expression on the right-hand side. From the definition of V we get

$$\underset{Z\in\{-1,1\}^n}{\text{arg max}}\ Z^T{\cdot}V \equiv \underset{Z\in\{-1,1\}^n}{\text{arg max}}\ (Z^T{\cdot}diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T){\cdot}Z).$$

The matrix representation of $diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T)$ using the rows of $\mathbf{X}$ is given as

$$diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T) = \begin{bmatrix} 0 & X_{1*}{\cdot}(X_{2*})^T & \cdots & X_{1*}{\cdot}(X_{n*})^T \\ X_{2*}{\cdot}(X_{1*})^T & 0 & \cdots & X_{2*}{\cdot}(X_{n*})^T \\ \vdots & \vdots & \ddots & \vdots \\ X_{n*}{\cdot}(X_{1*})^T & X_{n*}{\cdot}(X_{2*})^T & \cdots & 0 \end{bmatrix}.$$

We use this representation and transform the argument of the $\arg\max$ function as follows:

$$
\begin{aligned}
&Z^T \cdot diag^{=0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z = \\
&= Z^T \cdot \begin{bmatrix} 0 & X_{1*} \cdot (X_{2*})^T & \cdots & X_{1*} \cdot (X_{n*})^T \\ X_{2*} \cdot (X_{1*})^T & 0 & \cdots & X_{2*} \cdot (X_{n*})^T \\ \vdots & \vdots & \ddots & \vdots \\ X_{n*} \cdot (X_{1*})^T & X_{n*} \cdot (X_{2*})^T & \cdots & 0 \end{bmatrix} \cdot Z \\
&= Z^T \cdot \begin{bmatrix} 0 + z_2 \times (X_{1*} \cdot (X_{2*})^T) + \cdots + z_n \times (X_{1*} \cdot (X_{n*})^T) \\ z_1 \times (X_{2*} \cdot (X_{1*})^T) + 0 + \cdots + z_n \times (X_{2*} \cdot (X_{n*})^T) \\ \vdots \\ z_1 \times (X_{n*} \cdot (X_{1*})^T) + z_2 \times (X_{n*} \cdot (X_{2*})^T) + \cdots + 0 \end{bmatrix} \\
&= z_1 \times (0 + z_2 \times (X_{1*} \cdot (X_{2*})^T) + \cdots + z_n \times (X_{1*} \cdot (X_{n*})^T)) + \\
&\quad z_2 \times (z_1 \times (X_{2*} \cdot (X_{1*})^T) + 0 + \cdots + z_n \times (X_{2*} \cdot (X_{n*})^T)) + \\
&\quad \vdots \\
&\quad z_n \times (z_1 \times (X_{n*} \cdot (X_{1*})^T) + z_2 \times (X_{n*} \cdot (X_{2*})^T) + \cdots + 0).
\end{aligned}
$$

The vector products in the above expression are replaced by a sum, i.e., $X_{i*} \cdot (X_{j*})^T = \sum_{k=1}^{m} x_{ik} \times x_{jk}$, which gives

$$
\begin{aligned}
&Z^T \cdot diag^{=0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z = \\
&= z_1 \times (0 + z_2 \times \sum_{k=1}^{m} x_{1k} \times x_{2k} + \cdots + z_n \times \sum_{k=1}^{m} x_{1k} \times x_{nk}) + \\
&\quad z_2 \times (z_1 \times \sum_{k=1}^{m} x_{2k} \times x_{1k} + 0 + \cdots + z_n \times \sum_{k=1}^{m} x_{2k} \times x_{nk}) + \\
&\quad \vdots \\
&\quad z_n \times (z_1 \times \sum_{k=1}^{m} x_{nk} \times x_{1k} + z_2 \times \sum_{k=1}^{m} x_{nk} \times x_{2k} + \cdots + 0).
\end{aligned}
$$

Finally, we push the elements of the sign vector into the sum and replace $z_i \times x_{ik}$ by $\tilde{x}_{ik}$, which gives

$$\begin{aligned}
&Z^T \cdot diag^{\bar{=}0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z = \\
&= 0 + \sum_{k=1}^{m} \tilde{x}_{1k} \times \tilde{x}_{2k} + \cdots + \sum_{k=1}^{m} \tilde{x}_{1k} \times \tilde{x}_{nk} + \\
&\quad \sum_{k=1}^{m} \tilde{x}_{2k} \times \tilde{x}_{1k} + 0 + \cdots + \sum_{k=1}^{m} \tilde{x}_{2k} \times \tilde{x}_{nk} + \\
&\quad \vdots \\
&\quad \sum_{k=1}^{m} \tilde{x}_{nk} \times \tilde{x}_{1k} + \sum_{k=1}^{m} \tilde{x}_{nk} \times \tilde{x}_{2k} + \cdots + 0 \\
&= 2 \times \sum_{k=1}^{m} \sum_{j=2}^{n} \sum_{i=1}^{j-1} \tilde{x}_{ik} \times \tilde{x}_{jk}.
\end{aligned}$$

We insert this expression in the $\arg\max$ function, which gives Equation (4.1). □

Lemma 5 forms the basis for a new and equivalent maximization problem. Instead of maximizing $\|\mathbf{X}^T \cdot Z\|$, the product $Z^T \cdot V$ is maximized over all sign vectors $Z \in \{1, \text{-}1\}^n$. Since the computation of $V = diag^{\bar{=}0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z$ has quadratic space complexity (due to the construction of the matrix $\mathbf{X} \cdot \mathbf{X}^T$), we proceed by showing how to avoid this product and how to compute V directly from $\mathbf{X}$.

Lemma 6. *Let $\mathbf{X} = [X_{*1}| \ldots |X_{*m}]$ be an $n \times m$ matrix and v_i be the i-th element of the vector $V = diag^{\bar{=}0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z$. Then, the following holds:*

$$v_i = z_i \times (z_i \times X_{i*} \cdot Z^T \cdot \mathbf{X} - X_{i*} \cdot (X_{i*})^T).$$

Proof. We use the matrix representation of $diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)$ to compute V as follows:

$$diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z =$$

$$= \begin{bmatrix} 0 & X_{1*}\cdot(X_{2*})^T & \cdots & X_{1*}\cdot(X_{n*})^T \\ X_{2*}\cdot(X_{1*})^T & 0 & \cdots & X_{2*}\cdot(X_{n*})^T \\ \vdots & \vdots & \ddots & \vdots \\ X_{n*}\cdot(X_{1*})^T & X_{n*}\cdot(X_{2*})^T & \cdots & 0 \end{bmatrix} \cdot Z$$

$$= \begin{bmatrix} X_{1*}\cdot(0 + z_2\times(X_{2*})^T + \cdots + z_n\times(X_{n*})^T) \\ X_{2*}\cdot(z_1\times(X_{1*})^T + 0 + \cdots + z_n\times(X_{n*})^T) \\ \vdots \\ X_{n*}\cdot(z_1\times(X_{1*})^T + z_2\times(X_{2*})^T + \cdots + 0) \end{bmatrix}$$

$$= \begin{bmatrix} z_1 \times (z_1 \times X_{1*}\cdot \sum_{j=1}^{n}(z_j \times (X_{j*})^T) - X_{1*}\cdot(X_{1*})^T) \\ z_2 \times (z_2 \times X_{2*}\cdot \sum_{j=1}^{n}(z_j \times (X_{j*})^T) - X_{2*}\cdot(X_{2*})^T) \\ \vdots \\ z_n \times (z_n \times X_{n*}\cdot \sum_{j=1}^{n}(z_j \times (X_{j*})^T) - X_{n*}\cdot(X_{n*})^T) \end{bmatrix}$$

