Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Saeed VARASTEH YAZDI

Thèse dirigée par **Ahlame DOUZAL**

préparée au sein du **Laboratoire Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Représentations parcimonieuses et apprentissage de dictionnaires pour la classification et le clustering de séries temporelles

Time warp invariant sparse coding and dictionary learning for time series classification and clustering

Thèse soutenue publiquement le **15 novembre 2018**,
devant le jury composé de :

Monsieur PHILIPPE PREUX
PROFESSEUR, UNIVERSITE DE LILLE, Rapporteur
Monsieur MOHAMED NADIF
PROFESSEUR, UNIVERSITE PARIS 5, Rapporteur
Monsieur STEPHANE CANU
PROFESSEUR, NORMANDIE UNIVERSITE, Examinateur
Monsieur PATRICK GALLINARI
PROFESSEUR, SORBONNE UNIVERSITES - PARIS, Président
Monsieur JULIEN MAIRAL
CHARGE DE RECHERCHE, INRIA CENTRE DE GRENOBLE RHÔNE-ALPES, Examinateur
Madame AHLAME DOUZAL
PROFESSEUR ASSOCIE, UNIVERSITE GRENOBLE ALPES, Directeur de thèse

Final version Printed on January 25, 2019

Acknowledgements

First, I would like to thank my thesis advisor Prof. Ahlame Douzal for putting her trust in me and inviting me to do my PhD in France. I thank her for her insightful guidance and support. She has been always extremely generous with her time at every stage of my study. I feel grateful to work with her.

I am also thankful to my jury members. I thank Prof. Preux and Prof. Nadif for reviewing my thesis and providing me with useful comments and suggestions. I also thank Prof. Gallinari, Prof. Canu and Dr. Mairal for participating to the committee of my PhD defense. I thank them all for their valuable time and suggestions.

I would also like to thank all the members of the AMA team from LIG laboratory for their help and support in these three years.

Finally, I would like to thank my family for their unconditional support. My mother (Narges), my father (Mohammad Ali) and my sister (Samaneh). They have been my source of inspiration and they have always courage me for better education. I also thank my wife (Golbarg) for her constant support, encouragement, help and understanding during my PhD study.

Abstract

Learning dictionary for sparse representing time series is an important issue to extract latent temporal features, reveal salient primitives and sparsely represent complex temporal data. This thesis addresses the sparse coding and dictionary learning problem for time series classification and clustering under time warp. For that, we propose a time warp invariant sparse coding and dictionary learning framework where both input samples and atoms define time series of different lengths that involve varying delays.

In the first part, we formalize an l_0 sparse coding problem and propose a time warp invariant orthogonal matching pursuit based on a new cosine maximization time warp operator. For the dictionary learning stage, a non linear time warp invariant kSVD (TWI-kSVD) is proposed. Thanks to a rotation transformation between each atom and its sibling atoms, a singular value decomposition is used to jointly approximate the coefficients and update the dictionary, similar to the standard kSVD. In the second part, a time warp invariant dictionary learning for time series clustering is formalized and a gradient descent solution is proposed.

The proposed methods are confronted to major shift invariant, convolved and kernel dictionary learning methods on several public and real temporal data. The conducted experiments show the potential of the proposed frameworks to efficiently sparse represent, classify and cluster time series under time warp.

Résumé

L'apprentissage de dictionnaires à partir de données temporelles est un problème fondamental pour l'extraction de caractéristiques temporelles latentes, la révélation de primitives saillantes et la représentation de données temporelles complexes. Cette thèse porte sur l'apprentissage de dictionnaires pour la représentation parcimonieuse de séries temporelles. On s'intéresse à l'apprentissage de représentations pour la reconstruction, la classification et le clustering de séries temporelles sous des transformations de distortions temporelles. Nous proposons de nouveaux modèles invariants aux distortions temporelles.

La première partie du travail porte sur l'apprentissage de dictionnaire pour des tâches de reconstruction et de classification de séries temporelles. Nous avons proposé un modèle TWI-OMP (Time-Warp Invariant Orthogonal Matching Pursuit) invariant aux distorsions temporelles, basé sur un opérateur de maximisation du cosinus entre des séries temporelles. Nous avons ensuite introduit le concept d'atomes jumelés (sibling atomes) et avons proposé une approche d'apprentissage de dictionnaires TWI-kSVD étendant la méthode kSVD à des séries temporelles.

Dans la seconde partie du travail, nous nous sommes intéressés à l'apprentissage de dictionnaires pour le clustering de séries temporelles. Nous avons proposé une formalisation du problème et une solution TWI-DLCLUST par descente de gradient.

Les modèles proposés sont évalués au travers plusieurs jeux de données publiques et réelles puis comparés aux approches majeures de l'état de l'art. Les expériences conduites et les résultats obtenus montrent l'intérêt des modèles d'apprentissage de représentations proposés pour la classification et le clustering de séries temporelles.

List of Publications

The following publications are included in parts or in an extended version in this thesis [Yazdi 18a, Yazdi 18b]:

- Saeed Varasteh Yazdi and Ahlame Douzal-Chouakria. Time warp invariant ksvd: Sparse coding and dictionary learning for time series under time warp. *Pattern Recognition Letters 112, 1–8 (2018).* https://doi.org/10.1016/j.patrec.2018.05.017
- Saeed Varasteh Yazdi, Ahlame Douzal-Chouakria, Patrick Gallinari and Manuel Moussallam. Time Warp Invariant Dictionary Learning for Time Series Clustering: Application to Music Data Stream Analysis. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2018).* https://doi.org/10.1007/978-3-030-10925-7_22

In addition to the topics studied in this manuscript which are mentioned above, during my thesis I worked on several other problems of time series analysis leading to the following publications [Do 17, Yuan 18]:

- Jidong Yuan, Ahlame Douzal-Chouakria, Saeed Varasteh Yazdi and Zhihai Wang. A large margin time series nearest neighbour classification under locally weighted time warps. *Knowledge and Information Systems, (2018).* https://doi.org/10.1007/s10115-018-1184-z
- Cao-Tri Do, Ahlame Douzal-Chouakria, Sylvain Marié, Michèle Rombaut and Saeed Varasteh Yazdi. Multi-modal and multi-scale temporal metric learning for a robust time series nearest neighbors classification. *Information Sciences 418, 272-285 (2017).* https://doi.org/10.1016/j.ins.2017.08.020

Contents

List of Figures

List of Tables

List of Acronyms

ARIMA	Auto Regressive Integrated Moving Average
DWT	Discrete Wavelet Transform
DFT	Discrete Fourier Transform
ECG	Electrocardiography
PAA	Piece-wise Aggregate Approximation
SAX	Symbolic Aggregate Approximation
SR	Sparse Representation
SC	Sparse Coding
GSC	Group Sparse Coding
DL	Dictionary Learning
i.i.d.	Independent and Identically Distributed
MP	Matching Pursuit
OMP	Orthogonal Matching Pursuit
GOMP	Group Orthogonal Matching Pursuit
KOMP	Kernel Orthogonal Matching Pursuit
BPDN	Basis Pursuit De-Noising
LASSO	Least Angle Shrinkage and Selection Operator
SVD	Singular Value Decomposition
SVM	Support Vector Machine
DTW	Dynamic Time Warping
LLC	Locally constraint Liner Coding
SRC	Sparse Representation Based Classification
SSC	Sparse Subspace Clustering
SSC-BP	l_1 Sparse Subspace Clustering
SSC-OMP	l_0 Sparse Subspace Clustering
MOD	Method of Optimal Direction
FDDL	Fisher Discrimination Dictionary Learning
DLSI	Dictionary learning with Structured Incoherence
MOCOD	Method of Optimal Coherence constrained Directions
SISC	Shift Invariant Sparse Coding
KGA	General Alignment Kernel
κ-kSVD	Kernel kSVD
D-kSVD	Discriminative kSVD

LC-kSVD . . . . Label Consistent kSVD
COSTW . . . . COSine maximization Time Warp
TWI-OMP . . . Time Warp Invariant Orthogonal Matching Pursuit
TWI-kSVD . . . Time Warp Invariant kSVD
TWI-GDDL . . Time Warp Invariant Gradient Descent Dictionary Learning
TWI- DLCLUST Time Warp Invariant Dictionary Learning Clustering
UCI . . . . . University of California Invine
ANN . . . . . Artificial Neural Network
CNN . . . . . Convolutional Neural Network
RNN . . . . . Recurrent Neural Network
LSTM . . . . . Long Short-Term Memory

1 Introduction

Temporal data are part of many real-world data, especially in emerging applications such as sensor networks or internet of things. The fact that temporal data are growing in many areas as in medical, industry, entertainment or financial domains induce a large amount of interest in querying and analyzing these data in the last decade.

Time series are a type of temporal data. A time series is a collection of observations, each one being recorded at different points in time. While static data are commonly given in an N by p matrix where each row is a sample presented by a vector of p variables, one time series instance can be presented in a T by p matrix where T is the number of time stamps and p is the number of variables (Figure 1.1). The data points taken over time have internal dependencies (correlations) which is contrary to the the assumption of static data where the observations are independent and identically distributed. The time delay that presents naturally in the series also creates a huge challenge for many time series analysis. In addition, since these data are mostly produced from the output of the sensors, they often come in high dimension and usually are noisy.

The special characteristics of time series are commonly addressed in time series analysis. The natural temporal ordering and time delay give time series some unique properties that prevent researchers to apply conventional machine learning techniques for them. Temporal data representation are widely studied in

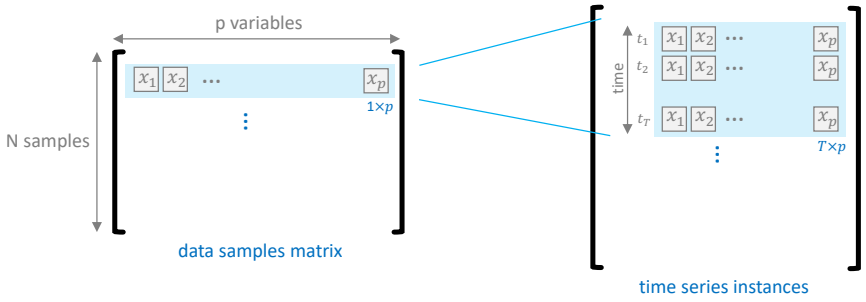

Figure 1.1: Static data vs. time series data.

literature. The objective is to reduce the data dimensionality and highlight the relevant information of the input time series to facilitate the further computations. Several approaches motivated by various domains have been investigated. The first group of approaches are from statistics or signal processing where the objective is to project time series into a new space defined by static descriptors. The new description is generally based on estimating some model parameters such as auto regressive models [Lütkepohl 05, Hamilton 94, Box 15], Hidden Markov Model [Rabiner 86, Juang 91], Fourier model [Agrawal 93], Wavelet model [Chan 99] or polynomial models [Keogh 01]. The discrete Fourier transform (DFT) that used in [Agrawal 93], for example, projects time series on sine and cosine function basis in the real domain where the resulting representation is a set of sinusoidal coefficients. To have a multiresolution decomposition, in [Chan 99] a set of scaled and shifted wavelet functions were used. Although such models are effective for data dimensionality reduction, valuable temporal information might be lost by applying such transformations. The second group is the segmentation methods, which unlike the previous approaches, preserve the temporal dimension of the data. A segmentation process consist in dividing a time series into a set of segments by moving a sliding-window over it. Inside a segment, the data can be represented by one specific value or a continuous function. The segmentation techniques have multiple objectives as reducing the temporal dimension of data, extracting the sub-sequences or salient patterns and filtering the noisy data. Several methods proposed based on the various segmentation methods and the ways to represent data within a segment [Guralnik 99, Fu 06, Fancourt 98, Chu 95]. For example, in [Stiefmeier 07] a set of polynomial coefficients is used to represent each segment or in [Lin 03] a symbolic representation has been used.

The described time series can be then analyzed by conventional approaches dedicated to static data. However, noisy and complex real-world temporal data can not be easily described by a set of model parameters. Despite the popularity and usage of these representation methods, such feature extraction techniques suffer from several drawbacks. Firstly, the temporal semantic of data are ignored after applying such transformation. Secondly, finding a suitable set of features to represent the data and capture the relevant information is not straightforward and depending on the task and the data. Expert knowledge may also be needed to define relevant domain specific features for a given task. In the other hand, to better model complex real-world data, instead of heavy crafting on data processing and feature engineering, the data representation learning methods such as neural networks, dictionary learning or deep learning can be applied to automatically discover the representation that is particularly interesting for a given task.

In recent years, sparse representation (SR) or sparse coding (SC) and dictionary learning (DL) has been widely used in many successful applications of machine learning and signal processing as denoising, compression or segmentation [Elad 06, Bryt 08], feature extraction [Guha 12, Ramirez 10], reconstruction [Aharon 06], classification [Wei 13a, Wright 09] or in computer vision for face recognition, activity recognition and action recognition [Wright 09, Mairal 08, Jiang 11, Guha 12]. Sparse coding is basically representing an input sample using a predefined set of basis on the assumption that these basis have sufficiently enough information to reconstruct the input. The basis are like directions in the input space and can be selected from the input samples itself or learned through dictionary learning methods. Dictionary learning algorithms are focused on developing dictionaries of atoms that provides efficient representations for signals. The learned atoms are like pseudo data points in the input space where the samples can then be represented as a linear combination of the neighboring atoms.

Despite the immense research efforts on dictionary learning in many domains, their performance on temporal data are less investigated. The major challenges are probably related to the nature of the time series that we discussed earlier. Challenging time series often involve varying delays through time and the salient events may be related to a part of observations that may appear at different time stamps. For example, the Electrocardiography (ECG) of a patient that records the electrical activity of the heart over a period of time, often comes with different

delays and lengths from one sample to another, or in classifying the ECG data, time series of the two different classes describe nearly the same global behaviour, whereas a small part of the observations has enough information to predict their class.

For temporal data analysis, sparse coding and dictionary learning are especially effective to extract class specific latent temporal features, reveal salient primitives and sparsely represent complex temporal features. The objective of this thesis is to explore the potential of these methods in reconstruction and representation of time series. We are looking for a space that reveals the latent temporal structure of data where the learned directions are defined atoms for temporal data and learned to reconstruct and represent the input time series. In addition, we intend to investigate the ability of the new representation for time series classification and clustering.

For that, we propose a non linear time warp invariant dictionary learning framework where both input samples and atoms define time series that may have different lengths and involve varying delays. That is, for the sparse coding, we propose a time warp invariant orthogonal matching pursuit (TWI-OMP) based on a new cosine maximization time warp operator (COSTW) and the induced sibling atoms. For the dictionary learning, we proposed two efficient solutions. First, thanks to a rotation transformation between each atom and its sibling atoms, a singular value decomposition is used to jointly approximate the coefficients and update the dictionary similar to the popular kSVD [Aharon 06] method. We also developed a gradient descent solution for the dictionary learning under time warp problem. The potential of the proposed methods is confronted to major alternative approaches on several character and digit handwritten trajectories, that define naturally time series of different lengths that include varying delays.

The major contributions of this thesis are summarized as follows:

1. We investigate the problem of sparse coding and dictionary learning for time series data and expose the weakness of the conventional methods to deal with the temporal data.
2. We propose a tractable solution for the sparse coding of temporal data and suggest a time warp invariant orthogonal matching pursuit based on a new cosine maximization time warp operator to sparse codes time series under time warp invariances.

3. We generalize the popular kSVD dictionary learning method to learn the temporal dictionary atoms for time series. We also develop a gradient descend solution for updating the temporal dictionary atoms.
4. We develop a time series classification schema based on the proposed sparse representation method.
5. We propose a time series clustering approach under sparse coding and dictionary learning setting and we provide a sparse representation of the clustered time series and learn, for each cluster, a sub-dictionary composed of the most discriminative primitives.
6. We conduct several experiments on several public and real datasets to compare the proposed approaches to the major alternative approaches in both classification and clustering settings.
7. We further inspect an application of music listening data streams on the data captured from the well known DEEZER platform.

Thesis outline

The remainder of this thesis is organized as follows. In Chapter 2, the state of the art sparse coding and dictionary learning formalization and the major solutions on both static and temporal data are reviewed. In Chapter 3, the proposed time warp invariant sparse coding and dictionary learning problem and the solutions for time series sparse representation under time warp are presented. The related classification and clustering applications are also expressed. Chapter 4 provides the conducted experiments and discussed the results obtained. Finally, Chapter 5 concludes this thesis and points out the future perspectives.