Since we have that $Z^T\cdot\mathbf{X} = \sum_{j=1}^{n}(z_j \times (X_{j*})^T)$, it follows that $v_i = z_i \times (z_i \times X_{i*}\cdot Z^T\cdot\mathbf{X} - X_{i*}\cdot(X_{i*})^T)$. □

Based on Lemma 6 we show that the space complexity of computing vector V (i.e., $diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z$) is linear in the number of rows of $\mathbf{X}$.

Lemma 7 (Linear Space)**.** *For an $n \times m$ matrix $\mathbf{X}$, the computation of $V = diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z$ has $O(n)$ space complexity.*

Proof. The result of $\sum_{j=1}^{n}(z_j \times (X_{j*})^T)$ is computed by keeping in memory a single row X_{j*} and one element of Z at a time, which requires $O(m)$ space. This sum is computed only once. To compute the individual elements $v_i = z_i \times (X_{i*}\cdot Z^T\cdot\mathbf{X} - X_{i*}\cdot(X_{i*})^T)$ of the result vector of $diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z$, $\mathbf{X}$ is read again, one row at a time. The result vector has length n. Since $n \gg m$, the space complexity of $diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z$ is $O(n)$. □

4.4.3 Computation of Maximizing Sign Vectors

We present now an algorithm with linear space complexity to compute the maximizing sign vector Z according to the maximization problem introduced in Lemma 5. The basic idea is as follows. We begin with the sign vector $Z = [1, \ldots, 1]^T$ and iteratively change the sign of one element in Z that increases $Z^T \cdot V$ most. The algorithm terminates when $Z^T \cdot V$ cannot be increased further with this strategy.

Algorithm 5: SSV($\mathbf{X}$, n, m)

```
Input: $n \times m$ matrix $\mathbf{X}$
Output: maximizing sign vector $Z^T = [z_1, \ldots, z_n]$
1  $pos = 0$;
2  repeat
       // Change sign
3      if $pos = 0$ then $Z^T = [1, \ldots, 1]$;
4      else change the sign of $z_{pos}$;
       // Determine $S$ and $V$
5      $S = \sum_{i=1}^{n}(z_i \times (X_{i*})^T)$;
6      $V = []$;
7      for $i = 1$ to $n$ do
8          $v_i = z_i \times (z_i \times X_{i*} \cdot S - X_{i*} \cdot (X_{i*})^T)$;
9          Insert $v_i$ in $V$;
       // Search next element
10     $val = 0, pos = 0$;
11     for $i = 1$ to $n$ do
12         if $(z_i \times v_i < 0)$ then
13             if $|v_i| > val$ then
14                 $val = v_i$;
15                 $pos = i$;
16 until $pos = 0$;
17 return $Z$;
```

Algorithm 5 implements this strategy and computes the maximizing sign vector. Note that V is computed directly from $\mathbf{X}$ by reading the matrix row by row, one row at a time: first to compute the intermediate vector S and then to compute the individual elements of V. We search for the index (pos) of the element $v_i \in V$ with the largest absolute value such that v_i and $z_i \in Z$ have different signs, i.e., $z_i \times v_i < 0$. If such an element is found, the sign of z_i is changed. A new vector V is computed, which is different from the vector in the previous iteration due to the sign change. The iteration terminates when the signs of all corresponding elements in V and Z are the same. The vector Z in the final iteration is the maximizing sign vector that maximizes $Z^T \cdot V$.

Example 13. To illustrate the computation of the sign vector using Algorithm 5, consider the input matrix of our running example, i.e.,

$$\mathbf{X} = \begin{bmatrix} 2 & \text{-}2 \\ 0 & 3 \\ \text{-}4 & 2 \end{bmatrix}.$$

First, Z is initialized with 1s, and S and V are computed:

$$S = \begin{bmatrix} 2 \\ \text{-}2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} + \begin{bmatrix} \text{-}4 \\ 2 \end{bmatrix} = \begin{bmatrix} \text{-}2 \\ 3 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 2 & \text{-}2 \end{bmatrix} \times \begin{bmatrix} \text{-}2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 & \text{-}2 \end{bmatrix} \times \begin{bmatrix} 2 \\ \text{-}2 \end{bmatrix} = \text{-}18$$

$$v_2 = \begin{bmatrix} 0 & 3 \end{bmatrix} \times \begin{bmatrix} \text{-}2 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 & 3 \end{bmatrix} \times \begin{bmatrix} 0 \\ 3 \end{bmatrix} = 0$$

$$v_3 = \begin{bmatrix} \text{-}4 & 2 \end{bmatrix} \times \begin{bmatrix} \text{-}2 \\ 3 \end{bmatrix} - \begin{bmatrix} \text{-}4 & 2 \end{bmatrix} \times \begin{bmatrix} \text{-}4 \\ 2 \end{bmatrix} = \text{-}6$$

i.e.,

$$Z^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad V^{(1)} = \begin{bmatrix} \text{-}18 \\ 0 \\ \text{-}6 \end{bmatrix}.$$

Two elements of $Z^{(1)}$ have a different sign from the corresponding elements in $V^{(1)}$. Thus, the algorithm iterates through the elements in $V^{(1)}$ to search for the index of the element $v_i \in V^{(1)}$ with the largest absolute value, such that $z_i \in Z^{(1)}$ and v_i have different signs. This search returns $pos = 1$. In the second iteration, we change the sign of the element at position 1 in $Z^{(1)}$ and we use the new sign vector $Z^{(2)}$ to compute $V^{(2)}$, similar to iteration 1, and get

$$Z^{(2)} = \begin{bmatrix} \text{-1} \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad V^{(2)} = \begin{bmatrix} \text{-18} \\ 12 \\ 18 \end{bmatrix}.$$

Since all corresponding elements in $Z^{(2)}$ and $V^{(2)}$ have the same sign, the algorithm terminates and $Z^{(2)}$ is returned as the maximizing sign vector that maximizes $Z^T \cdot V$.

4.5 Properties of the Algorithm

This section works out the main properties of the SSV algorithm. More specifically, we prove monotonicity, termination and correctness of our algorithm.

4.5.1 Monotonicity

Let $Z^{(k)}$ and $V^{(k)}$ refer, respectively, to vectors V and Z in the k-th iteration of the SSV algorithm. $v_i^{(k)}$ and $z_i^{(k)}$ denote, respectively, the i-th element of $V^{(k)}$ and $Z^{(k)}$. Lemma 8 shows that the computation of the maximizing sign vector in the SSV algorithm is *strictly monotonic*, i.e., each iteration increases the value of $Z^T \cdot V$.

Lemma 8 (Monotonicity). *For any two consecutive iterations k and $k+1$ in the SSV algorithm the following holds:*

$$(Z^{(k+1)})^T \cdot V^{(k+1)} > (Z^{(k)})^T \cdot V^{(k)}.$$

Proof. First, $(Z^{(k+1)})^T \cdot V^{(k+1)} > (Z^{(k+1)})^T \cdot V^{(k)}$ is proven. Let i be the index of the largest $|v_i^{(k)}|$ such that $v_i^{(k)} \times z_i^{(k)} < 0$. Thus, we change the sign of $z_i^{(k)}$ and compute $V^{(k+1)} = diag^{=0}(\mathbf{X} \cdot \mathbf{X}^T) \cdot Z^{(k+1)}$. For the computation of $v_i^{(k+1)}$, we multiply $z_i^{(k+1)}$ with the i-th diagonal element of $\mathbf{X} \cdot \mathbf{X}^T$. Since all diagonal elements are equal to 0 we get $v_i^{(k)} = v_i^{(k+1)}$. Next, assume a unit vector U_i with the same length as $Z^{(k)}$ where the i-th element is 1 and all other elements are 0. Using U_i we compute $Z^{(k+1)}$ as follows:

$$\begin{aligned}
Z^{(k+1)} &= Z^{(k)} - 2 \times U_i \\
\mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z^{(k+1)} &= \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot(Z^{(k)} - 2 \times U_i) \\
\mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z^{(k+1)} &= \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z^{(k)} - \\
&\qquad 2 \times \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot U_i
\end{aligned}$$

We substitute $V^{(k+1)} = \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot Z^{(k+1)}$ and get:

$$\begin{aligned}
V^{(k+1)} &= V^{(k)} - 2 \times \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot U_i \\
(Z^{(k+1)})^T\cdot V^{(k+1)} &= (Z^{(k+1)})^T\cdot V^{(k)} - \\
&\qquad 2 \times (Z^{(k+1)})^T\cdot \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot U_i
\end{aligned} \tag{4.2}$$

Let $Y = (Z^{(k+1)})^T\cdot \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)$. Since we changed the sign of $z_i^{(k)}$ we have $z_i^{(k+1)} < 0$ and get $y_i < 0$. Since u_i is the only element in U_i that is equal to 1 we know that in $Y\cdot U_i$ only y_i is multiplied by 1, whereas all other elements of Y are multiplied by 0. We use $Y\cdot U_i < 0$ to get $2 \times (Z^{(k+1)})^T\cdot \mathit{diag}^{=0}(\mathbf{X}\cdot\mathbf{X}^T)\cdot U_i < 0$. From (4.2), we get

$$(Z^{(k+1)})^T\cdot V^{(k+1)} \quad > \quad (Z^{(k+1)})^T\cdot V^{(k)}. \tag{4.3}$$

Second, we show $(Z^{(k+1)})^T \cdot V^{(k)} > (Z^{(k)})^T \cdot V^{(k)}$. By changing the sign of the element in $Z^{(k)}$ that corresponds to the largest $|v_i^{(k)}|$ such that $v_i^{(k)} \times z_i^{(k)} < 0$, we get:

$$\begin{aligned}
\sum_{i=1}^{n}(z_i^{(k+1)} \times v_i^{(k)}) \quad &> \quad \sum_{i=1}^{n}(z_i^{(k)} \times v_i^{(k)}) \\
(Z^{(k+1)})^T \cdot V^{(k)} \quad &> \quad (Z^{(k)})^T \cdot V^{(k)}
\end{aligned} \tag{4.4}$$

By transitivity using (4.3) and (4.4), the following holds:

$$(Z^{(k+1)})^T \cdot V^{(k+1)} > (Z^{(k)})^T \cdot V^{(k)}.$$

□

4.5.2 Termination

Lemma 9 (Termination). *Let $\mathbf{X}$ be an $n \times m$ matrix. SSV($\mathbf{X}, n, m$) terminates and performs at most n iterations.*

Proof. We show that each element in the sign vector, $z_i \in Z$, is changed at most once. Since Z contains n elements, the algorithm performs at most n iterations.

To prove that each element in the sign vector is changed at most once, we show that any value $v_i \in V$ that increases $Z^T \cdot V$ most in one of the iterations does not change its sign in subsequent iterations, i.e., $v_i < 0$. This prevents v_i to be considered as candidate element in future iterations, since the corresponding sign z_i is changed from 1 to -1 and has the same sign as v_i. From Lemma 5 we have

$$V^{(k)} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & x_{12} & \dots & x_{1n} \\ x_{21} & 0 & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & 0 \end{bmatrix}}_{diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)} \cdot Z^{(k)}$$

We show two consecutive iterations, $k+1$ and $k+2$, and assume without loss of generality that z_1 and z_n are changed, respectively.

Let $v_1^{(k)}$ be the element that increases $Z^T \cdot V$ most at the k-th iteration and assume that $z_n^{(k)} = 1$ has not yet been changed. We have

$$v_1^{(k)} = (z_2^{(k)} \times x_{12}) + \ldots + (z_{n-1}^{(k)} \times x_{1n-1}) + x_{1n} < 0. \tag{4.5}$$

Next, consider iteration $k+1$. Let $v_i^{(k+1)} < 0$ with $i \neq 1$ be the element that increases $Z^T \cdot V$ most (if no such element exists the algorithm terminates). Without loss of generality assume that $i = n$. Then, the sign of z_n is changed from 1 to -1, i.e., $z_n^{(k+2)} = -1$. Hence, in iteration $k+2$ we get

$$v_1^{(k+2)} = (z_2^{(k+2)} \times x_{12}) + \ldots + (z_{n-1}^{(z+2)} \times x_{1n-1}) - x_{1n}. \tag{4.6}$$

Now, we perform a case analysis on the element x_{1n} and prove that $v_1^{(k+2)} < 0$ holds.

Case $x_{1n} \geq 0$ We have that $z_i^{(k)} = z_i^{(k+2)}$ for $i = 2, \ldots, n-1$, thus the sum over the first $n-1$ elements in $v_1^{(k)}$ (equation 4.5) and $v_1^{(k+1)}$ (equation 4.6) is the same. Since $x_{1n} \geq 0$ we can conclude that $v_1^{(k+2)} < v_1^{(k)} < 0$.