Notations

X	a set of samples / time series
$\boldsymbol{x}_i$	a vector sample, a time series
x_{ij}	j^{th} element (time stamp) of $\boldsymbol{x}_i$
D	a dictionary
$\boldsymbol{d}_j$	a dictionary atom
$\boldsymbol{\alpha}$	sparse coefficient vector
A	sparse coefficients matrix
$\boldsymbol{\alpha}_{j.}$	j^{th} row of the matrix A
τ	sparsity level
λ	regularization parameter
Δ	alignment matrix
X_c	a set of samples belonging to class c
N	number of data in a set
C	number of classes
K	number of dictionary atoms or number of clusters
$\|\boldsymbol{x}\|_q$	q norm of the vector $\boldsymbol{x}$
$\boldsymbol{\pi}$	an alignment between two time series
C_l	representative of a cluster
D_l	sub dictionary
D_G	global dictionary consists of all the sub dictionaries
Ψ	a set of basis functions
$\boldsymbol{\psi}$	a basis function (atom)
$\mathcal{K}$	gram matrix (kernel matrix)
ϕ	embedding function to the the Hilbert space
$\varphi(.), \gamma_i(.)$	transformation functions

Life is essentially an endless series of problems. The solution to one problem is merely the creation of another.

— Mark Manson

2

Sparse Representation and Dictionary Learning

Contents

2.1 Introduction

Given a set of input samples, the aim of sparse coding methods is to represent each input sample as a linear combination of few elements taken from a set of representative patterns. These representative patterns are called *atoms* or *basis*, the set of patterns is called *dictionary* and the coefficient vector of this linear combination is referred as *sparse code*. Sparse coding problem is formalized basically as an optimization problem that minimizes the error of the reconstruction under l_0 or l_1 sparsity constraint. The l_0 constraint, that controls the maximum number of involved atoms, leads to a non convex and NP-hard problem. This problem can however be solved efficiently by a pursuit algorithm such as matching pursuit (MP) [Mallat 93] or orthogonal matching pursuit (OMP) [Tropp 04, Pati 93]. Relaxing the sparsity constraint from l_0 to l_1 norm yields a convex optimization problem, also known as LASSO [Tibshirani 96] problem.

The dictionary for the sparse representation can be selected among pre-specified family of basis functions (e.g., Gabors, Wavelets, Curvelets, etc.). Although these dictionaries allow fast transforms, their reconstruction potential is tightly related to the nature of the data. For instance, Wavelets show efficient reconstruction for natural images and textures [Ophir 11], Curvelets for edges [Candes 00] and Gabor for sounds [Chu 09]. The alternative to taking a family of basis functions is to use a dictionary learning approach that aims to learn an optimal dictionary from the input data to minimize the reconstruction error and the sparsity [Engan 99, Aharon 06, Mailhé 12]. A two-step optimization strategy is commonly used to learn such dictionaries: 1) keep the dictionary fixed and find the sparse representation using a sparse approximation algorithm, e.g., OMP, 2) keep the representation fixed and update the dictionary, either all the atoms at once as in MOD [Engan 99] or one atom at a time as in kSVD [Aharon 06].

The organization of this chapter is as follows. I begin by introducing some of the most well-known unsupervised and supervised sparse coding and dictionary learning methods; we focus on those that are more related to this work. Then, in the last section, I reviewed the temporal data dictionary learning approaches including shift invariant and kernel dictionary learning.

2.2 Sparse representation

Let $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K] \in \mathbb{R}^{q \times K}$ be a real matrix whose columns have unit Euclidean norm: $\|\boldsymbol{d}_j\|_2 = 1$ for $j = 1, 2, ..., K$. We refer to this matrix as a *dictionary* and the columns $(\boldsymbol{d}_j)$ as *atoms*. Given a data sample $\boldsymbol{x} \in \mathbb{R}^q$, the objective is to best represent it as a linear combination of a few atoms of D. Ideally, we want to solve the following linear system

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \;\; s.t. \;\; D\boldsymbol{\alpha} = \boldsymbol{x}. \tag{2.1}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^K$ is the sparse representation coefficient vector of the input $\boldsymbol{x}$ with respect to the dictionary D. The function $\|.\|_0 : \mathbb{R}^K \to \mathbb{R}$ simply returns the number of non-zero entries in its argument. In practice, with the presence of noise, the sparse solution of the problem 2.1 is alternatively formulated by allowing some reconstruction error $\epsilon \geq 0$, as

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \;\; s.t. \;\; \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2 \leq \epsilon. \tag{2.2}$$

This will be equivalent to the following optimization problem, where we seek the minimal error possible at a given level of sparsity $\tau \geq 1$

$$\min_{\boldsymbol{\alpha}} \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2 \;\; s.t. \;\; \|\boldsymbol{\alpha}\|_0 \leq \tau. \tag{2.3}$$

Problem 2.3 is called τ-*sparse* optimization problem. Unfortunately, solving the l_0 norm is not convex and makes the above problem NP-hard (Computing the optimal solution involves searching over all the possible $\binom{K}{\tau}$ combinations). Two approaches may be used to approximate 2.3: greedy pursuit algorithms [Tropp 04] and relaxation algorithms [Tibshirani 96]. In the latter, the l_0 norm is replaced by its nearest convex surrogate the l_1 norm, yielding convex optimization problems that admit tractable algorithms. Researchers in signal processing and machine leaning use two different versions of the l_1 norm minimization problem. In signal processing, the following form commonly used and is known as Basis Pursuit de-noising (BPDN) [Chen 01]

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \;\; s.t. \;\; \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2 \leq \epsilon. \tag{2.4}$$

Whereas in machine learning, the Least Angle Shrinkage and Selection Operator (LASSO) [Tibshirani 96] formulation is more common.

$$\min_{\boldsymbol{\alpha}} \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2 \;\; s.t. \;\; \|\boldsymbol{\alpha}\|_1 \leq \epsilon. \tag{2.5}$$

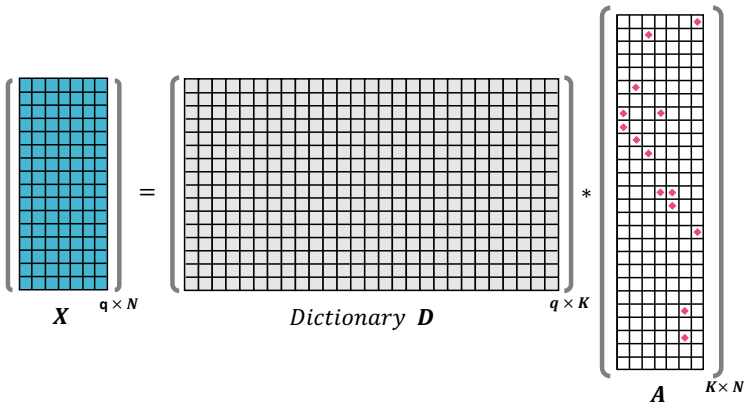

Figure 2.1: Sparse representation framework

The LASSO minimizes the reconstruction error and puts a restriction on the l_1 norm rather than minimizing the l_1 norm like BPDN. Furthermore, according to the Lagrange multiplier theorem, a constant λ exists such that problems 2.4 and 2.5 are equivalent to the following unconstrained minimization problem:

$$\min_{\boldsymbol{\alpha}} \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 \tag{2.6}$$

where the regularization parameter λ balances the trade-off between reconstruction error and the sparsity of $\boldsymbol{\alpha}$.

Another alternative to solve the problem 2.3 is greedy pursuit methods. Pursuit methods iteratively refine the current estimation of the sparse code vector $\boldsymbol{\alpha}$ by modifying one or several coefficients chosen to yield an improvement in the input approximation. Solving the l_1 norm minimization problem is computationally expensive, whereas greedy approximation algorithms are computationally fast and they have been found to be quite accurate.

It is also worth pointing out another sparse representation formalization called *group sparse coding* (GSC) [Bengio 09] which assumes that the inputs can be approximated by a union of a few subspaces where now an input is encouraged to be sparsely represented by a similar set of the dictionary atoms. Given the dictionary D, D_l denotes the sub-dictionary belongs to the group l with columns from D where the atoms in D_l span a subspace. The sparse representation optimization with the group sparsity constraint can be written as

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_{2,0} \;\; s.t. \;\; \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2 \leq \epsilon. \tag{2.7}$$

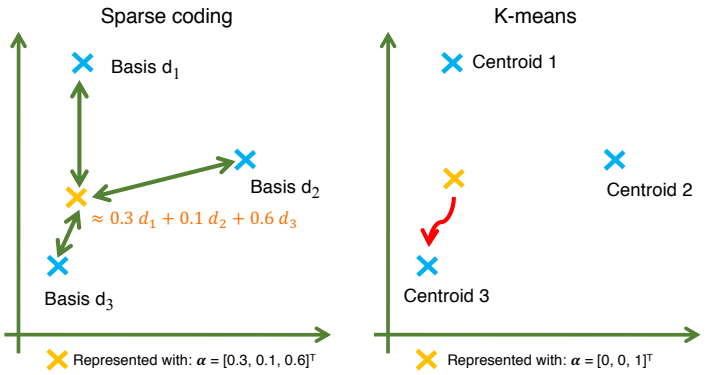

Figure 2.2: Vector quantization versus the sparse coding. The K-means algorithm can be seen as a particular case of the sparse coding, when only one atom of the dictionary is allowed in representing the input $\boldsymbol{x}$ and the corresponding coefficient must be one.

where the mixed norm defined as $\|\boldsymbol{\alpha}\|_{2,0} = \sum_l \mathcal{I}(\|\boldsymbol{\alpha}^l\|_2)$ and $\mathcal{I}(.)$ is an indicator function and defined as:

$$\mathcal{I}(\|\boldsymbol{\alpha}^l\|_2) = \begin{cases} 1, & if \|\boldsymbol{\alpha}^l\|_2 > 0 \\ 0, & otherwise. \end{cases}$$

with $\boldsymbol{\alpha}^l$ be the entries of $\boldsymbol{\alpha}$ associated to the group l.

Group Orthogonal Matching Pursuit (GOMP) [Majumdar 09] is proposed to solve the above sparse representation problem. A convex relaxation of Eq. 2.7 is also proposed where the $l_{2,0}$ norm replaced with $\|\boldsymbol{\alpha}\|_{2,1} = \sum_l \|\boldsymbol{\alpha}^l\|_2$. The corresponding problem can be transferred as an unconstrained optimization problem and solved by the group LASSO [Yuan 06].

2.2.1 From sparse coding to vector quantization

There is a strong relation between sparse representation and *Vector Quantization* (clustering). In clustering, a set of descriptive vectors (centroids) is learned, and each sample is represented by one of those vectors, usually the one closest to it. For example, the K-means algorithm finds a set of cluster centroids $\{\boldsymbol{d}_j\}_{j=1}^K$ and cluster assignments $\{\boldsymbol{\alpha}_i\}_{i=1}^N \in \mathbb{R}^K$ that minimize the distance between data points $\{\boldsymbol{x}_i\}_{i=1}^N \in \mathbb{R}^q$ and the closest center (usually under l_2 norm distance). In the sparse coding setting, the K-means algorithm can be formulated to solve the following optimization problem:

$$\min_{A,D} \sum_{i=1}^{N} \|\boldsymbol{x}_i - D\boldsymbol{\alpha}_i\|_2^2 \;\; s.t. \;\; \|\boldsymbol{\alpha}_i\|_0 = 1, \|\boldsymbol{\alpha}_i\|_1 = 1, \;\; \forall i = 1, 2, ..., N. \tag{2.8}$$

where $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, .., \boldsymbol{d}_K]$ with $\boldsymbol{d}_j \in \mathbb{R}^q$ are the K cluster centroids to be found and $A = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N]$ with $\boldsymbol{\alpha}_i \in \mathbb{R}^K$ are cluster membership indicators. The notation $\|\boldsymbol{\alpha}_i\|_0$ counts the number of non-zero entries in $\boldsymbol{\alpha}_i$ and $\|\boldsymbol{\alpha}_i\|_1$ is the l_1 norm of $\boldsymbol{\alpha}_i$ which is the summation of the absolute values of each elements in $\boldsymbol{\alpha}_i$. The constraints $\|\boldsymbol{\alpha}_i\|_0 = 1, \|\boldsymbol{\alpha}_i\|_1 = 1$ ensure that there is only one element of $\boldsymbol{\alpha}_i$ taking value 1 and all others being 0. After the optimization, the index of this non-zero element indicates which cluster the vector $\boldsymbol{x}_i$ belongs to. This problem can be seen as a restrictive case of sparse coding in the sense that only one dictionary atom is allowed to participate in the construction of the input and the related coefficient must be one.

In the following sub sections, we briefly present some of the major sparse representation solver and the algorithms.

2.2.2 Matching Pursuit (MP)

Matching Pursuit (MP) [Mallat 93] is one of the earliest methods for sparse approximation. It is an iterative greedy algorithm that looks for the sparse approximation of the input by sequentially selecting dictionary atoms. Given the input sample $\boldsymbol{x}$ and dictionary D, the main idea is to select at each iteration the atom $\boldsymbol{d}_j$ that is highly correlated to the the input or to its residual part. The correlation is measured as the inner product of the input sample and the dictionary column $\boldsymbol{d}_j$:

$$\boldsymbol{d}_j = \underset{\boldsymbol{d}_j \in D}{\arg\max} < \boldsymbol{x}, \boldsymbol{d}_j > \tag{2.9}$$

The corresponding coefficient (α_j) is then updated by this inner product and the residual approximation error is given by:

$$\boldsymbol{r} = \boldsymbol{x} - < \boldsymbol{x}, \boldsymbol{d}_j > \boldsymbol{d}_j \tag{2.10}$$

The process is repeated until τ columns are selected. Note that MP only updates the corresponding coefficient to the last selected atom and other coefficients are not modified. In this projection approach, as the dictionary atoms are not all orthogonal to each other, a dictionary atom could be selected more than once in different iterations, which reduces the performance of the algorithm.

Algorithm 1 Orthogonal Matching Pursuit (OMP)

Input: $\boldsymbol{x}$, D, τ.
Output: $\boldsymbol{\alpha}$.
1: $\boldsymbol{r} = \boldsymbol{x}, \Omega = \{\emptyset\}$.
2: **while** $|\Omega| \leq \tau$ **do**
3: Select the atom $\boldsymbol{d}_j$ ($j \notin \Omega$) that maximizes $|\frac{\boldsymbol{r}^T \boldsymbol{d}_j}{\|\boldsymbol{r}\|_2 \|\boldsymbol{d}_j\|_2}|$
4: Update the set of selected atoms: $\Omega = \Omega \cup \{j\}$
5: Update the coefficients: $\boldsymbol{\alpha}_\Omega = (D_\Omega^T D_\Omega)^{-1}(D_\Omega^T \boldsymbol{x})$, ($D_\Omega$ *is the sub-dictionary of the selected atoms and* $\boldsymbol{\alpha}_\Omega$ *is the related coefficients*)
6: Estimate the residual: $\boldsymbol{r} = \boldsymbol{x} - D_\Omega \boldsymbol{\alpha}_\Omega$
7: **end while**

2.2.3 Orthogonal Matching Pursuit (OMP)

Orthogonal Matching Pursuit (OMP) [Tropp 04, Pati 93] has been proposed as an improvement of matching pursuit. The algorithm is akin to MP in the sense of selecting one atom at each iteration. The process described in Algorithm 1 starts with setting the residual vector $\boldsymbol{r}$ to be equal to the input sample $\boldsymbol{x}$ (step 1). Then, at each iteration, the column of D that is strongly correlated with the residual vector is selected (step 3). Next, the coefficients α_j are obtained by an orthogonal projection on the subspace defined by the yet selected atoms (step 5). Note that, the α_j value represents the contribution of $\boldsymbol{d}_j$ to reconstruct $\boldsymbol{x}$. This estimation requires the solution of a least square problem. The process is reiterated until the maximum number τ of atoms is reached.

Unlike other sparse coding methods, the number of non zero entries in OMP can easily be fixed a priori, which makes it an appealing algorithm. Note that, the coefficients are updated by computing orthogonal projection of the input or the residual onto the subspace spanned by the atoms selected from all previous iterations. This result in improvement of the convergence rate compare to the MP method. Algorithm 1 gives the main steps of the OMP method.

2.2.4 Sparse representation for classification

The main objective of the sparse representation is to obtain a suitable reconstruction of the input sample using as few dictionary atoms as possible. To benefit the sparse representation framework for classification, several methods have been proposed.

A group of methods try to generate discriminative sparse codes by replacing the sparse coding objective function. The obtained sparse codes are then used to learn a classifier. Huang and Aviyente [Huang 06] proposed a representation which includes a discriminative term based on the Fisher's criterion (inner and between class variances), into the objective function. This results in discriminative sparse vectors of different input classes. A Support Vector Machine (SVM) classifier is then trained on the learned sparse code vectors and used for classification of the new samples.