Case $x_{1n} < 0$ As in the previous case, let $v_1^{(k)}$ and $v_n^{(k+1)}$ be the two elements that increase $Z^T \cdot V$ most at iteration k and $k+1$, respectively. Since $diag^{=0}(\mathbf{X} \cdot \mathbf{X}^T)$ is a symmetric matrix, we have $x_{1n} = x_{n1}$. Again, we do a case analysis on x_{1n} (for simplicity, we omit z elements in the following equations):

- *Case* $x_{1n} < x_{n2} + \ldots + x_{nn-1}$: By simple transformations and $x_{1n} = x_{n1}$ we get

  $$0 < -x_{n1} + x_{n2} + \ldots + x_{nn-1} = v_n^{(k+1)}.$$

  This leads to a contradiction with our assumption that $v_n^{(k+1)}$ increases $Z^T \cdot V$ most in iteration $k+1$, hence $v_n^{(k+1)} < 0$.

- *Case* $x_{1n} \geq x_{n2} + \ldots + x_{nn-1}$: Since $v_1^{(k)}$ increases $Z^T \cdot V$ most at the k-th iteration, the following holds:

  $$v_1^{(k)} < v_n^{(k)}$$
  $$x_{12} + \ldots + x_{1n-1} + x_{1n} < x_{1n} + x_{n2} \ldots + x_{nn-1}$$
  $$x_{12} + \ldots + x_{1n-1} < x_{n2} + \ldots + x_{n-1n}$$

  We substitute the right-hand side in the above equation by our assumption and get $x_{12} + \ldots + x_{1n-1} < x_{1n}$ and further $x_{12} + \ldots + x_{1n-1} - x_{1n} = v_1^{(k+2)} < 0$.

By using a similar reasoning, we can generalize the proof for iterations k and $k+p$ with $1 \leq p < n-k$ and elements v_i and v_j, $i \neq j$, that increase $Z^T \cdot V$ most, respectively. □

4.5.3 Greedy Strategy

Lemma 10 (Local optimal choice)**.** *The SSV algorithm changes in each iteration the element of the sign vector Z that most increases $Z^T \cdot V$.*

Proof. We perform a case analysis to prove that SSV makes the local optimal choice.

Case $z_i^{(k)} \times v_i^{(k)} < 0$: The SSV algorithm changes in each iteration the sign of an element $z_i^{(k)}$ that maximizes $|z_i^{(k)} \times v_i^{(k)}|$. Since $diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T)$ is a symmetric matrix, we have

$$V^{(k)} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & x_{12} & \dots & x_{1n-1} & x_{1n} \\ x_{12} & 0 & \dots & x_{2n-1} & x_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{1n-1} & x_{2n-1} & \dots & 0 & x_{n-1n} \\ x_{1n} & x_{2n} & \dots & x_{n-1n} & 0 \end{bmatrix}}_{diag^{=0}(\mathbf{X}\cdot\mathbf{X}^T)} \cdot Z^{(k)}$$

Without loss of generality assume that $z_1^{(k)} \times v_1^{(k)} < 0$, $z_n^{(k)} \times v_n^{(k)} < 0$ such that $|v_1^{(k)}| > |v_n^{(k)}|$. Let Z_1 and Z_n be the sign vectors resulting from changing the sign of z_1 and z_n respectively. Then, we have

$$\begin{aligned} (Z_1)^T \cdot V^{(k)} &> (Z_n)^T \cdot V^{(k)} \Rightarrow \\ (Z_1)^T \cdot (diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T) \cdot Z^{(k)}) &> (Z_n)^T \cdot (diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T) \cdot Z^{(k)}) \Rightarrow \\ \sum_{i=2}^{n-1} x_{in} &> \sum_{i=2}^{n-1} x_{1i} \end{aligned} \tag{4.7}$$

Let's now suppose that we get a bigger benefit by changing the sign of z_n instead of z_1. Then, we get

$$\begin{gathered} (Z_1)^T \cdot V_1^{(k+1)} < (Z_n)^T \cdot V_n^{(k+1)} \Rightarrow \\ (Z_1)^T \cdot (diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T) \cdot Z_1) < (Z_n)^T \cdot (diag^{=0}(\mathbf{X}{\cdot}\mathbf{X}^T) \cdot Z_n) \Rightarrow \\ \sum_{i=2}^{n-1} x_{in} < \sum_{i=2}^{n-1} x_{1i} \end{gathered}$$

Which contradicts (4.7). Therefore, we get a bigger benefit by choosing the element that maximizes $|z_i^{(k)} \times v_i^{(k)}|$, i.e., z_1.

Case $z_i^{(k)} \times v_i^{(k)} \geq 0$: If instead we would change the sign of an element $z_i^{(k)}$ for which $z_i^{(k)} \times v_i^{(k)} \geq 0$ we get $Z^{(k+1)} = Z^{(k)} + 2 \times U$ (cf. Lemma 8), which implies $(Z^{(k+1)})^T \cdot V^{(k+1)} \leq$

$(Z^{(k)})^T \cdot V^{(k)}$. Therefore, $Z^T \cdot V$ will not be increased. □

4.5.4 Correctness

In this section we prove that our greedy approach computes the optimal solution.

Lemma 11 (Global maximum)**.** *The SSV algorithm computes the maximizing sign vector for which the final product $Z^T \cdot V$ is globally maximal.*

Proof. In order to prove the correctness of our greedy algorithm, we need to demonstrate that our algorithm satisfies two properties that make any greedy approach optimal (see [CLRS09] for further details).

1) Greedy Choice Property: This property states that an optimal solution exists that is consistent with the first greedy choice. We demonstrate that there exists an optimal solution which includes the first greedy choice.

Let $Z = [z_1, \ldots, z_n]$ be a sign vector and $P = \{p_1, \ldots, p_k\}$, $k \leq n$, be the ordered set of sign change positions in Z as computed by the SSV algorithm. For instance, $P = \{2, 3, 4, 1\}$ indicates that in the first iteration z_2 has been changed, in the second iteration z_3, etc. We use Z_P to refer to the sign vector where the positions in P have been flipped. Furthermore, let $P^* = \{p_1^*, \ldots, p_l^*\}$ be the ordered set of sign change positions in Z for the optimal solution. We can distinguish two cases:

- $p_1 \in P^*$: The first greedy choice is included in the optimal solution, hence the greedy choice property holds.

- $p_1 \notin P^*$: The first greedy choice is not included in the optimal solution. Without loss of generality we substitute the first element $p_1^* \in P^*$ by p_1 and get $P' = (P^* - \{p_1^*\}) \cup \{p_1\} = \{p_1, p_2^*, \ldots, p_n^*\}$. From the greedy strategy we know that $z_{p_1} \times v_{p_1} < 0$ and that p_1 is the position with the largest value, i.e., $|v_{p_1}^{(k)}| \geq |v_i^{(k)}|$ for $i = 1, \ldots, n$. Therefore, by replacing p_1^* with p_1 in P' the product $Z_{P'}^T \cdot V$ will increase, which is a contradiction to the fact that P^* are the positions of the optimal solution.

Thus, the first greedy choice is part of the optimal solution.

2) Optimal Substructure Property: This property states that solutions to subproblems of an optimal solution are also optimal. We demonstrate by contradiction that the optimal solution after a greedy choice contains an optimal solution to the remaining subproblem.

Let $P^* = \{p_1^*, \ldots, p_k^*\}$ be the ordered set of sign change positions in Z for the optimal solution and let $P' = P^* - \{p_1^*\} = \{p_2^*, \ldots, p_k^*\}$ be the ordered set of sign changes of all positions but p_1^*. Assume that P' is not optimal. Then, there exists an optimal solution P'' with $|P''| = k-1$ and $p_1^* \notin P''$ such that

$$(Z_{P''})^T \cdot V > (Z_{P'})^T \cdot V.$$

By adding the position p_1^* on both sides we obtain

$$(Z_{P'' \cup \{p_1^*\}})^T \cdot V > (Z_{P' \cup \{p_1^*\}})^T \cdot V = (Z_{P^*})^T \cdot V.$$

This contradicts our assumption that P^* produces the optimal solution. Therefore, P' is optimal.

Since the SSV algorithm satisfies the greedy choice property and the optimal substructure property, we conclude that the result of the algorithm is a global optimum. □

4.5.5 Complexity Analysis

The SSV algorithm keeps in memory the sign vector Z and V, each with $O(n)$ space complexity, where n is the number of rows in $\mathbf{X}$. Therefore, the total space complexity is $O(n)$.

The total runtime complexity is $O(xn)$, where x is the number of changed elements in the returned sign vector Z. In the worst case, the sign of each element is changed, yielding a time complexity of $O(n^2)$. The experiment in Figure 4.10 shows that the average number of sign changes in Z is $\frac{n}{2}$.

4.6 Empirical Evaluation

4.6.1 Setup

We refer to SCD and QCD as the Centroid Decomposition (CD) using respectively SSV and QSV. We implemented SCD, QCD and SVD algorithms in Java on top of an Oracle database. We connect to the database through the 11.2 JDBC driver. For the experiments the client and the database server run on the same 2.6 GHz machine with 4GB RAM.

The empirical evaluation is performed on real world datasets that describe hydrological time series[1] where each tuple records a timestamp and a value of a specific observation. The hydrological time series have been normalized with the z-score normalization technique [JNR05]. The values of the observations are stored as 4-byte floating numbers. We conducted also experiments on raw time series from the UCR repository [KZH^{+}11].

In what follows, we evaluate scalability, efficiency and correctness of our algorithm. Furthermore, we empirically determine the number of iterations performed by the SSV algorithm and we show the impact of the distribution of the sign of values across different time series on the number of iterations. For each experiment, we display the average result over five runs of the algorithms.

4.6.2 Experiments

Efficiency and Scalability

In order to evaluate the efficiency and scalability, we choose the longest time series from the UCR repository. We concatenate the time series that belong to the same dataset to get time series with the same length as the hydrological ones. Table 4.1 describes the used time series.

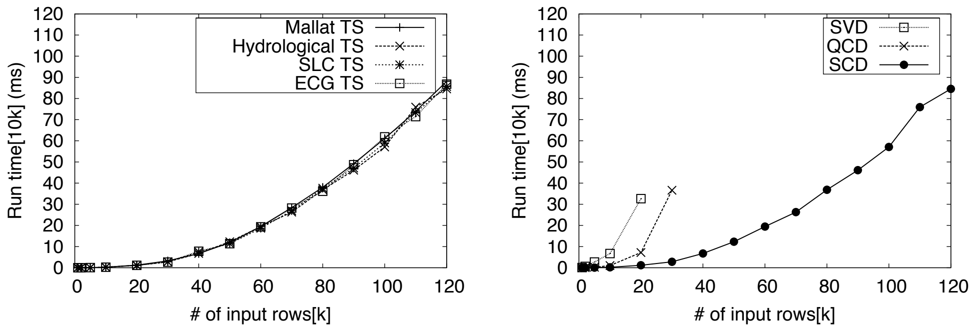
The experiment in Figure 4.5 evaluates the runtime of the SCD algorithm and compares it against other techniques. The computation of the matrix $\mathbf{X}{\cdot}\mathbf{X}^{T}$ is included in the running times of QCD. For each technique we implement the algorithm that computes the full decomposition. In Figure 4.5(a), the number m of time series is four and the number n of rows varies between

[1]The data was kindly provided by the environmental engineering company HydroloGIS (http://www.hydrologis.edu).

Table 4.1: Description of the First Set of Time Series.

Name	Provenance	Max_Length	Number TS
Hydrological TS	Hydrologis	120’000	217
MALLAT	UCR repository	2345	1024
StarLightCurves (SLC)	UCR repository	8236	1024
CinC_ECG_torso (ECG)	UCR repository	1380	1639

zero and $120k$. This experiment shows that, for all time series, SCD has quadratic runtime with respect to the number of rows of the input matrix $\mathbf{X}$. Figure 4.5(b) compares the runtime of SCD against QCD and SVD using hydrological time series. This experiment shows that SCD, QCD and SVD have quadratic runtime. The QCD algorithm runs out of memory for $n > 30k$, whereas SVD runs out of memory for $n > 20k$. In contrast, SCD performs the decomposition of a matrix that contains four time series of $120k$ observations each in less than seven minutes.



(a) Runtime of SCD using Different Time Series (b) Runtime of Different Techniques using Hydrological Time Series

Figure 4.5: Runtime by Varying n.

In the experiment in Figure 4.6, n is set to $5k$ and m varies between 20 and 100. The results show that the runtime of the SCD and QCD algorithms increases linearly with m. Using SCD, the decomposition of a matrix of 100 time series with $5k$ observations each is performed in approximately 80 seconds. We did not include SVD, which has a cubic runtime complexity with respect to m.

Figure 4.7 compares the memory usage of SCD against QCD and SVD (notice the log-scale on the y-axis). For each of the three algorithms we sum the allocated space for all data structures. The results of the calculation of memory allocation confirm the linear space complexity of SCD

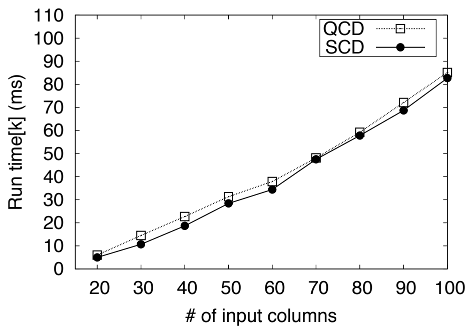


Figure 4.6: Runtime by Varying m in Hydrological Time Series.

with respect to the number of rows of the input matrix $\mathbf{X}$, whereas QCD and SVD have quadratic space complexity. For $n > 30k$ and $n > 20k$, respectively, QCD and SVD run out of memory.