Wang et al. [Wang 10] proposed a similar sparse coding schema, called Locally constraint Liner Coding (LLC), to generate discriminative sparse codes. They replaced the sparse coding regularization in Eq. 2.6 by a locality adapter and introduce the following optimization problem:

$$\min_{\boldsymbol{\alpha}} \|D\boldsymbol{\alpha} - \boldsymbol{x}\|_2^2 + \lambda \|p \odot \boldsymbol{\alpha}\|_2^2 \;\; s.t. \;\; \mathbf{1}^T \boldsymbol{\alpha} = 1. \tag{2.11}$$

where the symbol $\odot$ denotes element-wise multiplication, and p is a vector with the same size of $\boldsymbol{\alpha}$ which the j^{th} entry indicates the distance between $\boldsymbol{x}$ and the j^{th} atom of D. The LLC goal is to generate similar sparse codes for the closer samples. The learned sparse codes can then be used in training a classifier. Note that the LLC sparse codes are sparse in a sense that they have only a few significant values. In practice, they simply threshold the small coefficients to be zero.

Sparse representation based classification (SRC)

Another supervised sparse coding approach was proposed by Wright et al. [Wright 09]. The method called SRC (Sparse Representation based Classification). It achieved a great success in face recognition in terms of high classification accuracy and handling the problem of face occlusion. Let C be the number of the input classes. The dictionary in SRC is represented as $D = \{X_1, ..X_c, .., X_C\}$, where $X_c \in \mathbb{R}^{q \times N_c}$ is the subset of all N_c training samples of the class c. Note that, in SRC the dictionary is composed from the original training data. All atoms of D are normalized to have a unit l_2 norm. Denote by $\boldsymbol{y} \in \mathbb{R}^q$ a test sample, first the method finds its sparse representation based on D via l_1 norm minimization similar to Eq. 2.4:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \;\; s.t. \;\; \|D\boldsymbol{\alpha} - \boldsymbol{y}\|_2 \leq \epsilon. \tag{2.12}$$

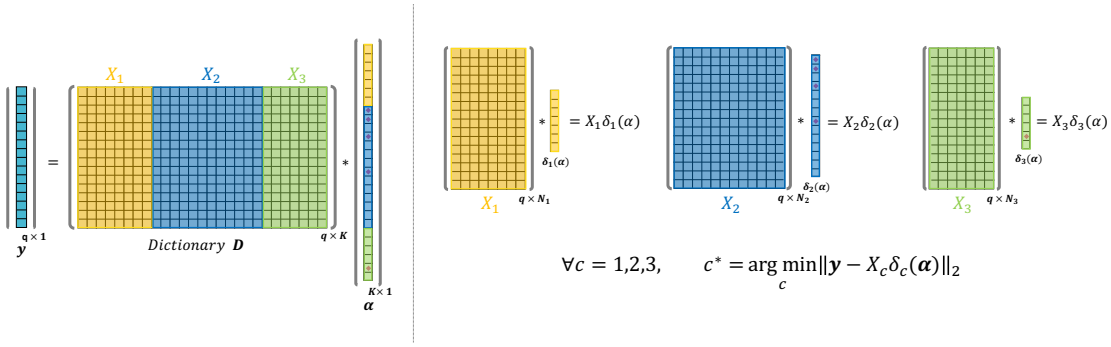

Figure 2.3: The sparse representation based classification (SRC) framework

Once sparse coding performed, classification will be based on the minimum class-wise reconstruction error. i.e., assign $\boldsymbol{y}$ to the c^{th} class such that:

$$c^* = \arg\min_c \|\boldsymbol{y} - X_c\delta_c(\boldsymbol{\alpha})\|_2^2 \tag{2.13}$$

where $\delta_c(.)$ is a function that extracts the elements corresponding to the c^{th} class. i.e., $\delta_c(\boldsymbol{\alpha}) \in \mathbb{R}^{N_c \times 1}$ is the entries of $\boldsymbol{\alpha}$ associated with the atoms of the class c. SRC is rely on the philosophy that with "sufficient" training samples, a test sample $\boldsymbol{y}$ belongs to the class c, should be presented in the column space of X_c. Beside the success of SRC in face recognition, it can be applied to any pattern recognition task. Note that, with a small training set size, SRC seems computationally efficient, however, the large number of training data may decrease its performance in terms of time and accuracy, all the more for the noisy training samples.

The sparse coding stage of SRC may utilizes any l_0 or l_1 norm spare representation algorithms that we mentioned in the previous section. A variant of SRC is proposed by Majumdar and Ward [Majumdar 09]. It uses the Group OMP (GOMP) for the sparse coding stage. The idea is approximating a test sample by a linear combination of "all" the training samples of one class to improve the classification performance of the SRC. As the dictionary provides the corresponding classes of the atoms (atom's group), in the sparse coding stage, a sample is encouraged to be represented by a group of the dictionary atoms. The GOMP algorithm is similar to the OMP algorithm (Algorithm 1) except for step 4. In this step, instead of updating the set of selected atoms by only adding the index with the highest correlation ($\Omega = \Omega \cup \{j\}$) the indices of the entire group containing the highest correlation are chosen ($\Omega = \Omega \cup \{group(j)\}$).

2.2.5 Sparse representation for clustering

In the previous section we discussed how classification can be done under the sparse representation framework. In the context of classification, the discriminative sparse codes are generally learned from the labeled dictionary (composed from the training data) and then used for the classification. In clustering setting, we distinguish a category of approaches that assumes the data structured into a union of subspaces [Elhamifar 13, You 16, Bradley 00, Tseng 00] where each sample may be represented as a linear combination of the other input samples, ideally belonging to the same subspace. Several of these approaches are related to the sparse subspace clustering (SSC) approach [Elhamifar 13, You 16, Li 17] which is first introduced by Elhamifar and Vidal [Elhamifar 13] to solve the motion segmentation problem.

SSC is based on the so-called *self-expressiveness* property of the data points. The idea of SSC is to first sparse code the input samples based on the other samples as dictionary using l_1 [Elhamifar 13] or l_0 [You 16] minimization problem. Once the sparse representation for each sample obtained, the segmentation of the data is found by applying a clustering method (e.g., spectral clustering [Ng 02]) to the affinity matrix formed based on the sparse coefficient vectors. The number of subspaces as well as their dimension may be fixed beforehand or induced from the affinity graph. SSC requires solving N optimization problems over N data points, which is computationally expensive with the large number of data and not suitable for fast online clustering. Chong et al. [You 16] proposed a variant of the SSC method based on the l_0 (OMP) solver that brings up the scalability of the SSC.

Let $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in \mathbb{R}^{q \times N}$ be the matrix describing the N input samples $\boldsymbol{x}_i \in \mathbb{R}^q$. To cluster these N data points into K clusters. First, each sample $\boldsymbol{x}_i$ is sparse represented by a linear combination of all the other columns of X as:

$$\forall i \quad \min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_1 \;\; s.t. \;\; \|X\boldsymbol{\alpha}_i - \boldsymbol{x}_i\|_2 \leq \epsilon, \;\; \alpha_{ii} = 0. \tag{2.14}$$

where $\boldsymbol{\alpha}_i$ is the sparse code of $\boldsymbol{x}_i$ with respect to X and the constraint $\alpha_{ii} = 0$ makes sure that the i^{th} entry of $\boldsymbol{\alpha}_i$ being zero, which avoids the trivial solution of representing a point by itself. The main assumption here is that, there exists a solution $\boldsymbol{\alpha}_i$, where the nonzero entries correspond to data points from the same subspace as $\boldsymbol{x}_i$.

Once the sparse representation for each data point $\boldsymbol{x}_i$ is found (Eq. 2.14), spectral clustering method can be applied on the graph affinity matrix $W = |A| + |A|^T$ to

Algorithm 2 Sparse subspace clustering (SSC)

Input: $X = \{\boldsymbol{x}_i\}_{i=1}^N$, the number of clusters K.
Output: Clustering partitions $\{C_1, ..., C_K\}$.
1: Find $\boldsymbol{\alpha}_i$, $i = 1, ..., N$ based on Eq. 2.14 and set $A = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N]$.
2: Compute the affinity matrix $W = |A| + |A|^T$
3: Compute the normalized Laplacian matrix L of W.
4: Find the k largest eigenvectors of L to form the columns of the new matrix $U = [\boldsymbol{u}_1, ..., \boldsymbol{u}_k] \in \mathbb{R}^{N \times k}$.
5: For $i = 1, ..., N$, let $\boldsymbol{y}_i \in \mathbb{R}^k$ be the vector corresponding to the the i^{th} row of U.
6: Cluster $\{\boldsymbol{y}_i\}_{i=1}^N \in \mathbb{R}^k$ with the K-means algorithm into clusters $\{C_1, ..., C_K\}$.
7: Return $\{C_1, ..., C_K\}$, segmentation of data in X.

obtain the clustering segmentation of the data in X, where $A = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N]$ is the sparse coefficient matrix. This choice of W seems suitable, hence most of the nonzero entries of the coefficient vector is found to be related to the points from the same subspace. The symmetrization also helps to connect the data points i to j when either data point $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$ is used for sparse representation of another. The SSC method is summarized in Algorithm 2. In the following we briefly explain the spectral clustering procedure.

Spectral clustering is a popular clustering method, that aims to find the partitioning of data points by solving an eigenvalue problem of the affinity matrix. Let S be the similarity matrix whose ij entry shows the similarity between points i and j. Generally, Gaussian kernel with Euclidean distance is used to build the similarity matrix. Given the matrix S, the next step is to form an affinity matrix L. Typical choices for L are the similarity matrix itself $L = S$, the unnormalized graph Laplacian $L = \mathbb{D} - A$, with $\mathbb{D} = diag(S\mathbf{1})$ and $\mathbf{1}$ be the vector of all 1's, or its normalized version $L_N = \mathbb{D}^{-1/2} L \mathbb{D}^{-1/2}$. Once the affinity matrix computed, the clustering is become a graph partitioning problem, where connected graph components are interpreted as clusters. The segmentation is then found by applying the K-means algorithm to the K top or bottom eigenvectors of the matrix L or L_N. Note that, defining a good affinity matrix is a challenge for spectral clustering algorithm. In an ideal similarity matrix, hence similarity graph, nodes within the same cluster (subspace) must have high connection values and there should be no edges connecting different nodes correspond to different subspaces.

2.3 Dictionary learning

In the previous section, we studied the problem of sparse coding. The sparse problem is that we are given a set of basis (dictionary D), an input $\boldsymbol{x}$ and we want to find the representation of $\boldsymbol{x}$ based on a few elements of D. Depending on the application (e.g., denoising, compression, reconstruction) and the nature of the data, the dictionary for the sparse representation could be different. A set of pre-defined basis such as Fourier, Gabor or Wavelets are commonly used. The ability of such fixed basis atoms is different from one application to another. For example, Wavelets show efficient reconstruction for natural images and textures [Ophir 11], Curvelets for edges [Candes 00] and Gabor for sounds [Chu 09]. Thus, dictionary selection is crucial to attain a good representation model. To achieve an improved reconstruction or representation, one solution is to learn the dictionary. In this process, the dictionary may be initialized with the common basis functions or from the data itself. Such learned dictionaries from the data have more potential than the former ones. In the following we present the dictionary learning problem formalization and some of the most used solutions.

Let $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in \mathbb{R}^{q \times N}$ be the training set, with $\boldsymbol{x}_i \in \mathbb{R}^q$ be a data sample. One can learn a dictionary $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K] \in \mathbb{R}^{q \times K}$ by solving the following optimization problem which is a generalization of the sparse coding problem given in Eq. 2.3 when both sparse codes A and dictionary D are learned to minimize the error of reconstruction of the input samples in X.

$$\begin{aligned} &\min_{A,D} \sum_{i=1}^{N} \|D\boldsymbol{\alpha}_i - \boldsymbol{x}_i\|_2 \\ &s.t. \quad \forall i \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau, \quad \forall j \ \ \|\boldsymbol{d}_j\|_2 = 1 \end{aligned} \tag{2.15}$$

where $A = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N]$ with $\boldsymbol{\alpha}_i \in \mathbb{R}^K$ the sparse code of $\boldsymbol{x}_i$ and the last constraint ensure for each column of the dictionary $\boldsymbol{d}_j \in \mathbb{R}^q$ to have unit l_2 norm.

The optimization problem in Eq. 2.15 is not convex w.r.t both A and D. Typically, a block-coordinate-descent method can be used for solving it by alternating two phases: 1) keep the dictionary fixed and find the sparse representation using a sparse approximation algorithm, for example, OMP, l_1 minimization or iterative

thresholding, 2) keep the sparse codes fixed and update the dictionary which leads to the following optimization problem:

$$\min_{D} \sum_{i=1}^{N} \|D\boldsymbol{\alpha}_i - \boldsymbol{x}_i\|_2 \;\; s.t. \;\; \forall j \;\; \|\boldsymbol{d}_j\|_2 = 1. \tag{2.16}$$

which is equivalent to:

$$\min_{D} \mathcal{J}(D) = \|X - DA\|_F^2 \;\; s.t. \;\; \forall j \;\; \|\boldsymbol{d}_j\|_2 = 1. \tag{2.17}$$

where the notation $\|.\|_F$ stands for the Frobenius norm of a matrix. The sparse coding and dictionary update processes are iterated until convergence. The differences between the existing dictionary learning algorithms that have been proposed are in the method used for the sparse approximation and the procedure used for updating the dictionary (i.e., solve 2.17). We focus in the following on several commonly used dictionary learning methods reported in the literature.

2.3.1 Dictionary learning by MOD (Method of Optimal Direction)

Method of Optimal Direction (MOD) [Engan 99] is a least square based method that directly computes the dictionary that minimizes the overall representation mean square error $\|X - DA\|_F^2$ when the coefficients are fixed. By setting the derivative of error formulation with respect to D as zero, $-2(X - DA)A^T = 0$, the result is given by a pseudo-inverse:

$$\begin{aligned} D &\leftarrow XA^T.(AA^T)^{-1} \\ \forall j \;\; \boldsymbol{d}_j &\leftarrow \frac{\boldsymbol{d}_j}{\|\boldsymbol{d}_j\|_2} \end{aligned} \tag{2.18}$$

The MOD is an effective method and usually converged in a few iterations. However, the solution requires a matrix inversion at each step which increases the complexity with a large number of training data. Several works that lead to more efficient results [Rubinstein 10] have been proposed as well as an online and recursive implementation of the MOD, with the focus on reducing its complexity.

2.3.2 Dictionary learning by descend gradient

Another idea to solve the dictionary learning problem in Eq. 2.17 by using a gradient descend approach to update each atom of the dictionary sequentially is as follows:

$$\begin{aligned} \forall j \;\; \boldsymbol{d}_j^{(m+1)} &\leftarrow \boldsymbol{d}_j^{(m)} - \frac{\eta^m}{2} \frac{\partial \mathcal{J}}{\partial \boldsymbol{d}_j^{(m)}} \\ \forall j \;\; \boldsymbol{d}_j^{m+1} &\leftarrow \frac{\boldsymbol{d}_j^{m+1}}{\|\boldsymbol{d}_j^{m+1}\|_2} \end{aligned} \tag{2.19}$$

where the gradient of the cost function in Eq. 2.17 with respect to the atom $\boldsymbol{d}_j$ is:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{d}_j} = -2(X - DA).\boldsymbol{\alpha}_{j.}^T$$

with $\boldsymbol{\alpha}_{j.}$ be the j^{th} row of A and η^m be the learning rate at iteration m which can either be fixed as in [Olshausen 96] or adaptive as in [Mailhé 12]. Mailhe and Plumbley [Mailhé 12] proposed a method to set an optimal step size for the gradient descent method to avoid the local minima. It is shown that the optimal step of the descent is proportional to the inverse of the coefficient vector energy ($\eta^* \propto \frac{1}{\|\boldsymbol{\alpha}_{j.}\|_2^2}$).

2.3.3 Dictionary learning by kSVD

Among dictionary learning methods, kSVD algorithm proposed by Aharon et al. [Aharon 06], which generalizes the K-means clustering, has become more popular and widely used. It is also an iterative approach. After the sparse approximation step with OMP, in the second step, the atoms are updated sequentially by using a singular value decomposition (SVD). At this step, based on the learned A, the objective is to update one atom and its related coefficients at a time; This results in faster convergence compare to the previous approaches that only the dictionary is getting updated.

Let us denote $\boldsymbol{\alpha}_{j.} = (\alpha_{j1}, ..., \alpha_{jN})$ the j^{th} row of A; it provides the contributions of the atom $\boldsymbol{d}_j$ to reconstruct the N input samples. To update a given atom $\boldsymbol{d}_k$, the objective function given in Eq. (2.17) can be formulated as:

$$\begin{aligned} &\|X - DA\|_F^2 = \|X - \sum_{j=1}^{K} \boldsymbol{d}_j \, \boldsymbol{\alpha}_{j.}\|_F^2 \\ &\|(X - \sum_{j \neq k} \boldsymbol{d}_j \boldsymbol{\alpha}_{j.}) - \boldsymbol{d}_k \boldsymbol{\alpha}_{k.}\|_F^2 = \|E_k - \boldsymbol{d}_k \boldsymbol{\alpha}_{k.}\|_F^2 \end{aligned} \tag{2.20}$$

Algorithm 3 kSVD dictionary learning

Input: X, D, τ.
Output: D, A.
repeat
{Sparse coding step:}
  for $i = 1, ..., N$ **do**
    $\boldsymbol{\alpha}_i =$OMP$(\boldsymbol{x}_i, D, \tau)$
  end for
{Dictionary update step:}
  for $k = 1, ..., K$ **do**
    Compute $E_k = X - \sum_{j \neq k} \boldsymbol{d}_j \boldsymbol{\alpha}_{j.}$
    $\Omega_k = \{i \,/\, \alpha_{ki} \neq 0, \;\; i = 1, ..., N\}$
    Define $E_k^{\Omega_k}$ as the restriction of E_k to Ω_k
    Apply an SVD on $E_k^{\Omega_k} = U \Sigma V^T$
    Update $\boldsymbol{d}_k = \boldsymbol{u}_1$ and $\boldsymbol{\alpha}_{k.}^{\Omega_k} = \sigma_1 \boldsymbol{v}_1^T$
  end for
until Convergence (stopping rule)

where in Eq. (2.20) DA is expressed as the sum of K rank-1 matrices, each one giving the sparse representation of X involving one atom. The matrix $E_k \in \mathbb{R}^{q \times N}$ stands for the error of reconstruction for the N samples excluding the k^{th} atom.

An SVD rank-1 approximation on E_k can be used to find $\boldsymbol{d}_k$ and $\boldsymbol{\alpha}_{k.}$, however, the new $\boldsymbol{\alpha}_{k.}$ may not be sparse anymore. To preserve the sparsity of $\boldsymbol{\alpha}_{k.}$, the residual matrix E_k is limited to only samples that involve atom $\boldsymbol{d}_k$. Let us note $E_k^{\Omega_k} \in \mathbb{R}^{q \times |\Omega_k|}$ such restricted residual matrix and $\boldsymbol{\alpha}_{k.}^{\Omega_k}$ its related coefficients with Ω_k the set of the input sample indices that involve the atom $\boldsymbol{d}_k$. Subsequently, SVD is used to estimate the closest rank-1 matrix that approximates $E_k^{\Omega_k}$ and the first left singular vector ($\boldsymbol{u}_1$), singular value (σ_1) and right singular vector ($\boldsymbol{v}_1$) are then used to update the atom $\boldsymbol{d}_k$ and its related coefficients $\boldsymbol{\alpha}_{k.}$. Algorithm 3 summarizes the main steps of the kSVD approach.

What kSVD do and that is the key innovation of this algorithm is to remove an atom from the dictionary and then replacing it with the error vector that represents the major direction within which it is not getting good representation. In other words, it collects all the error vectors into a matrix and take the SVD of that matrix. The SVD is giving in the columns of matrix U the vectors oriented along the major axis of the variance of the original data. These vectors will be sorted

thus the very first column of U will defiantly be oriented in the main direction of the variance and it is probably the best choice for replacing the atom which is removed from the dictionary.

The kSVD algorithm guaranteed to reduce or keep unchanged the reconstruction error. The stopping rule in Algorithm 3 could be the stabilization of the reconstruction error. The number of iterations until convergence depends on the application and on the initial dictionary [Aharon 06]. The computational complexity of kSVD is computed considering the two stages. When using the OMP, the whole sparse coding stage required $\mathcal{O}(qN\tau K)$ operations. The dictionary learning stage results a total of $\mathcal{O}(qN\tau)$ operations. Combining the two stages, the entire kSVD algorithm requiems $\mathcal{O}(qN\tau K)$ operations for each iteration.

2.3.4 Dictionary learning for classification

In the standard dictionary learning framework, the learned dictionary is only used for data reconstruction and is not optimal for classification. In order to use dictionary learning in classification, several approaches have been developed to learn a discriminative classification oriented dictionary by using the label information. These approaches can be divided into two groups. The first group are in the same spirit of SRC with a simple modification which is, instead of using the original training samples as dictionary, they learn separate and/or joint dictionary from each class of the data, then all these dictionaries are put together to form a global dictionary and SRC is applied [Yang 10, Yang 11, Wang 14, Wei 13b, Ramirez 10, Mairal 08]. The second group in the other hand, attempt to add a classification term into the dictionary learning formalization and learn a joint optimization problem [Zhang 10, Jiang 11, Pham 08, Mairal 09]. The first group causes the dictionary to be discriminative and uses the representation error for the classification, while, the second group forces the sparse coefficient vectors discriminative to use them as new feature vectors for classification. In the following we briefly introduce few of these methods.

Meta-face learning

Yang et al. proposed a method called Meta-face learning [Yang 10] to learn a compact and robust dictionary on the training data of each class. Although it was originally proposed for classification of face images, it can be used in any application. Denote by C the number of classes and $X_c \in \mathbb{R}^{q \times N_c}$ the subset of all the N_c training samples from class c, a class-specific dictionary D_c is learned from the samples in X_c. The dictionary learning formalization can be presented as:

$$\forall c \quad \min_{A_c, D_c} \|X_c - D_c A_c\|_F^2 + \lambda \sum_{i=1}^{N} \|\boldsymbol{\alpha}_i^c\|_1 \;\; s.t. \;\; \forall j \;\; \|\boldsymbol{d}_j^c\|_2^2 = 1. \tag{2.21}$$

where A_c is the sparse coefficients matrix for class c, $\boldsymbol{\alpha}_i^c$ the sparse code of the i^{th} sample of the class c and $\boldsymbol{d}_j^c$ the j^{th} column of the c^{th} class-specific sub-dictionary. Like other dictionary learning problem, they solve Eq. 2.21 by optimizing A_c and D_c alternatively. After finding the sparse representations A_c with fixed D_c, each dictionary column is updated at a time in a similar fashion as kSVD, while using a Lagrangian solver instead of SVD. With a simplified notation ($D = D_c$ and $A = A_c$), the following procedure is used to update atom $\boldsymbol{d}_k$:

$$\begin{aligned}
\mathcal{J}(\boldsymbol{d}_k) &= \arg\min_{\boldsymbol{d}_k} \|X - DA\|_F^2 \;\; s.t \;\; \boldsymbol{d}_k^T \boldsymbol{d}_k = 1. \\
&= \arg\min_{\boldsymbol{d}_k} \|X - \sum_j \boldsymbol{d}_j \boldsymbol{\alpha}_{j.}\|_F^2 \;\; s.t \;\; \boldsymbol{d}_k^T \boldsymbol{d}_k = 1. \\
&= \arg\min_{\boldsymbol{d}_k} \|(X - \sum_{j \neq k} \boldsymbol{d}_j \boldsymbol{\alpha}_{j.}) - \boldsymbol{d}_k \boldsymbol{\alpha}_{k.}\|_F^2 \;\; s.t \;\; \boldsymbol{d}_k^T \boldsymbol{d}_k = 1. \\
&= \arg\min_{\boldsymbol{d}_k} \|E_k - \boldsymbol{d}_k \boldsymbol{\alpha}_{k.}\|_F^2 \;\; s.t \;\; \boldsymbol{d}_k^T \boldsymbol{d}_k = 1.
\end{aligned} \tag{2.22}$$

Using a Lagrange multiplier, Eq. 2.22 is equivalent to:

$$\mathcal{J}(\boldsymbol{d}_k, \gamma) = \arg\min_{\boldsymbol{d}_k} tr(-E_k \boldsymbol{\alpha}_{k.}^T \boldsymbol{d}_k^T - \boldsymbol{d}_k (\boldsymbol{\alpha}_{k.} E_k^T) + \boldsymbol{d}_k (\boldsymbol{\alpha}_{k.} \boldsymbol{\alpha}_{k.}^T - \gamma) \boldsymbol{d}_k^T + \gamma) \tag{2.23}$$

where γ is a scalar variable. Taking the derivative of $\mathcal{J}(\boldsymbol{d}_k, \gamma)$ with respect to $\boldsymbol{d}_k$ and set it to zero, we obtain the following relation to update $\boldsymbol{d}_k$:

$$\tilde{\boldsymbol{d}}_k = \frac{E_k \boldsymbol{\alpha}_{k.}^T}{(\boldsymbol{\alpha}_{k.} \boldsymbol{\alpha}_{k.}^T - \gamma)^{-1}} \tag{2.24}$$

while the solution for Eq. 2.24 under the constraint $\boldsymbol{d}_k^T \boldsymbol{d}_k = 1$. is

$$\tilde{\boldsymbol{d}}_k = \frac{E_k \boldsymbol{\alpha}_{k.}^T}{\|E_k \boldsymbol{\alpha}_{k.}^T\|_2} \tag{2.25}$$

The above procedure is used to update each dictionary atom.

The individual learned dictionaries D_c are then put together to form a global dictionary as $D_G = [D_1, ..., D_C]$ and the classification of a test sample is done as described for SRC utilizing D_G.

Supervised dictionary learning

Mairal et al.[Mairal 08] proposed a supervised dictionary learning method where a new discriminative term added into the sparse representations formulation on the sparse coefficients. The idea is to encourage the class-specific dictionaries to be good at reconstructing samples of its own class and bad for the others. Consider $R^*(\boldsymbol{x}, D) = \|\boldsymbol{x} - D\boldsymbol{\alpha}\|_2^2$ as the representation error of $\boldsymbol{x}$ on D that should be minimized for $\boldsymbol{x}$ in the class c and maximized for other D's. For that, a *softmax* discriminative cost function is used as follow:

$$\zeta_c^{\lambda}(z_1, z_2, ..., z_C) = log\left(\sum_{m=1}^{C} e^{-\lambda(z_m - z_c)}\right) \quad \forall\, c = 1, ..., C \tag{2.26}$$

where the value of ζ_c^{λ} is close to zero when z_c is the smallest value among z_m's and provides a linear penalty cost $\lambda(z_c - min_m z_m)$ otherwise. $\lambda > 0$ controls the relative penalty cost. Using Eq. 2.26 the optimization problem to learn C discriminative sub-dictionaries is defined as:

$$\underset{\{D_m\}_{m=1}^{C}}{min} \sum_{\substack{c=1 \\ \boldsymbol{x}_l \in X_c}}^{C} \zeta_c^{\lambda}(\{R^*(x_l, D_m)\}_{m=1}^{C}) + \lambda\gamma R^*(x_l, D_c), \tag{2.27}$$

where X_c as before denotes the subset of the training samples from class c and the parameter $\gamma \geq 0$ control the trade-off between reconstruction and discrimination. With a high value for γ, the model is close to the classical reconstructive one. They proposed a kSVD like optimization procedure to solve Eq. 2.27. See [Mairal 08] for more details.

Once the dictionaries have been learned the class of the test sample $\boldsymbol{y}$ is found by solving:

$$c^* = \underset{(c)}{argmin}\ R^*(\boldsymbol{y}, D_c)$$

where $R^*(\boldsymbol{y}, D) = \|\boldsymbol{y} - D\boldsymbol{\alpha}\|_2^2$ represents the best representation error of $\boldsymbol{y}$ on D. This last step is similar to the SRC where, test samples are assigned to the class with the minimum reconstruction error.

Fisher Discrimination Dictionary Learning (FDDL)

Yang et al. [Yang 11] proposed a method called FDDL (Fisher Discrimination Dictionary Learning) that attempts to engage the Fisher discrimination criterion into the dictionary learning problem. The formalization results in learning discriminative class-specific sub-dictionaries as well as discriminative sparse coefficients similar to [Huang 06].

Denote by C the number of classes and $D = [D_1, ..., D_C]$ where D_c is the class-specific sub-dictionary associated with the class c, the FDDL model is defined as:

$$\arg\min_{D,A} \sum_{c=1}^{C} r(X_c, D, A_c) + \lambda_1 \|A\|_1 + \lambda_2 f(A) \tag{2.28}$$

where $A = [A_1, ...A_C]$, A_c is the sub-matrix containing the coding coefficients of X_c based on D, $\|A\|_1$ is the sparsity constraint and the functions r and f are defined as:

$$r(X_c, D, A_c) = \|X_c - DA_c\|_F^2 + \|X_c - D_c A_c^c\|_F^2 + \sum_{m \neq c}^{C} \|D_m A_c^m\|_F^2 \tag{2.29}$$

$$f(A) = tr(S_W(A) - S_B(A)) + \eta \|A\|_F^2 \tag{2.30}$$

where A_c^m is the sparse codes of X_c over the sub-dictionary D_m. $S_W(A)$ and $S_B(A)$ are the within-class and between-class scatter of A respectively.

FDDL model in Eq. 2.28 presents three terms, the sub-dictionary discriminative term (r), the sparsity term and the discriminative coefficient term (f). The function r ensures that the sub-dictionary corresponding to each class, well represents the samples from the same class, while having a poor representation power for the samples of the other classes. The function f, on the other hand, forces the coding coefficients of the different classes to be discriminative by using Fisher criterion, which minimizes the within-class scatter and maximizes the between-class scatter of the sparse coefficients.

The FDDL optimization problem in Eq. 2.28 can be solved literately by fixing D and updating A and updating D by fixing A. Similar to SRC, the classification of a test sample $\boldsymbol{y}$ can be done based on a minimum class-wise reconstruction error plus, the distance between its coefficient vector and the mean coefficient vector of a class.

Discriminative kSVD (D-kSVD) and LC-kSVD

Zhang and Li [Zhang 10] proposed a method called Discriminative kSVD (D-kSVD) to jointly learn a linear classifier on the sparse coefficients while looking for a desirable dictionary for reconstruction. To achieve this, a classifier term is added into the standard dictionary learning formalization and kSVD is applied to find the solution for all the parameters. The objective function of the D-kSVD is formulated as:

$$\begin{aligned} &\underset{A,D,W}{min} \|X - DA\|_F^2 + \gamma\|H - WA\|_F^2 + \beta\|W\|_F^2 \\ &s.t. \quad \forall i \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau. \end{aligned} \tag{2.31}$$

where H is the matrix composed of the label information of the training samples and W is the parameter of the classifier. γ and β are scalars controlling the relative contribution of the corresponding terms.

To employ the kSVD framework, the first two terms in Eq. 2.31 are combined and the problem is reformulated. For the classification phase, the sparse coefficient vector of a test sample is computed based on the learned dictionary D and the learned classifier W is used to predict its class. The main disadvantage of this method is utilizing a linear classifier which may leads to poor performance in difficult classification tasks.

Jiang et al. [Jiang 11] add a label consistent term to the D-kSVD method that forces the sparse vectors to be more discriminative. The Label Consistent kSVD (LC-kSVD) objective function is formulated as:

$$\begin{aligned} &\underset{A,D,W,T}{min} \|X - DA\|_F^2 + \gamma\|H - WA\|_F^2 + \beta\|Q - TA\|_F^2 \\ &s.t. \quad \forall i \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau. \end{aligned} \tag{2.32}$$

where the first two terms, the reconstruction error term and the classification error term, are as the same as D-kSVD. $Q = [\boldsymbol{q}_1, ..., \boldsymbol{q}_N] \in \mathbb{R}^{K \times N}$ is the optimal discriminative sparse coefficients matrix for X and D. Each column of Q ($\boldsymbol{q}_i = [0, ..., 1, 1, ..., 0]^T \in \mathbb{R}^K$) is the discriminative sparse code of the sample $\boldsymbol{x}_i$. For example, with $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4]$ and $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \boldsymbol{d}_4]$ where $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{d}_1, \boldsymbol{d}_2$ are from class one and $\boldsymbol{x}_3, \boldsymbol{x}_4$ and $\boldsymbol{d}_3, \boldsymbol{d}_4$ are from class two, the matrix Q is defined as:

$$Q = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Finally, T is a linear transformation matrix that transforms the original sparse codes to be most discriminative sparse coefficients (Q). Note that, the LC-kSVD sparse vectors are almost identical within a class.

Similar to D-kSVD, the three terms in Eq. 2.32 are fused together and the problem is reformulated to be solved using the conventional kSVD algorithm.

Dictionary Learning with Structured Incoherence (DLSI)

Ramirez et al. [Ramirez 10] proposed a structural incoherent dictionary learning method (DLSI). Followed by Meta-face learning approach, they proposed to add an incoherence term to the dictionary learning formalization to encourage the class-specific sub-dictionaries to be as independent as possible. The proposed formalization for data classification is:

$$\min_{\forall c\ (A_c,D_c)} \sum_c \left\{ \|X_c - D_c A_c\|_F^2 + \lambda \sum_{i=1}^{N} \|\boldsymbol{\alpha}_i^c\|_1 \right\} + \eta \sum_{n \neq m} \|D_n^T D_m\|_F^2 \tag{2.33}$$

where the last term is the dictionaries incoherence term and defined as the inner product between the two sub-dictionaries D_n and D_m. To minimize Eq. 2.33 an alternative minimization approach with two phases has been used, sparse coding and dictionary learning. The solution for learning the dictionary is a variation of the MOD method which is called MOCOD [Ramírez 09].

It is found empirically that even after learning dictionaries with incoherence term, sub-dictionaries shared some common atoms that would make the reconstruction error with different dictionaries similar. Thus, as an improvement, the common atoms with the inner product larger than a predefined threshold (0.95 in [Ramirez 10]) are also ignored.