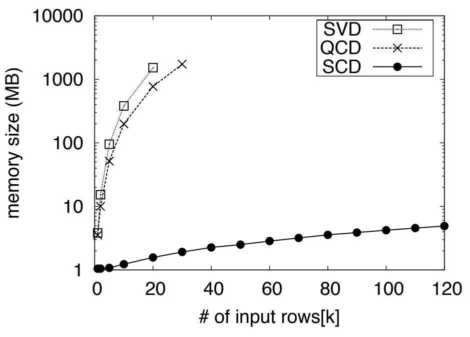


Figure 4.7: Memory Usage.

Algorithm Properties

The properties of the SSV algorithm are evaluated using the time series described in Table 4.1.

Figure 4.8 evaluates the trend of the product $(Z^{(k)})^T \cdot V^{(k)}$ computed by the SSV algorithm. This experiment confirms the monotonicity property stated in Lemma 8. We extract 1000 values from each time series and compute the product $(Z^{(k)})^T \cdot V^{(k)}$ for 20 iterations. The experiment shows that $(Z^{(k)})^T \cdot V^{(k)}$ computed by our algorithm is monotonically increasing.

In Figure 4.9, we compare the sign vectors computed by the SSV algorithm against those computed by QSV algorithm. This experiment aims to confirm the correctness property stated in Lemma 11. We compute the percentage of correct sign vectors, i.e., the sign vectors computed by SSV that are equal to those computed by QSV. As expected, the experiment confirms that

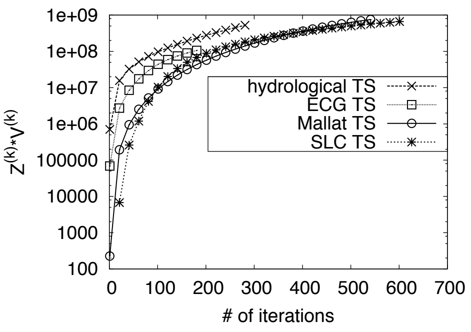


Figure 4.8: Monotonicity Property of SSV.

SSV computes the correct sign vectors in all cases.

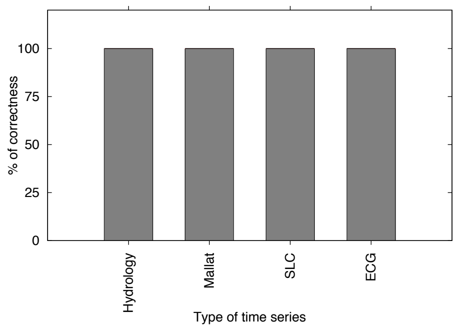


Figure 4.9: Correctness of SSV.

Number of Iterations

The time series from Table 4.1 are used. In the experiment of Figure 4.10, we show the number of iterations in the SSV algorithm that are required to compute the maximizing sign vectors. For all used time series, our algorithm performs on average $\frac{n}{2}$ iterations, i.e., it performs only half of the maximum number of iterations.

Impact of Sign of Values

We select three time series from the UCR repository in such a way that we have different distributions of negative values at the same timestamp across the selected time series. Table 4.2 describes the used time series, where the third column is the number of values with a negative sign at the same timestamp in all time series.

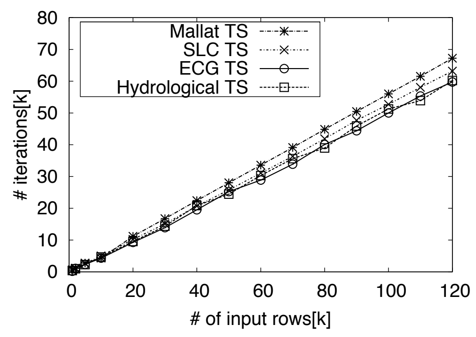


Figure 4.10: Number of Iterations in SSV.

Table 4.2: Description of the Second Set of Time Series.

Name	Provenance	length	# negative rows (x)
Gun_Point	UCR	150	90
Cricket_X	UCR	300	21
Beef	UCR	470	0

Figure 4.11 illustrates the impact of the number of negative rows (x) on the number of iterations, and hence on the runtime. In the Gun_Point dataset, x is bigger than half of the input rows. The number of iterations is on average equal to half of the input rows. In the Cricket_X dataset, x is between 1 and half of the input rows. In this case, the SSV algorithm iterates on average $x + 1$ times. In the case where all values in all time series have the same sign, the number of iterations is equal to 1 as expected. That is, if the sign of all elements is positive or negative, all elements of the weight vector computed in the first iteration of the algorithm, i.e., $V^{(1)}$, are positive. In both cases the sign vector that contains only 1s is the maximizing vector and our algorithm requires only one iteration to find the maximizing vector. Figure 4.11 shows also that the increase in the number of input rows together with a higher x implies a higher runtime. In the case where the negative sign is randomly distributed across different time series, the number of iterations is on average equal to $\frac{n}{2}$ as shown in Figure 4.10.

In Figure 4.12, we used the Beef time series that does not contain any negative value and we incrementally change the sign of 10% of input rows. This experiment shows that the runtime increases with the number of negative rows. If all rows are negative, the runtime is the same as when all rows are positive and the number of iterations performed by SSV is equal to 1.

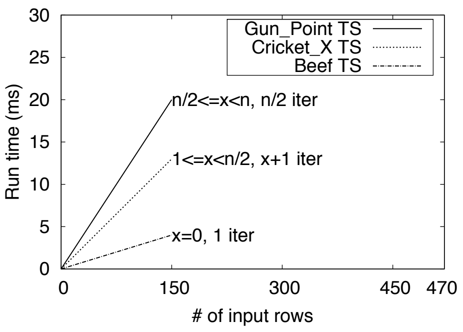


Figure 4.11: Impact of the Sign of Input Rows.

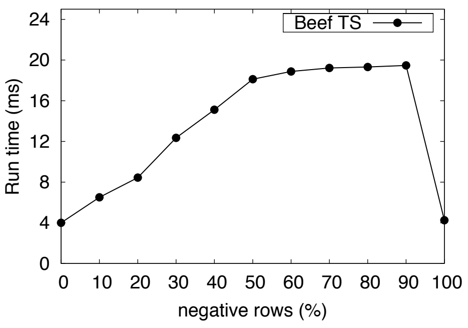
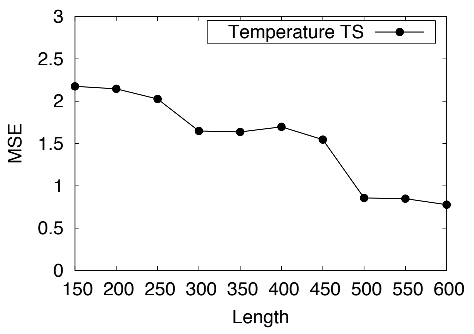


Figure 4.12: Run Time with Varying Percentage of Negative Rows.