Extension to clustering: Ramirez et al. [Ramirez 10] have also extended their method for unsupervised data clustering. Basically what they proposed is simultaneously learning a set of dictionaries that represent each cluster, parallel to the K-means type approaches that look for a set of centroids for the clusters.

Given a set of unlabelled samples $X = \{\boldsymbol{x}_i\}_{i=1}^N$ and the number of clusters K,the problem of finding a set of dictionaries $\{D_l\}_{l=1}^K$ and $C = \{C_l\}_{l=1}^K$ the partitioning of X into K clusters, is formulated as:

$$\min_{\forall l\ (C_l,D_l)} \sum_{l=1}^{K} \sum_{\boldsymbol{x}_i \in C_l} \min_{\boldsymbol{\alpha}_i^l} \|\boldsymbol{x}_i - D_l \boldsymbol{\alpha}_i^l\|_2^2 + \lambda \|\boldsymbol{\alpha}_i^l\|_1 + \eta \sum_{n \neq m} \|D_n^T D_m\|_F^2 \tag{2.34}$$

Similar to K-means, the solution for the optimization problem in Eq. 2.34 is an iterative refinement technique. In one iteration, each input is assigned to the cluster C_l for which the best representation (i.e., minimum reconstruction error) is obtained. Then the dictionaries D_l are updated based on the assignments found in the previous step. The dictionary update process is the same as for the classification. The refinement steps are repeated until the convergence (i.e., no changes in cluster assignments).

The authors proposed two initialization for the algorithm. One concerning the data (C_l) and one for sub-dictionaries (D_l). On both cases the main idea is to construct an affinity matrix for the spectral clustering algorithm. To do so, they first train an initial dictionary $D_0 = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{K0}]$ over the whole training set X and build two similarity matrices based on the corresponding sparse representations $A = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N]$; One measures the similarity of two inputs $S_1 = |A^T A| \in \mathbb{R}^{N \times N}$ and the other one stores the similarity of two atoms $S_2 = |AA^T| \in \mathbb{R}^{K0 \times K0}$. To obtain the initial partition of the inputs (C_l) or the sub-dictionaries (D_l), spectral clustering is applied on S_1 or S_2 respectively.

2.4 Temporal data and dictionary learning

There are numerous applications of supervised and unsupervised sparse coding and dictionary learning on static data such as images, however, small effort has been done for temporal data (e.g., time series, sequences, traces). The reason might be the special characteristics of these data that I mentioned in Chapter 1 such as internal dependencies or time delay. Time series data usually arise with varying delays where the standard Euclidean distance, which is commonly used in dictionary learning frameworks, can not be applied for them, due to the time warping.

In this section, we would like to address several methods that study the dictionary learning for temporal data. These approaches can be divided into two groups. The first group investigate the shift invariant sparse representations algorithms while the second groups benefit kernel methods to overcome the problem of time warp or varying length inputs.

2.4.1 Shift invariant sparse representation

The shift invariant sparse representation is originally proposed by Lewicki and Sejnowski [Lewicki 99] where an input sample is represented by a few set of kernel basis functions of small lengths that can be placed at any arbitrary position within the input. The model can be expressed in convolutional form [Smith 05] based on a circulant matrix to convolve the kernel functions at all the translation positions. The Gammatone functions are used as bases and the optimal coefficients found by maximizing a posterior distribution. The idea of convolutional sparse coding is further expanded by Grosse et al. [Grosse 07] to learn the sparse representation of a long audio signal. The proposed shift invariant sparse coding (SISC) algorithm is designed to learn both coefficients and the bases. Their algorithm reconstructs an input using a weighted combination of a few of the learned basis functions.

Denote by $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$, $\boldsymbol{x}_i \in \mathbb{R}^q$ a set of input samples (time series) and $\Psi = \{\boldsymbol{\psi}_m\}_{m=1}^{M}$, $\boldsymbol{\psi}_m \in \mathbb{R}^p$ a set of basis functions that are allowed to be of a lower dimension than the input ($p \leq q$), the SISC optimization problem is formulated as an extension to the sparse coding optimization problem as follows:

$$\begin{aligned} &\underset{(\Psi,S)}{min} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \sum_{m=1}^{M} \boldsymbol{\psi}_m * \boldsymbol{s}_{(i,m)}\|_2^2 + \beta \sum_{(i,m)} \|\boldsymbol{s}_{(i,m)}\|_1 \\ &s.t. \quad \forall m \ \|\boldsymbol{\psi}_m\|_2^2 \leq c. \end{aligned} \tag{2.35}$$

where $\boldsymbol{s}_{(i,m)} \in \mathbb{R}^{q-p+1}$ is the coefficient vector corresponding to the input $\boldsymbol{x}_i$ and basis $\boldsymbol{\psi}_m$, that is representing the magnitude for each possible temporal offset of the basis $\boldsymbol{\psi}_m$ within $\boldsymbol{x}_i$.

The above optimization problem are then solved by alternatively solving two large convex optimization problem: solving for the coefficients $\boldsymbol{s}_{(i,m)}$ given a fixed basis set as an l_1-regularized least squares problem and keeping the coefficients fixed, solving for bases as an l_2-constraint least squares problem over a vector of complex variables in the Fourier domain, knowing the fact that convolution in time domain is equivalent to the dot product in Fourier domain. A similar formalization and solutions are also proposed by [Bristow 13] where the inputs can be either $1D$ signals like audio or $2D$ signals like images and the dictionary also can be $1D$ vectors or $2D$ matrix accordingly. Grosse et al. [Grosse 07] proposed a computationally efficient implementations for both optimization steps. In the context of classification,

the learned coefficients on the training set are used as features to train an SVM classifier. The input feature vector is constructed as the average of the coefficient vectors over the entire input. Another solution for convolutional dictionary learning was proposed in [Kavukcuoglu 10] where a feed-forward, non-linear encoder is used to predict the sparse codes from the inputs.

The idea of shift invariant sparse coding is further studied in [Barthélemy 12] where the authors proposed a shift invariant sparse coding for multivariate time series. They proposed a multivariate version of the OMP followed by a multivariate dictionary learning algorithm to sparsely reconstruct the $2D$ handwritten temporal data. Mailhe et al. [Mailhé 08] proposed an extension of kSVD algorithm to learn a family of M shift invariant patterns form a long signal. The optimization problem uses a nested sum instead of convolution operator. A translation operator that allows to generate, for each pattern, all its translated copies is also proposed. As in the standard kSVD, the dictionary patterns are updated sequentially and the coefficients are updated accordingly before updating the next pattern. The proposed method has been tested to learn patterns on a long music track. Similar procedure is used in the algorithm proposed by Aharon and Elad [Aharon 08] to learn a dictionary to present varying size image patches. The method called image signature dictionary (ISD) where the dictionary is composed of varying size image patches extracted from varying locations of the images. A modified OMP and kSVD algorithms are used in learning the ISD. In the same spirit, [Jost 06] formalizes the translation invariant dictionary learning as a convex optimization problem to estimate atoms and their time-translation to maximize their correlation to the training data under uncorrelated atom constraint. Huang et al. [Huang 12] proposed a temporal pyramid pooling method to extract discriminative and shift invariant representation of audio signals. The coefficients are computed based on the model proposed by [Smith 05] and different pooling strategies are used to create the input feature vectors of an SVM classifier.

Shift invariant sparse coding models are mostly proposed for reconstruction and representation of a long signal where similar patterns may appear in different locations within it e.g., audio or music track. However, the generated sparse coefficients of such models are not discriminative nor suitable for a classification task. For example in [Grosse 07], the time series are partitioned into small segments of the same length, then a dictionary is learned to sparse represent these small segments.

Although the learned dictionary has been shown good ability to reconstruct the input time series, the classification accuracy based on the learned sparse codes is not satisfying. The main reason is probably due to the learned dictionary, that is composed of basis atoms that are not discriminative nor specific for a class of data and instead are commonly shared by the different classes.

In the next section, we introduce the basic idea of kernel sparse representation and dictionary learning. We then explain how these methods are used for classification of time series data.

2.4.2 Kernel sparse representation

To capture the non-linearity present in the data, kernel sparse representation method has been proposed [Gao 10] which presents a sparse coding schema in the feature space. Similarly, Nguyen et al. [Van Nguyen 12] present non-linear versions of the well-known dictionary learning approaches such as MOD and kSVD followed by a kernelized version of the orthogonal matching pursuit algorithm (KOMP) to sparse code inputs in the feature space. In the following we explain the formalization and the proposed solutions for the kernelized sparse representation problem.

Suppose there exists a non-linear mapping function $\phi : \mathbb{R}^q \rightarrow \mathcal{F}$ from $\mathbb{R}^q$ into a higher dimensional feature space $\mathcal{F}$. Denote by $\Phi(X) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ..., \phi(\boldsymbol{x}_N)]$ the matrix whose columns are obtained by embedding the inputs X using the mapping ϕ and $\Phi(D)$ the non-linear dictionary in the feature space $\mathcal{F}$, the kernel sparse representation and dictionary learning can be formulated as a kernelized form of the problem in Eq. 2.15:

$$\begin{aligned} &\min_{A,\Phi(D)} \sum_{i=1}^{N} \|\Phi(D)\boldsymbol{\alpha}_i - \phi(\boldsymbol{x}_i)\|_2^2 \\ &s.t. \quad \forall i \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau. \end{aligned} \tag{2.36}$$

where A as before is the matrix whose i^{th} column $\boldsymbol{\alpha}_i$ is the sparse vector of $\phi(\boldsymbol{x}_i)$ with maximum of τ non-zero entries. It has been shown in [Van Nguyen 13] that there exists an optimal solution $\Phi(D)^*$ with the form $\Phi(D)^* = \Phi(X)M$, for some $M \in \mathbb{R}^{N \times K}$. The assumption here is that the atoms are lying within the sub-space spanned by the input data. The problem in Eq.2.36 can be then re-written with the matrix notation as:

$$\min_{A,M} \|\Phi(X)MA - \Phi(X)\|_F^2 \quad s.t. \quad \forall i \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau. \tag{2.37}$$

Algorithm 4 Kernel Orthogonal Matching Pursuit (KOMP)

Input: $\boldsymbol{x}$, M, $\boldsymbol{\mathcal{K}}$, $\boldsymbol{k}_x$ τ.
Output: $\boldsymbol{\alpha}$ such that $\Phi(X)M\boldsymbol{\alpha}$ approximates $\phi(\boldsymbol{x})$.
1: $\Omega = \{\phi\}, s = 0, \boldsymbol{r} = \boldsymbol{0}$.
2: **while** $|\Omega| \leq \tau$ **do**
3: $\omega_i = (\boldsymbol{k}_x - \boldsymbol{r}^T\boldsymbol{\mathcal{K}})\boldsymbol{m}_i, \ \forall i \notin \Omega$.
4: Select $i_{max} = \arg\max_i |\omega_i|$ and update $\Omega = \Omega \cup \{i_{max}\}$.
5: Update the coefficients: $\boldsymbol{\alpha} = (M_\Omega^T \boldsymbol{\mathcal{K}} M_\Omega)^{-1}(\boldsymbol{k}_x M_\Omega)^T$.
6: Estimate the residual: $\boldsymbol{r} = M_\Omega \boldsymbol{\alpha}$.
7: **end while**

The above optimization can be carried out by iteratively solving two steps: 1) keep the dictionary $\Phi(X)M$ fixed and look for the sparse codes $\boldsymbol{\alpha}_i$'s in the feature space, 2) update the atom representation dictionary M while keeping the sparse representations A fixed.

For the first step, the matrix M is assumed to be fixed and the problem reduced to solve the N distinct sparse coding problem of the form:

$$\forall i, \quad \min_{\boldsymbol{\alpha}_i} \|\Phi(X)M\boldsymbol{\alpha}_i - \phi(\boldsymbol{x}_i)\|_2^2 \ \ s.t. \ \ \|\boldsymbol{\alpha}_i\|_0 \leq \tau. \tag{2.38}$$

where by introducing the kernel matrix $\boldsymbol{\mathcal{K}} = \Phi(X)^T\Phi(X)$ and kernel vector $\boldsymbol{k}_x = \Phi(X)^T\phi(\boldsymbol{x})$, the kernel orthogonal matching pursuit (KOMP) algorithm, as a generalization of the standard OMP, is used to find $\boldsymbol{\alpha}_i$s. The procedure follows the same steps of the OMP algorithm and the pseudo code is given in Algorithm 4.

In the second step and once the sparse codes of the training data are found, the dictionary representation matrix M is updated such that the following reconstruction error is minimized:

$$\|\Phi(X) - \Phi(X)MA\|_F^2 \tag{2.39}$$

Based on the idea of MOD method the following update rule can be applied to find M,

$$M = A^T(AA^T)^{-1}. \tag{2.40}$$

while in the kernel kSVD Eq. 2.39 is re-written as:

$$\begin{aligned} \|\Phi(X) - \Phi(X)MA\|_F^2 &= \|\Phi(X) - \Phi(X)\sum_{j=1}^{K} \boldsymbol{m}_j \, \boldsymbol{\alpha}_{j.}\|_F^2 \\ &= \|\Phi(X)(I - \sum_{j\neq k} \boldsymbol{m}_j\boldsymbol{\alpha}_{j.}) - \Phi(X)(\boldsymbol{m}_k\boldsymbol{\alpha}_{k.})\|_F^2 \end{aligned} \tag{2.41}$$

Algorithm 5 Kernel kSVD dictionary learning

Input: X, $\boldsymbol{\mathcal{K}}$, τ.
Output: The dictionary atom representation M and the sparse coefficients A.
Set τ random elements of each column in M to be 1.
repeat
{Sparse coding step:}
  for $i = 1, ..., N$ **do**
    $\boldsymbol{\alpha}_i$ =KOMP$(\boldsymbol{x}_i, M, \boldsymbol{\mathcal{K}}, \boldsymbol{k}_{x_i}, \tau)$.
  end for
{Dictionary update step:}
  for $k = 1, ..., K$ **do**
    Compute $E_k = (I - \sum_{j \neq k} \boldsymbol{m}_j \boldsymbol{\alpha}_{j.})$.
    Define $\Omega_k = \{i \,/\, \alpha_{ki} \neq 0, \;\; i = 1, ..., N\}$.
    Compute $E_k^{\Omega_k} = E_k \Omega_k$ as the restriction of E_k to Ω_k.
    Apply SVD decomposition: $(E_k^{\Omega_k})^T \boldsymbol{\mathcal{K}} (E_k^{\Omega_k}) = V \Sigma V^T$.
    Update $\boldsymbol{m}_k = \sigma_1^{-1} E_k^{\Omega_k} \boldsymbol{v}_1$ and $\boldsymbol{\alpha}_{k.}^{\Omega_k} = \sigma_1 \boldsymbol{v}_1^T$.
  end for
until Convergence (stopping rule)

A singular value decomposition (SVD) is then applied on the matrix $\Phi(X)(I - \sum_{j \neq k} \boldsymbol{m}_j \boldsymbol{\alpha}_{j.})$ to update $\boldsymbol{m}_k$ and $\boldsymbol{\alpha}_{k.}$. Note that since the row dimension of this matrix may be very large, the SVD decomposition is applied on its Gram matrix, which is independent of the row dimension. The entire kernel kSVD procedure is summarize in Algorithm 5.

To use the sparse representation coefficient for data classification in the high dimensional feature space, the kernel sparse representation based classification (kernel SRC) has been proposed [Yin 12, Zhang 12]. The classification strategy of kernel SRC is as we discussed as in the previous section for SRC, but this time is kernelized.

Since the mentioned non-linear dictionary learning algorithms (e.g., kernel kSVD) exploit the sparsity of the data in the high dimensional feature space, with an appropriate choice of the kernel, they circumvent the problem of delay and length for time series sparse representations. Recently, the authors of [Chen 15] introduced kernel sparse representation for time series classification. By using a Gaussian RBF kernel and substituting the Euclidean distance with the elastic distance measures such as DTW, time series were embedded into an implicit kernel space, which

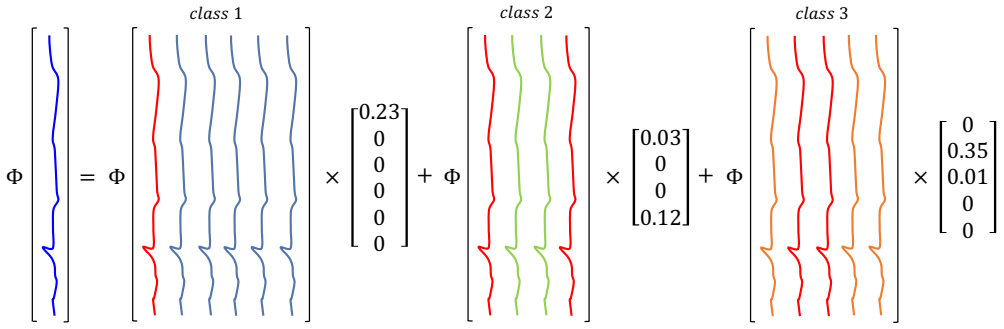

Figure 2.4: Kernel sparse representation of time series (Kernel SRC).

allowed the use of kernel SRC and kernel kSVD (Figure 2.4). Experimental results showed that the proposed method could be a competitive to the state of the art time series classification approaches.

Another effort to use kernels and a similar framework to SRC for time series classification is [Jeni 14]. In [Jeni 14] authors handle the temporal alignment problem by using General Alignment Kernel (KGA) [Cuturi 11] and proposed a structured sparse reconstruction to solve kernel sparse coding problem. The proposed method has been evaluated in the applications of gesture recognition and facial expression classification. Zhou et al [Zhou 13] have also proposed a kernel based sparse representation to generalize the capability of SRC to classify multivariate time series data. To design a dictionary from training data for SRC they proposed a feature extraction technique called Covariance Matrix Singular Value Decomposition for Kernelization (CovSVDK). They applied SVD on covariance matrix of each multivariate time series and used the related eigenvectors and eigenvalues as the features for sparse representation. This feature extraction strategy can overcome the problem of varying length inputs. A kernelized SRC is then applied in a human gesture classification application. Similarly, in a recent work, Poularakis and Katsavounidis [Poularakis 13] proposed a gesture recognition framework based on the SRC. Linear interpolation technique is used to represent gesture data and to create input features for the sparse representation framework.

2.5 Summary

The sparse representation framework consists of two main components. The sparse coding is defined as finding the coefficients for the representation of an input sample

by a linear combination of a few samples from a set of atoms called dictionary. Several approaches are proposed to solve the sparse approximation problem based on a l_0 or l_1 sparsity constraint. Dictionary learning, on the other hand, is a procedure of learning the set of samples or patterns to be used for the sparse representation framework. Among numerous dictionary learning approaches kSVD, MOD and the gradient descent approaches are the most popular. Although the standard dictionary learning problems learn the sparse codes and the dictionary in unsupervised manner, researches focus on designing methods for supervised dictionary learning either by manipulating its optimization problem or by producing discriminative sparse coefficient vectors.

Temporal dictionary learning has been also studied. The shift invariant sparse coding aims at learning a group of patterns to represent a long time series; usually the length of these pattern are much shorter than the original time series input. For that, convolutional sparse coding is formalized. Kernel sparse representation is also investigated for time series. Kernel OMP and kernel kSVD are proposed for time series reconstruction. SRC was also kernelized and used for temporal data classification where the Gaussian DTW kernel is used to bypass the delay problem.

We become what we think about.

— Earl Nightingale

3

Sparse Coding and Dictionary Learning under Time Warp

Contents

The existing challenges and issues for time series sparse representations and dictionary learning are discussed in Chapter 1. In standard dictionary learning methods the input samples and atoms are supposed of the same dimension. For temporal data, both input samples and atoms are time series that may involve

varying delays and be of different lengths, which renders the standard dictionary learning methods unusable. Several strategies have been proposed to address that problem, as discussed in the previous chapter. This chapter details the main contribution of the thesis which is a time warp invariant sparse representation and dictionary learning for temporal data. It starts with the problem statement, then details the proposed solutions.

3.1 Problem formalization

In this section, the proposed formalization of sparse coding and dictionary learning for time series under time warp is introduced. The problem is defined as a non convex optimization problem, which in general, can be resolved by using a block coordinate-descent method, that consists of updating just one or a few blocks of variables at a time while keeping the rest fixed. Efficient solutions for each phase of the optimization are detailed in the following sections. Note that, the coordinate descent technique is based on the idea that the minimization of a complex multivariate function can be achieved by minimizing it along one direction at a time. Such updating is computationally much cheaper than the batch update. On the other hand, convergence requires more tight conditions and typically takes more iterations.

Let $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$ be a set of N input time series $\boldsymbol{x}_i = (x_{i1}, ..., x_{iq_i})^T \in \mathbb{R}^{q_i}$ of length q_i and $D = \{\boldsymbol{d}_j\}_{j=1}^{K}$ the dictionary defined as a set of K time series atoms $\boldsymbol{d}_j \in \mathbb{R}^{p_j}$. Note that both inputs $\boldsymbol{x}_i$ and atoms $\boldsymbol{d_j}$ are time series of different lengths that may involve varying delays. The time warp invariant sparse coding and dictionary learning problem can be formalized as:

$$\begin{aligned} \min_{A,D,\boldsymbol{\Delta}} \mathcal{J}(A, D, \boldsymbol{\Delta}) \;\; &= \;\; \sum_{i=1}^{N} \|\boldsymbol{x}_i - \sum_{j=1}^{K} \Delta_{ij}\, \boldsymbol{d}_j\, \alpha_{ji}\|_2^2 \\ s.t. \qquad & \forall i \;\|\boldsymbol{\alpha}_i\|_0 \leq \tau, \quad \forall j \;\; \|\boldsymbol{d}_j\|_2 = 1 \\ & \Delta_{ij} \in \{0,1\}^{q_i \times p_j}, \;\; \Delta_{ij}\, \mathbf{1}_{p_j} = \mathbf{1}_{q_i} \end{aligned} \tag{3.1}$$

where $\boldsymbol{\Delta} = \{\Delta_{ij}\}$, $i = 1, ..., N; j = 1, ..., K$, with Δ_{ij} a binary matrix that encodes the alignment between $\boldsymbol{x}_i$ and $\boldsymbol{d}_j$, that we detail in the following. $A = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_N]$ is the sparse coefficients matrix with $\boldsymbol{\alpha}_i = (\alpha_{1i}, ..., \alpha_{Ki})^T$ and τ is the sparsity constraint factor. The last constraint is a row normalization of the estimated Δ_{ij} that ensures for $\boldsymbol{x}_i$ equally weighted time stamps.

In the same spirit as the standard dictionary learning approaches, the above optimization problem can be solved by iterating two steps. The first step is to fix dictionary D and estimate A and $\mathbf{\Delta}$ by decomposing the problem defined in Eq. 3.1 into N distinct time warp invariant sparse coding problems, that can be solved each by using TWI-OMP which is detailed in the following section. In the second step, based on the learned A and $\mathbf{\Delta}$, the objective is to update D to find a dictionary that leads to the lowest reconstruction error.

To solve the optimization problem in Eq. 3.1, as in conventional dictionary learning approaches, we divide the problem into two distinct problems. First, for the sparse coding step, a time warp invariant orthogonal matching pursuit (TWI-OMP), based on a new cosine maximization time warp operator (COSTW) and the induced sibling atoms, is proposed. In the second step, we address the dictionary update problem by developing two solutions: A non linear time warp invariant kSVD (TWI-kSVD) and a gradient descent based solution (TWI-GDDL).

3.2 Time warp invariant sparse coding

We start with the formalization of the time series sparse coding under time warp invariances. Let us note that the standard approaches such as OMP is not applicable on the time series inputs since the dictionary can not be structured as a matrix and the cosine similarity can not be computed due the varying length of the inputs. A naive solution would be the zero padding of the data which leads to a weak result and also simply ignore the potentially existing delay. To resolve this problem, we present a new operator COSTW that ensures cosine maximization between time series under time warp and give the recurrence relation that ensures its computation in quadratic complexity. Finally, thanks to COSTW and to the induced sibling atoms, we present a time warp invariant OMP (TWI-OMP), as a solution for the time series sparse coding problem under time warp.

Thus, for one input time series sample $\boldsymbol{x} = (x_1, ..., x_q)^T$ and the dictionary $D = \{\boldsymbol{d}_j\}_{j=1}^K$ of K atoms $\boldsymbol{d}_j \in \mathbb{R}^{p_j}$. The sparse coding problem under time warp

invariances can be formalized as:

$$\begin{aligned} & \min_{\boldsymbol{\alpha},\,\Delta} \|\boldsymbol{x} - \sum_{j=1}^{K} \Delta_j\, \boldsymbol{d}_j\, \alpha_j\|_2^2 \\ s.t. \quad & \|\boldsymbol{\alpha}\|_0 \leq \tau, \\ & \Delta_j \in \{0,1\}^{q \times p_j},\ \Delta_j\, \mathbf{1}_{p_j} = \mathbf{1}_q \end{aligned} \tag{3.2}$$

where $\Delta = \{\Delta_j\}_{j=1}^{K}$. The binary matrix $\Delta_j \in \{0,1\}^{q \times p_j}$ encodes the alignment between $\boldsymbol{x}$ and $\boldsymbol{d}_j$ obtained by dynamic programming as detailed after. The problem defined in Eq. (3.2) remains to estimate the coefficients $\boldsymbol{\alpha}$ to sparse code $\boldsymbol{x}$ as a linear combination of the warped atoms $\Delta_j\, \boldsymbol{d}_j$. To resolve this problem, we propose an extended variant of OMP that can be mainly summarized in the following steps:

1. For each $\boldsymbol{d}_j$, estimate Δ_j by dynamic programming to maximize the cosine between $\boldsymbol{x}$ and $\boldsymbol{d}_j$.
2. Use the projector Δ_j to align $\boldsymbol{d}_j$ to $\boldsymbol{x}$. Let $\boldsymbol{d}_j^s = \Delta_j\, \boldsymbol{d}_j \in \mathbb{R}^q$ be the obtained aligned atom of the same length as $\boldsymbol{x}$, denoted in the following as $\boldsymbol{d}_j$'s *sibling* atom.
3. Estimate the sparse code $\boldsymbol{\alpha}$ based on the sibling atoms.

For that and to estimate the projectors $\Delta_j, j = 1, ..., K$, first we propose a new operator COSTW to estimate the cosine between two time series under time warp. To the best of our knowledge, this is the first time that the cosine operator is generalized to time series under time warp. Then, we present a time warp invariant OMP (TWI-OMP), that extends the standard OMP approach, to sparse code time series under non linear time warping transformations.

The problem of estimating the cosine between two time series that involve varying delays amounts to learn a time series alignment that maximizes their cosine. In the following, we recall the standard definition of a valid alignment between time series, give some standard applications of the temporal alignment, then formalize the cosine maximization under time warp problem. Finally, we propose a recurrence relation that allows to perform the computation of the alignment with quadratic complexity.

3.2.1 Time series alignment

Let $\boldsymbol{x} = (x_1, ..., x_{q_x})$, $\boldsymbol{y} = (y_1, ..., y_{q_y})$ be two time series of length q_x and q_y. An alignment $\boldsymbol{\pi}$ of length $|\boldsymbol{\pi}| = m$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as the set of m increasing couples of aligned elements of $\boldsymbol{x}$ to elements of $\boldsymbol{y}$:

$$\boldsymbol{\pi} = ((\pi_1(1), \pi_2(1)), (\pi_1(2), \pi_2(2)), ..., (\pi_1(m), \pi_2(m)))$$

where the applications π_1 and π_2 defined from $\{1, ..., m\}$ to, respectively, $\{1, .., q_x\}$ and $\{1, .., q_y\}$ and that obey to the following boundary and monotonicity conditions:

$$1 = \pi_1(1) \leq \pi_1(2) \leq ... \leq \pi_1(m) = q_x$$
$$1 = \pi_2(1) \leq \pi_2(2) \leq ... \leq \pi_2(m) = q_y$$

and $\forall l \in \{1, ..., m\}$,

$$\pi_1(l+1) \leq \pi_1(l) + 1 \text{ and } \pi_2(l+1) \leq \pi_2(l) + 1,$$
$$(\pi_1(l+1) - \pi_1(l)) + (\pi_2(l+1) - \pi_2(l)) \geq 1$$

Intuitively, an alignment $\boldsymbol{\pi}$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ describes a way to associate each element of $\boldsymbol{x}$ to one or more elements of $\boldsymbol{y}$ and vice versa. Such an alignment can be conveniently represented by a path in the $q_x \times q_y$ grid (Figure 3.1 left), where the above monotonicity conditions ensure that the path is neither going back nor jumping. In the following, we denote $\mathcal{A}$ the set of all valid alignments between two time series.

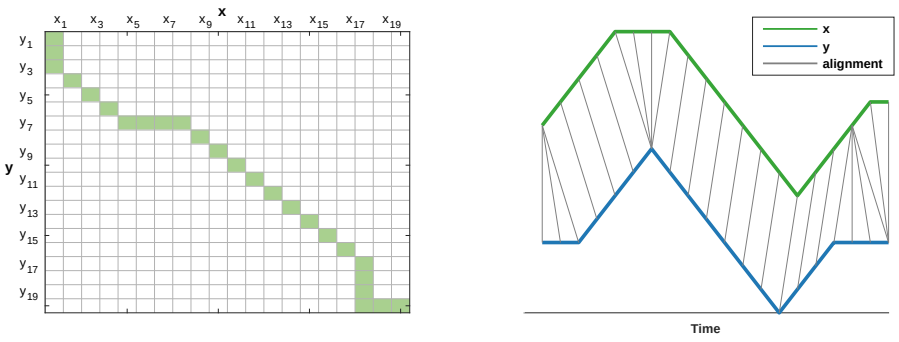

Figure 3.1: A possible alignment (path) between time series $\boldsymbol{x}$ and $\boldsymbol{y}$.

3.2.2 Standard use of time series alignments

Temporal alignment is generally required to estimate the proximity between time series. For instance, Dynamic Time Warping (DTW) is undeniably the most frequently used measure for time series comparison under time warp. Its standard form is defined as:

$$\begin{aligned} \text{DTW}(\boldsymbol{x}, \mathbf{y}) &= s(\boldsymbol{\pi}^*) \\ \boldsymbol{\pi}^* &= \arg\min_{\boldsymbol{\pi}\in\mathcal{A}} s(\boldsymbol{\pi}) \\ s(\boldsymbol{\pi}) &= \frac{1}{|\boldsymbol{\pi}|}\sum_{(t,t')\in\boldsymbol{\pi}} (x_t - y_{t'})^2 = \frac{1}{|\boldsymbol{\pi}|}\sum_{i=1}^{|\boldsymbol{\pi}|}(x_{\pi_1(i)} - y_{\pi_2(i)})^2 \end{aligned} \tag{3.3}$$

3.2.3 The cosine estimation between time series (COSTW)

To estimate the cosine between time series under time warp, the problem remains to seek for the alignment that maximize their cosine; that can be then formalized as:

$$\begin{aligned} \text{COSTW}(\boldsymbol{x}, \boldsymbol{y}) &= s(\boldsymbol{\pi}^*) \\ \boldsymbol{\pi}^* &= \arg\max_{\boldsymbol{\pi}\in\mathcal{A}} s(\boldsymbol{\pi}) \\ s(\boldsymbol{\pi}) &= cos(\boldsymbol{x}_{\pi_1}, \boldsymbol{y}_{\pi_2}) = \frac{< \boldsymbol{x}_{\pi_1}, \boldsymbol{y}_{\pi_2} >}{\|\boldsymbol{x}_{\pi_1}\|_2 \, \|\boldsymbol{y}_{\pi_2}\|_2} \\ &= \frac{\sum_{i=1}^{|\boldsymbol{\pi}|} x_{\pi_1(i)}\, y_{\pi_2(i)}}{\sqrt{\sum_{i=1}^{|\boldsymbol{\pi}|} x_{\pi_1(i)}^2}\,\sqrt{\sum_{i=1}^{|\boldsymbol{\pi}|} y_{\pi_2(i)}^2}} \end{aligned} \tag{3.4}$$

where s is the cost function of an alignment $\boldsymbol{\pi}$ and $cos(\boldsymbol{x}_{\pi_1}, \boldsymbol{y}_{\pi_2})$ is the standard cosine between the aligned time series $\boldsymbol{x}_{\pi_1} = (x_{\pi_1(1)}, ..., x_{\pi_1(m)})$ and $\boldsymbol{y}_{\pi_2} = (y_{\pi_2(1)}, ..., y_{\pi_2(m)})$. The solution of the Eq. 3.4 is obtained by dynamic programming, where the main trick is to define a useful recurrence relation for the cosine estimation, as detailed hereafter.