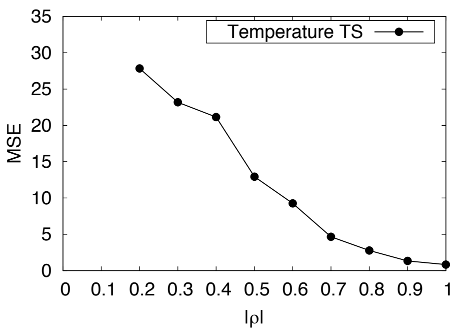
Length of Time Series

The final experiment shows the impact of using long time series for the recovery of missing values [KB12]. We remove a block of 100 values from a temperature time series and recover the removed block by an iterative computation of matrices $\mathbf{L}$ and $\mathbf{R}$. The input matrix contains as columns the time series with the removed block and three other temperature time series. Then, we perform a one rank reduction, i.e., we compute only three vectors in the matrices $\mathbf{L}$ and $\mathbf{R}$ instead of four, and we iterate until the difference in the Frobenius norm [MKW94] between the matrix before the decomposition and the one after the decomposition is less than 10^{-5}. To measure the accuracy, we compute the Mean Square Error ($MSE = \frac{1}{k}\sum_{i=1}^{k}(\tilde{x}_i - x_i)^2$; original value x_i; recovered value $\tilde{x}_i$; number of observations k) between the original and the recovered blocks [LMPF09], [KHQ01].

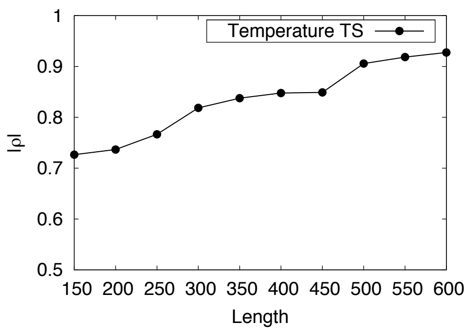
Figure 4.13 shows the result of this experiment. In Figure 4.13(a), the length of the time series is varied. Longer time series significantly reduce the MSE. In Figure 4.13(b), we take different time series of 200 values such that the absolute value of the Pearson correlation ρ between the



(a) Length of TS vs MSE



(b) Correlation vs MSE



(c) Length of TS vs Correlation

Figure 4.13: Impact of the Length of the Time Series and the Correlation Between Them on the Recovery of Missing Values.

time series with the missing block and the other time series varies, and we compute the MSE. The experiment shows that the more correlated the time series are the better is the recovery. Finally, Figure 4.13(c) shows that the correlation between the time series increases with the length of time series (the same time series as in Figure 4.13(a) are used).

4.7 Conclusion and Future Work

In this paper, we introduced the Scalable Sign Vector algorithm that performs the Centroid Decomposition of a matrix in linear space complexity. We provided proofs that show the scalability, the termination and the correctness of our algorithm. An empirical evaluation on real world hydrological data sets and also on data sets from the UCR repository demonstrates that our algorithm has the same runtime as the most efficient algorithm to compute the Centroid Decomposition, but reduces the space complexity from quadratic to linear.

In future work, we plan to investigate an incremental version of the Centroid Decomposition that could be applied for dynamic time series. Another promising direction is to investigate segmentation techniques of time series.

CHAPTER 5

Conclusion and Future Work

In this thesis, we propose a parameter-free solution, called REBOM, that accurately recovers blocks of missing values in irregular time series. REBOM learns the shape, the amplitude and the width of the missing blocks from the time series that contains the missing blocks and its correlated time series. Our approach is based on a greedy strategy that iteratively selects from a set of time series the ones that locally reduce the recovery error. We empirically show that the recovery result is independent from the initialization of the missing values. We implemented REBOM as a graphical tool that currently performs the recovery of blocks of missing values using real world hydrological time series, but can also be applied for any type of correlated time series.

We introduce a Centroid Decomposition based recovery technique that produces an accurate block recovery when using time series with mixed correlations. The proposed solution reduces the recovery error to the used time series proportionally to their correlation. We compare the decomposition process performed by the Centroid Decomposition against the one performed by the Singular Value Decomposition that is used by REBOM. Our experimental results show that for similar number of lowly and highly correlated time series, the recovery accuracy of the CD based approach outperforms the one of REBOM.

We propose a scalable implementation of the Centroid Decomposition technique. We show that by mapping the optimization problem into an equivalent problem, we avoid the construction of the square correlation matrix and thus, reduce the space complexity from quadratic to linear. We prove that the proposed greedy approach computes the correct result and terminates. We run extensive experiments on real world hydrological time series to support the scalability and the correctness of the proposed solution.

Future Work We currently limit the block recovery to static time series. It is of interest to apply the block recovery on streams of time series where the new data could be used to improve former block recoveries. Recomputing the matrix decomposition from scratch once new data (eventually a block of data) arrives, will be inefficient. An incremental computation will be investigated.

Currently, we assume that the lowly correlated time series that exhibit shape and/or trend similarities are given as input and we choose them using their graphical similarity. In order to automate the selection of input time series we plan to refine the definition of lowly correlated time series by investigating a measure that detects lowly time series with shape and/or trend similarity.

Another promising direction is to distribute the computation of the Centroid Decomposition technique for domains with long time series of fine-grained granularity, e,g,. finance. A straightforward distribution of the computation is inefficient due to the lineage of weight vectors that has to be stored. An efficient distribution of the computation requires the parallelization of the optimization problem.

Bibliography

[ABB00] Orly Alter, Patrick O. Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, August 2000.

[AM07] Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *J. ACM*, 54(2), 2007.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[Bjö96] Äke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[BL07] James Bennett and Stan Lanning. The netflix prize. *KDD cup and workshop*, 2007.

[BNR10] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. *CoRR*, abs/1006.4046, 2010.

[Bra02] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings, Part I*, pages 707–720, 2002.

[Bra03] Matthew Brand. Fast online SVD revisions for lightweight recommender systems. In *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 37–46, 2003.

[CCL+07] Qiuxia Chen, Lei Chen, Xiang Lian, Yunhao Liu, and Jeffrey Xu Yu. Indexable PLA for efficient similarity search. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 435–446, 2007.

[CF01] Moody T. Chu and Robert E. Funderlic. The centroid decomposition: Relationships between discrete variational decompositions and svds. *SIAM J. Matrix Anal. Appl.*, 23(4):1025–1044, 2001.

[CFG95] Moody T. Chu, Robert E. Funderlic, and Gene H. Golub. A rank-one reduction formula and its applications to matrix factorizations. *SIAM Review*, 37(4):512–530, 1995.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[DM01] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.

[DS10] Yufeng Ding and Jeffrey S. Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *Journal of Machine Learning Research*, 11:131–170, 2010.

[DTS+08] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.

[GH07] Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*, volume Analytical methods for social research. Cambridge University Press, New York, 2007.

[GNHS11] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 69–77, 2011.

[GS06] Vaughan D. Griffiths and Ian-Moffat Smith. *Numerical methods for engineers*. CRC press, 2006.

[GVL96] Gene .H. Golub and Charles F. Van-Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[HCRC11] Morgan Harvey, Mark James Carman, Ian Ruthven, and Fabio Crestani. Bayesian latent variable models for collaborative item rating prediction. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 699–708, 2011.