Let $\boldsymbol{x}_{q+1} = (x_1, ..., x_{q+1})$, $\boldsymbol{y}_{q+1} = (y_1, ..., y_{q+1})$ be two time series of length $q+1$, assumed without delays for the sake of clarity. Let $\boldsymbol{x}_q$, $\boldsymbol{y}_q$ be the sub-time series composed of the q first elements of $\boldsymbol{x}_{q+1}$, $\boldsymbol{y}_{q+1}$, respectively. The following incremental relation can be established between $cos(\boldsymbol{x}_{q+1}, \boldsymbol{y}_{q+1})$ and $cos(\boldsymbol{x}_q, \boldsymbol{y}_q)$:

$$\begin{aligned} cos(\boldsymbol{x}_q, \boldsymbol{y}_q) &= \frac{< \boldsymbol{x}_q, \boldsymbol{y}_q >}{\sqrt{\|\boldsymbol{x}_q\|_2^2}\sqrt{\|\boldsymbol{y}_q\|_2^2}} \\ cos(\boldsymbol{x}_{q+1}, \boldsymbol{y}_{q+1}) &= \frac{< \boldsymbol{x}_q, \boldsymbol{y}_q > + x_{q+1}y_{q+1}}{\sqrt{\|\boldsymbol{x}_q\|_2^2 + x_{q+1}^2}\sqrt{\|\boldsymbol{y}_q\|_2^2 + y_{q+1}^2}} \end{aligned} \tag{3.5}$$

For time series including delays and based on the incremental property given in Eq. 3.5, we introduce the computation and recurrence relation that allows to estimate the alignment $\boldsymbol{\pi}^*$ that maximize $\text{COSTW}(\boldsymbol{x}, \boldsymbol{y})$ in Eq. 3.4.

Computation and recurrence relation: Let $C \in \mathbb{R}^{q_x \times q_y \times 3}$ be the cost matrix with general term $C_{i,j} = (x_i y_j, x_i^2, y_j^2)$. Let's denote by $\boldsymbol{x}_{ij}$ and $\boldsymbol{y}_{ij}$ the aligned time series that maximize the cosine between the sub-time series $\boldsymbol{x}_i = (x_1, ..., x_i)$ and $\boldsymbol{y}_j = (y_1, ..., y_j)$. Let $M \in \mathbb{R}^{q_x \times q_y \times 3}$ be the alignment matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$ with $M_{i,j} = (< \boldsymbol{x}_{ij}, \boldsymbol{y}_{ij} >, \|\boldsymbol{x}_{ij}\|_2^2, \|\boldsymbol{y}_{ij}\|_2^2)$.

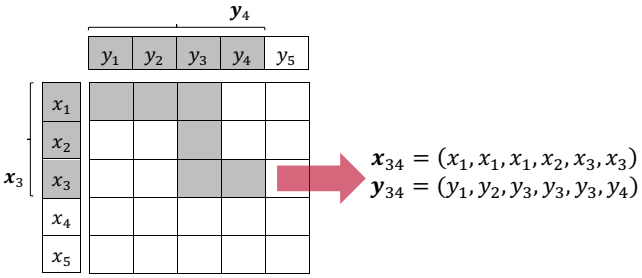

Figure 3.2: An example of the intermediate aligned time series: $\boldsymbol{x}_{34}$ and $\boldsymbol{y}_{34}$ are two aligned time series that maximize the cosine between the sub-time series $\boldsymbol{x}_3 = (x_1, x_2, x_3)$ and $\boldsymbol{y}_4 = (y_1, y_2, y_3, y_4)$.

Based on the incremental property established in Eq. 3.5, computing recursively for $(i, j) \in \{1, ..., q_x\} \times \{1, ..., q_y\}$ the terms $M_{i,j}$ as:

$$
\begin{aligned}
M_{1,1} &= C_{1,1} \\
\forall i \geq 2, j = 1, M_{i,1} &= M_{i-1,1} \oplus C_{i,1} \\
\forall j \geq 2, i = 1, M_{1,j} &= M_{1,j-1} \oplus C_{1,j} \\
\forall i \geq 2, j \geq 2, M_{i,j} &= \arg\max(f(M_{i-1,j} \oplus C_{i,j}), f(M_{i,j-1} \oplus C_{i,j}), f(M_{i-1,j-1} \oplus C_{i,j}))
\end{aligned}
$$

with $\oplus$ the sum operator for triplets [1] and $f(M_{i,j})$ the cosine reached at $M_{i,j}$ defined as:

$$f(M_{i,j}) = \frac{M_{i,j}[1]}{\sqrt{M_{i,j}[2]}\sqrt{M_{i,j}[3]}}$$

[1] $(a, b, c) \oplus (e, f, g) = (a + e, b + f, c + g)$

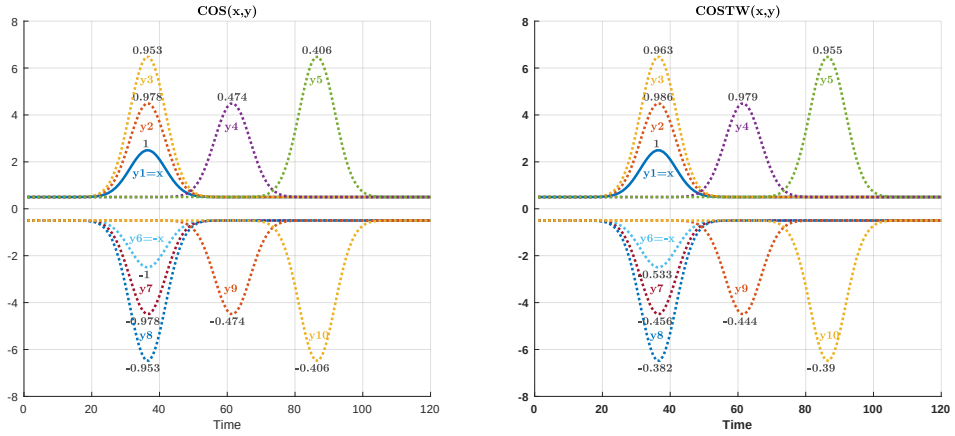

Figure 3.3: The progression of the standard cosine and COSTW between $\boldsymbol{x}$ (blue curve) and the time series $\boldsymbol{y}_1$ to $\boldsymbol{y}_{10}$ that involve different delays and amplitude variations, with in particular $\boldsymbol{y}_1 = \boldsymbol{x}$ and $\boldsymbol{y}_6 = -\boldsymbol{x}$.

we obtain an estimation of $\text{COSTW}(\boldsymbol{x}, \boldsymbol{y}) \approx f(M_{q_x,q_y})$. Note that due to the non-monotonicity of the cost function, the Bellman's principle of optimality is not satisfied. Even the solution is sub-optimal, it leads to a good approximation for cosine maximization for time series with a quadratic complexity of $\mathcal{O}(q_x q_y)$. The three first equations give the first row and column update rules, the fourth equation gives the recurrence formula that retains among the triplets $M_{i-1,j}, M_{i,j-1}$ and $M_{i-1,j-1}$ the one that maximizes the cosine at $M_{i,j}$. The optimal path $\boldsymbol{\pi}^*$ can be found by backtracking the optimal direction at each step, starting from M_{q_x,q_y} to $M_{1,1}$. The binary alignment matrices $\Delta_{xy} \in \{0,1\}^{q_x \times q_y}$ defined in Eq. 3.1 and 3.2 are then obtained from $\boldsymbol{\pi}^*$ as $\Delta_{xy}(i,j) = 1$ if $(i,j) \in \boldsymbol{\pi}^*$ and 0 otherwise. It is worth to say that by restraining the search space using constraint techniques (e.g Sakoe-Chiba band [Sakoe 78]) we can reduce the complexity of COSTW and speed up the alignment finding process.

Figure 3.3 shows the progression of the standard cosine (left) and COSTW (right) between the time series $\boldsymbol{x}$ (in blue) and 10 time series $\boldsymbol{y}_j$, $j = \{1, ..., 10\}$ that involve different delays and amplitude variations. We can note that the standard cosine $\text{COS}(\boldsymbol{x}, \boldsymbol{y}_j)$, that ignores the temporal dependency, reaches 1 or almost 1 for $\boldsymbol{y}_1$ to $\boldsymbol{y}_3$, that have no delays but amplitude variations, while it decreases for $\boldsymbol{y}_4$ and $\boldsymbol{y}_6$ that involve progressive delays. On the other hand, $\text{COS}(\boldsymbol{x}, \boldsymbol{y}_j)$ reaches -1 or almost -1 for $\boldsymbol{y}_6$ to $\boldsymbol{y}_8$, that have no delays with amplitude variations, while it increases

for $\boldsymbol{y}_9$ and $\boldsymbol{y}_{10}$ that involve increasing delays. For $\textsc{costw}(\boldsymbol{x}, \boldsymbol{y}_j)$, that seeks for an alignment that maximizes the cosine, the value is 1 or almost -1 for $\boldsymbol{y}_1$ to $\boldsymbol{y}_5$ regardless of the varying delays, and leads to higher values than the standard cosine for $\boldsymbol{y}_6$ to $\boldsymbol{y}_{10}$, showing its invariance in the face of delays.

3.2.4 The dot product estimation between time series (DPTW)

As noted in Section 3.2.3, the previous estimation of COSTW is sub-optimal due to the non-monotonicity of the cost function. In this section, we propose a COSTW approximation based on an optimal solution of the dot product under time warp by dynamic programming that satisfies the Bellman's principle of optimality. The dot product maximization time warp (DPTW) is formalized as:

$$\begin{aligned} \textsc{dptw}(\boldsymbol{x}, \boldsymbol{y}) &= s(\boldsymbol{\pi}^*) \\ \boldsymbol{\pi}^* &= \arg\max_{\boldsymbol{\pi} \in \mathcal{A}} s(\boldsymbol{\pi}) \\ s(\boldsymbol{\pi}) &= < \boldsymbol{x}_{\pi_1}, \boldsymbol{y}_{\pi_2} > = \sum_{i=1}^{|\boldsymbol{\pi}|} x_{\pi_1(i)}\, y_{\pi_2(i)} \end{aligned} \tag{3.6}$$

where this time the cost function s of the alignment $\boldsymbol{\pi}$ is defined by the standard dot product between the aligned time series $\boldsymbol{x}_{\pi_1}$ and $\boldsymbol{y}_{\pi_2}$. Let $\boldsymbol{x}_{q+1} = (x_1, ..., x_{q+1})$, $\boldsymbol{y}_{q+1} = (y_1, ..., y_{q+1})$ be two time series of length $q+1$, assumed without delays for the sake of clarity. Let $\boldsymbol{x}_q$, $\boldsymbol{y}_q$ be the sub-time series composed of the q first elements of $\boldsymbol{x}_{q+1}$, $\boldsymbol{y}_{q+1}$, respectively. Taking the following recurrence formulas of the dot product into account,

$$< \boldsymbol{x}_{q+1}, \boldsymbol{y}_{q+1} > \;=\; < \boldsymbol{x}_q, \boldsymbol{y}_q > + x_{q+1} y_{q+1}, \tag{3.7}$$

the solution of the Eq. 3.6 to find $\textsc{dptw}(\boldsymbol{x}, \boldsymbol{y})$ can be obtained by dynamic programming. The detailed computations are given hereafter.

Computation and recurrence relation: Let $C \in \mathbb{R}^{q_x \times q_y}$ be the cost matrix with general term $C_{i,j} = x_i y_j$. Let's denote by $\boldsymbol{x}_{ij}$ and $\boldsymbol{y}_{ij}$ the aligned time series that maximize the inner product between $\boldsymbol{x}_i = (x_1, ..., x_i)$ and $\boldsymbol{y}_j = (y_1, ..., y_j)$. Let $M \in \mathbb{R}^{q_x \times q_y}$ be the alignment matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$ with $M_{i,j} =< \boldsymbol{x}_{ij}, \boldsymbol{y}_{ij} >$.

Based on the incremental property established in Eq. 3.7, computing recursively for $(i,j) \in \{1,...,q_x\} \times \{1,...,q_y\}$ the terms $M_{i,j}$ as:

$$\begin{aligned} & M_{1,1} = C_{1,1} \\ \forall i \geq 2, j = 1, \quad & M_{i,1} = M_{i-1,1} + C_{i,1} \\ \forall j \geq 2, i = 1, \quad & M_{1,j} = M_{1,j-1} + C_{1,j} \\ \forall i \geq 2, j \geq 2, \quad & M_{i,j} = max(M_{i,j-1}, M_{i-1,j}, M_{i-1,j-1}) + C_{i,j} \end{aligned}$$

we obtain $\text{DPTW}(\boldsymbol{x},\boldsymbol{y}) = M_{q_x,q_y}$ with a quadratic complexity of $\mathcal{O}(q_x q_y)$ and $\text{COSTW}(\boldsymbol{x},\boldsymbol{y}) \approx \frac{DPTW(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}_{\pi_1^*}\|\|\boldsymbol{y}_{\pi_2^*}\|}$, where $\boldsymbol{x}_{\pi_1^*}$ and $\boldsymbol{y}_{\pi_2^*}$ are the time series $\boldsymbol{x}$ and $\boldsymbol{y}$ once aligned through the optimal path $\boldsymbol{\pi}^*$ reached by the DPTW.

3.2.5 Time warp invariant OMP (TWI-OMP)

Based on the estimated COSTW given either in Section 3.2.3 or Section 3.2.4, let us present the time warp invariant OMP (TWI-OMP) to solve the problem in Eq. 3.2 that extends the standard OMP algorithm to deal with time series input samples and atoms under varying delays.

The proposed TWI-OMP follows the three steps given in Section 3.2. First, perform a COSTW between $\boldsymbol{x}$ and each $\boldsymbol{d}_j$. Let $\{\Delta_j\}_{j=1}^K$ be the induced alignment matrices. Then, select the atom $\boldsymbol{d}_j$ that maximizes $\text{COSTW}(\boldsymbol{x},\boldsymbol{d}_j)$ (line 3-4 in Algorithm 6). Let $\boldsymbol{d}_j^s = \Delta_j \boldsymbol{d}_j$ be the $\boldsymbol{d}_j$'s sibling atom, update the dictionary $S_\Omega = [\boldsymbol{d}_j^s]_{j\in\Omega}$ of the yet selected atoms $\boldsymbol{d}_j$ (line 5). The updated S_Ω is then used to estimate the coefficients as in the standard OMP (line 6-7). The process is reiterated on the residuals of $\boldsymbol{x}$ until the sparsity factor τ is reached.

3.3 Time warp invariant dictionary learning

For the dictionary learning step, the problem in Eq. 3.1 becomes to learn the dictionary D under time warp where, the sparse coefficients $A = [\boldsymbol{\alpha}_1,...,\boldsymbol{\alpha}_N]$ and alignment matrices $\boldsymbol{\Delta} = \{\Delta_{ij}\}$, $i = 1,...,N; j = 1,...,K$ are assumed to be learned by TWI-OMP described in the previous section. Having A and $\boldsymbol{\Delta}$, we can formalize dictionary learning problem under time warp as:

$$\begin{aligned} & \min_D \sum_{i=1}^N \|\boldsymbol{x}_i - \sum_{j=1}^K \Delta_{ij}\, \boldsymbol{d}_j\, \alpha_{ji}\|_2^2 \\ s.t. \quad & \forall j \;\; \|\boldsymbol{d}_j\|_2 = 1 \end{aligned} \tag{3.8}$$

Algorithm 6 Time Warp Invariant Orthogonal Matching Pursuit (TWI-OMP)

Input: $\boldsymbol{x}$, D, τ.
Output: $\boldsymbol{\alpha}$, Δ.
$\boldsymbol{r} = \boldsymbol{x}, \Omega = \{\emptyset\}$.
while $|\Omega| \leq \tau$ **do**
  For all $j \notin \Omega$, perform COSTW$(\boldsymbol{r}, \boldsymbol{d}_j)$ and set Δ_j.
  Select the atom $\boldsymbol{d}_j$ $(j \notin \Omega)$ that maximizes $|\text{COSTW}(\boldsymbol{r}, \boldsymbol{d}_j)|$
  Update the set of selected atoms $\Omega = \Omega \cup \{j\}$ and $S_\Omega = [\boldsymbol{d}_j^s]_{j \in \Omega}$
  Update the coefficients: $\boldsymbol{\alpha}_{\boldsymbol{\Omega}} = (S_\Omega^T S_\Omega)^{-1}(S_\Omega^T \boldsymbol{x})$
  Estimate the residual: $\boldsymbol{r} = \boldsymbol{x} - S_\Omega \boldsymbol{\alpha}_{\boldsymbol{\Omega}}$
end while

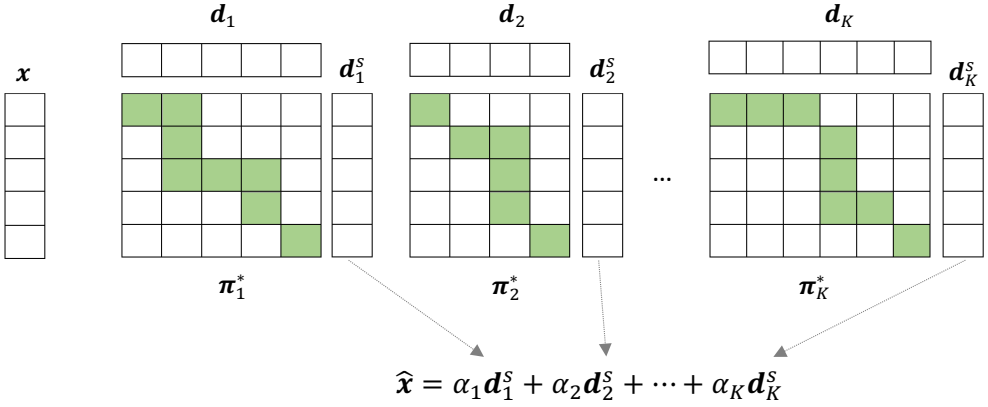

Figure 3.4: Reconstruction of the input $\boldsymbol{x}$ based on sibling dictionary atoms $\boldsymbol{d}_j^s$.