[He06] Yan He. *Missing Data Imputation for Tree-Based Models*. PhD thesis, University of California, 2006.

[HKP11] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, Third Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[Hyd12] HydroloGIS. http://www.hydrologis.eu/, 2012. Accessed September 14, 2012.

[IBM13] IBM. Ibm informix, version 12.10. http://www-01.ibm.com/software/data/informix/timeseries/, 2013.

[INF13] INFLUXdb. Influxdb project. http://influxdb.com/, 2013.

[JCW04] Ankur Jain, Edward Y. Chang, and Yuan-Fang Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 11–22, 2004.

[JNR05] Anil K. Jain, Karthik Nandakumar, and Arun Ross. Score normalization in multimodal biometric systems. *Pattern Recognition*, 38(12):2270–2285, 2005.

[KAES99] Kothuri Venkata Ravi Kanth, Divyakant Agrawal, Amr El Abbadi, and Ambuj K. Singh. Dimensionality reduction for similarity searching in dynamic databases. *Computer Vision and Image Understanding*, 75(1-2):59–72, 1999.

[Kal96] Dan Kalman. A singularly valuable decomposition: The svd of a matrix. *The College Mathematics Journal*, 27(1):pp. 2–23, 1996.

[KB09] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[KB12] Mourad Khayati and Michael H. Böhlen. REBOM: recovery of blocks of missing values in time series. In *Proceedings of the 18th International Conference on Management of Data, COMAD 2012, 2012, Pune, India*, pages 44–55, 2012.

[KBG14] Mourad Khayati, H. Michael Böhlen, and Johann Gamper. Memory-efficient centroid decomposition for long time series. In *ICDE*, pages 100–111, 2014.

[KBV09] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[KHQ01] Fredrik Kahl, Anders Heyden, and Long Quan. Minimal projective reconstruction including missing data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):418–424, 2001.

[KO98] Tamara G. Kolda and Dianne P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Trans. Inf. Syst.*, 16(4):322–346, 1998.

[KO00] Tamara G. Kolda. and Dianne P. O'Leary. Algorithm 805: computation and uses of the semidiscrete matrix decomposition. *ACM Trans. Math. Softw.*, 26(3):415–435, 2000.

[Kon05] Erricos J. Kontoghiorghes, editor. *Handbook on Parallel Computing and Statistics*. Chapman and Hall/CRC, Philadelphia, PA, USA, 2005.

[KT98] Kosmas Karadimitriou and John M. Tyler. The centroid method for compressing sets of similar images. *Pattern Recognition Letters*, 19(7):585–593, 1998.

[KZH$^+$11] E. Keogh, Q. Zhu, B. Hu, Hao. Y., X. Xi, L. Wei, and C.A. Ratanamahatana. The UCR Time Series Classification/Clustering, homepage: www.cs.ucr.edu/~eamonn/time_series_data/, 2011.

[Lag91] Jeffrey C. Lagarias. Monotonicity properties of the toda flow, the qr-flow, and subspace iteration. *SIAM J. Matrix Anal. Appl.*, 12(3):449–462, June 1991.

[LBKL15] Mu Li, Wei Bi, James T. Kwok, and Bao-Liang Lu. Large-scale nyström kernel matrix approximation using randomized SVD. *IEEE Trans. Neural Netw. Learning Syst.*, 26(1):152–164, 2015.

[LMP10] Willis Lang, Michael D. Morse, and Jignesh M. Patel. Dictionary-based compression for long time-series similarity. *IEEE Trans. Knowl. Data Eng.*, 22(11):1609–1622, 2010.

[LMPF09] Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. Dynammo: mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 507–516, 2009.

[LTS13] Boduo Li, Sandeep Tata, and Yannis Sismanis. Sparkler: supporting large-scale matrix factorization. In *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 625–636, 2013.

[MAL$^+$03] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces, January 12-15, 2003, Miami, FL, USA*, pages 263–266, 2003.

[Mey00] Carl D. Meyer, editor. *Matrix analysis and applied linear algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[MKT07] András Benczúr Miklós Kurucz and Balázs Torma. Methods for large scale svd with missing values. In *KDDCup 2007*, 2007.

[MKW94] Changxue Ma, Yves Kamp, and Lei F. Willems. A frobenius norm approach to glottal closure detection from the speech signal. *IEEE Transactions on Speech and Audio Processing*, 2(2):258–265, 1994.

[MM09] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 627–636, 2009.

[MNL10] Abdullah Mueen, Suman Nath, and Jie Liu. Fast approximate correlation for massive time-series data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 171–182, 2010.

[PFS12] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I*, pages 521–536, 2012.

[PJR03] Haesun Park, Moongu Jeon, and J-Ben Rosen. Lower dimensional representation of text data based on centroids and least squares. *BIT*, 43:427–448, 2003.

[PRO12] PROmetheus. Prometheus project. http://prometheus.io/, 2012.

[Ren10] Steffen Rendle. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000, 2010.

[Ren13] Steffen Rendle. Scaling factorization machines to relational data. *PVLDB*, 6(5):337–348, 2013.

[RPBA09] Umaa Rebbapragada, Pavlos Protopapas, Carla E. Brodley, and Charles R. Alcock. Finding anomalous periodic time series. *Machine Learning*, 74(3):281–313, 2009.

[RS04] Vanessa Romero and Antonio Salmeron. *Multivariate imputation of qualitative missing data using Bayesian networks.*, pages 605–612. Berlin: Springer, 2004.

[SAP10] SAP. Sap time management. https://help.sap.com/saphelp_scm70/helpdata/en/86/4b0c00ebac41a8ac49137f73775ec5/content.htm, 2010.

[SJ03] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 720–727, 2003.

[SK98] Thomas Seidl and Hans-Peter Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 154–165, 1998.

[Ski07] David Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007.

[STP07] Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *J. Mach. Learn. Res.*, 8:1623–1657, December 2007.

[SZB$^+$11] John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 2011.

[TB97] Llyod N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[TSD10] TSDB. Open tsdb, version 2.0.1. http://opentsdb.net/, 2010.

[WR97] Patrick Waldemar and Tor A. Ramstad. Hybrid KLT-SVD image compression. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '97, Munich, Germany, April 21-24, 1997*, pages 2713–2716, 1997.

[XRZ$^+$13] Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. Shark: SQL and rich analytics at scale. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 13–24, 2013.

[YHSD12] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pages 765–774, 2012.

[YJL04] Jieping Ye, Ravi Janardan, and Qi Li. GPCA: an efficient dimension reduction scheme for image compression and retrieval. In *Proceedings of the Tenth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004, pages 354–363, 2004.

[YSJ$^+$00] Byoung-Kee Yi, Nikolaos Sidiropoulos, Theodore Johnson, H. V. Jagadish, Christos Faloutsos, and Alexandros Biliris. Online data mining for co-evolving time sequences. In *ICDE*, pages 13–22, 2000.

[ZWSL10] Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 2595–2603, 2010.