3.3.1 Time warp invariant kSVD (TWI-kSVD)

In the same spirit as in a standard kSVD, our objective is to update one atom and its related coefficients at a time. Recall from the kSVD, at each step, we remove one atom from the dictionary and replace it with the the major direction of the residuals. Similarly, once we remove atom $\boldsymbol{d}_k$, it induces the removal of all its sibling atoms $\boldsymbol{d}_k^{s_i}$. Thus, the residuals $\boldsymbol{e}_i \in \mathbb{R}^{q_i}$ of $\boldsymbol{x}_i$ are this time estimated w.r.t the sibling atoms $\boldsymbol{d}_k^{s_i}$ whereas in standard kSVD all the residuals are determined based on the same removed atom $\boldsymbol{d}_k$. As the residuals are not associated to a single referential vector, building the residual matrix E_k (Eq. 2.20) is unfeasible and SVD inapplicable.

To address this issue, we propose a solution that consists of two main steps:

1. use a projection and rotation transformation $\varphi(\boldsymbol{e_i})$ to represent all the residuals w.r.t a common referential vector, the residual matrix E_k can then be estimated

Algorithm 7 Rotation

Input: $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$.
Output: $\boldsymbol{a}_r$.

1: Set $\theta = \theta_{b,c}$, $\boldsymbol{u} = \dfrac{\boldsymbol{b}}{\|\boldsymbol{b}\|}$, $\boldsymbol{v} = \dfrac{\boldsymbol{c} - (\boldsymbol{u}^t\boldsymbol{c})\boldsymbol{u}}{\|\boldsymbol{c} - (\boldsymbol{u}^t\boldsymbol{c})\boldsymbol{u}\|}$
2: Compute the rotation matrix R as follows:

$$R = I - \boldsymbol{u}\boldsymbol{u}^t - \boldsymbol{v}\boldsymbol{v}^t + [\boldsymbol{u}; \boldsymbol{v}] R_\theta [\boldsymbol{u}; \boldsymbol{v}]^t$$

where $R_\theta \in \mathbb{R}^{2\times 2} = [(cos(\theta), sin(\theta))^t; (-sin(\theta), cos(\theta))^t]$
3: Compute $\boldsymbol{a}_r = R\,\boldsymbol{a}$.

and SVD applied to estimate the first left-singular vector $\boldsymbol{u}_1$ and update $\boldsymbol{d}_k$ accordingly,

2. use back transformations $\gamma_i(\boldsymbol{u}_1)$ of $\boldsymbol{u}_1$ to update each sibling atom $\boldsymbol{d}_k^{s_i}$ as well as its related coefficients.

In the following we detail these steps and define the related projection and rotation transformations.

Let $\boldsymbol{d}_k$ and $\{\boldsymbol{d}_k^{s_i}\}_{i=1}^N$ be the k^{th} atom and its sibling atoms to be removed. The residual $\boldsymbol{e}_i$ (i.e., the reconstruction error) of $\boldsymbol{x}_i$ once $\boldsymbol{d}_k^{s_i}$ removed is computed as:

$$\boldsymbol{e}_i = \boldsymbol{x}_i - \sum_{j\neq k} \boldsymbol{d}_j^{s_i} \alpha_{ji} \tag{3.9}$$

Let us define $\varphi(\boldsymbol{e}_i) : \mathbb{R}^{q_i} \to \mathbb{R}^{p_k}$ as the transformation that represents the residuals $\boldsymbol{e}_i$ w.r.t. the common referential vector $\boldsymbol{d}_k$. First, it consists to align $\boldsymbol{e}_i$ and $\boldsymbol{d}_k^{s_i}$ to $\boldsymbol{d}_k$ thanks to the projector Δ_{ik}. As the alignment matrix Δ_{ik} is used to project $\boldsymbol{d}_k$ to $\boldsymbol{x}_i$, similarly Δ_{ik}^T can do the opposite, i.e., align the residual to the atom $\boldsymbol{d}_k$. Let $\Delta_{ik}^T\,\boldsymbol{e}_i$ and $\Delta_{ik}^T\,\boldsymbol{d}_k^{s_i}$ be the obtained aligned vectors of the same lengths as $\boldsymbol{d}_k$. Subsequently, $\Delta_{ik}^T\,\boldsymbol{e}_i$ is rotated such that the angle between $\varphi(\boldsymbol{e}_i)$ and $\boldsymbol{d}_k$ is the same as the one between $\boldsymbol{e}_i$ and $\boldsymbol{d}_k^{s_i}$. The effect of the transformation φ is described in Figure 3.5 and formalized as:

$$\varphi(\boldsymbol{e}_i) \quad = Rotation(\Delta_{ik}^T\,\boldsymbol{e}_i, \Delta_{ik}^T\,\boldsymbol{d}_k^{s_i}, \boldsymbol{d}_k) \tag{3.10}$$

where $\boldsymbol{a}_r = Rotation(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ operates a rotation of the vector $\boldsymbol{a}$ to $\boldsymbol{a}_r$ such that $\theta_{\boldsymbol{a},\boldsymbol{b}} = \theta_{\boldsymbol{a}_r,\boldsymbol{c}}$ [2] [Arfken 99], the procedure is given in Algorithm 7 (More details are given in Appendix A).

[2] $\theta_{\boldsymbol{a},\boldsymbol{b}}$ denotes the angle between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$

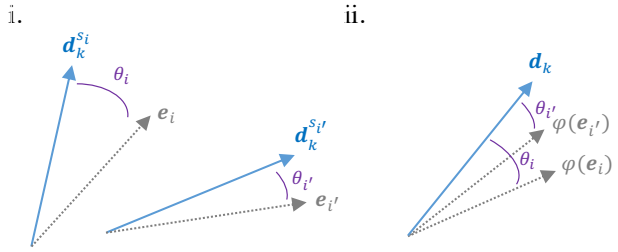

Figure 3.5: φ: representation of the residuals $\boldsymbol{e}_i$, $\boldsymbol{e}_{i'}$ w.r.t the common referential $\boldsymbol{d}_k$.

Let us denote $E_k[\varphi(\boldsymbol{e}_i)]_{i\in\boldsymbol{\omega}_k} \in \mathbb{R}^{p_k\times|\boldsymbol{\omega}_k|}$ the matrix of the residuals $\varphi(\boldsymbol{e}_i)$ with $\boldsymbol{\omega}_k = \{i\,|\alpha_{ki} \neq 0, i = 1, ...N\}$ is the set of $\boldsymbol{x}_i$ indices that involve $\boldsymbol{d}_k^{s_i}$. An SVD [3] is then applied on $E_k = U\Sigma V^T$ to determine the first left-singular vector $\boldsymbol{u}_1$. The k^{th} atom is updated as $\boldsymbol{d}_k = \boldsymbol{u}_1$. The sibling $\boldsymbol{d}_k^{s_i}$ for $i \in \boldsymbol{\omega}_k$ as well as their coefficients are obtained based on the back transformation γ_i of $\boldsymbol{u}_1$ as:

$$\gamma_i(\boldsymbol{u}_1) = \Delta_{ik}\, Rotation(\boldsymbol{u}_1, \boldsymbol{d}_k, \Delta_{ik}^T \boldsymbol{d}_k^{s_i}) \tag{3.11}$$

$$\boldsymbol{d}_k^{s_i} = \gamma_i(\boldsymbol{u}_1) \tag{3.12}$$

$$\alpha_{ki} = \frac{< \boldsymbol{e}_i, \gamma_i(\boldsymbol{u}_1) >}{\|\gamma_i(\boldsymbol{u}_1)\|} \tag{3.13}$$

where $\gamma_i(\boldsymbol{u}_1) : \mathbb{R}^{p_k} \rightarrow \mathbb{R}^{q_i}$ is a function that allows to represent $\boldsymbol{u}_1$ w.r.t. the sibling referential $\boldsymbol{d}_k^{s_i}$. It consists first to rotate $\boldsymbol{u}_1$ such that to have the angle between $\gamma_i(\boldsymbol{u}_1)$ and $\boldsymbol{d}_k^{s_i}$ the same as between $\boldsymbol{u}_1$ and $\boldsymbol{d}_k$, then aligns the rotated $\boldsymbol{u}_1$ to $\boldsymbol{d}_k^{s_i}$ based on the projector Δ_{ik}. Figure 3.6 illustrates the effect of γ_i. Finally, Figure 3.7 shows the whole transformation process for the update of $\boldsymbol{d}_k$ and its sibling atoms.

The Algorithm 8 summarizes the main steps described above of the time warp invariant kSVD (TWI-kSVD). Similar to kSVD, once the K atoms and the corresponding siblings are updated, the sparse coding and dictionary learning steps are iterated until convergence (i.e., desirable reconstruction error or maximal number of iteration reached). It is worth to say that the time complexity of each iteration for TWI-kSVD is $\mathcal{O}(q^2 N\tau K)$ with K being the dictionary size, q maximum time series length and N number of training time series.

[3] It is important to notice that there exists an inherent ambiguity in the sign of singular vectors resulting from SVD decomposition, i.e., their sign is not defined and the individual singular vectors have an arbitrary sign. This becomes an issue as the proposed COSTW algorithm is sensitive to the sign of the vectors. Bro et al. [Bro 08] addressed this problem and proposed a sign disambiguation technique. They provide an analytic solution for this problem which is, for each singular vector the corresponding sign is flipped if it is not similar to the sign of the majority of vectors it is representing. The same technique is used for the sign correction throughout the experiments.

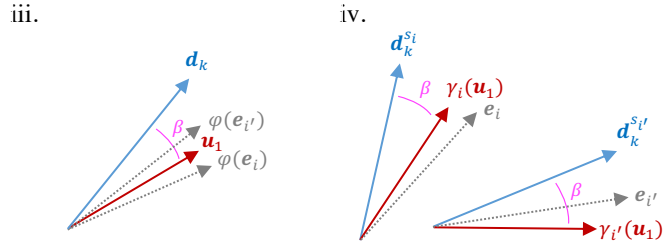

Figure 3.6: γ_i: representation of $\boldsymbol{u}_1$ singular vector w.r.t the sibling referential $\boldsymbol{d}_k^{s_i}$ and $\boldsymbol{d}_k^{s_{i'}}$.

Algorithm 8 Time Warp Invariant kSVD (TWI-kSVD)

Input: $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$ $(\boldsymbol{x}_i \in \mathbb{R}^{q_i})$, $D = \{\boldsymbol{d}_j\}_{j=1}^{K}$ $(\boldsymbol{d}_j \in \mathbb{R}^{p_j})$, τ.
Output: A, $\boldsymbol{\Delta}$, D.

repeat
{Sparse coding step:}
  for $i = 1, ..., N$ **do**
    $(\boldsymbol{\alpha}_i, \{\Delta_{ij}\}_{j=1}^{K}\}) =$TWI-OMP$(\boldsymbol{x}_i, D, \tau)$
  end for
{Dictionary update step:}
  for $k = 1, ..., K$ **do**
    Set $\boldsymbol{\omega}_k = \{i \,|\, \alpha_{ki} \neq 0,\ i = 1, ..., N\}$ (the set of samples involving $\boldsymbol{d}_k$)
    Estimate $\varphi(\boldsymbol{e}_i)$ for $i \in \boldsymbol{\omega}_k$ by using Eqs. 3.9 and 3.10
    Apply an SVD on $E_k[\varphi(\boldsymbol{e}_i)]_{i \in \boldsymbol{\omega}_k}$ to estimate $\boldsymbol{u}_1$
    Update $\boldsymbol{d}_j = \boldsymbol{u}_1$ and α_{ki}, $\boldsymbol{d}_k^{s_i}$ for $i \in \boldsymbol{\omega}_k$ by using Eq. 3.12 and Eq. 3.13
  end for
until Convergence (stopping rule)

3.3.2 TWI-kSVD for time series classification

Given a set of labeled time series and a query time series $\boldsymbol{y}$, the aim of time series classification is to assign it a class label that has the most similarity to it. The challenge is to discover the discriminative features between classes to help the assignment process. We discussed several supervised dictionary learning methods in Chapter 2, in this section we provide a procedure for classifying time series based on the proposed dictionary learning approach.

The dictionary learned by TWI-kSVD is optimized to have the best reconstruction of the input time series. The training is done in an unsupervised manner and since the discriminative information of the inputs are not considered during the learning,

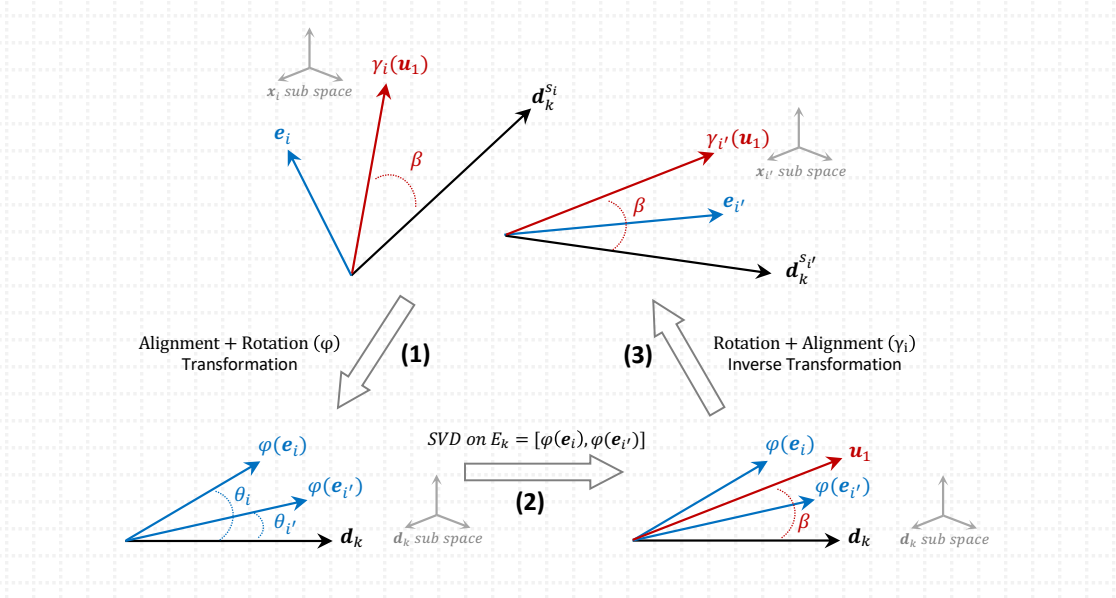

Figure 3.7: The transformation process for the update of $\boldsymbol{d}_k$ and its sibling atoms.

the learnt time series atoms are not optimal for the classification task. Based on the idea of the SRC, we proposed a time series classification procedure. Although unlike SRC, instead of directly using the input time series as dictionary, we learn a class-specific dictionary for each class of the data separately. All these sub-dictionaries are then concatenated to form the global dictionary for SRC.

Denote by C the number of classes and X_c the subset of all the training time series from the class c, a class-specific dictionary D_c is learned from the samples in X_c using TWI-kSVD Algorithm 8. The individual learned dictionaries D_c are then put together to form a global dictionary as $D_G = [D_1, ..., D_C]$. All columns of D_G are normalized to have a unit l_2 norm and attached with a class label. The classification of a new arrival time series $\boldsymbol{y}$ is done similar to SRC as first we solve the following sparse coding problem to find $\boldsymbol{\alpha}$ and $\Delta = \{\Delta_j\}_{j=1}^{K_G}$:

$$\begin{aligned} & \min_{\boldsymbol{\alpha}, \Delta} \|\boldsymbol{y} - \sum_{j=1}^{K_G} \Delta_j \, \boldsymbol{d}_j^G \, \alpha_j\|_2^2 \\ s.t. \quad & \|\boldsymbol{\alpha}\|_0 \leq \tau, \\ & \Delta_j \in \{0,1\}^{q \times p_j}, \; \Delta_j \, \mathbf{1}_{p_j} = \mathbf{1}_q \end{aligned} \tag{3.14}$$

where $\boldsymbol{d}_j^G$ is the j^{th} column of D_G and K_G is the number of atom time series in D_G. The solution of Eq. 3.14 can be found by TWI-OMP (Algorithm 6).

After the sparse coding step, the classification will be based on the minimum class-wise reconstruction error. i.e., assign $\boldsymbol{y}$ to the c^{th} class such that:

$$c^* = \arg\min_c \|\boldsymbol{y} - \sum_{j \in \nu_c} \Delta_j \, \boldsymbol{d}_j^G \, \alpha_j\|_2^2 \tag{3.15}$$

where ν_c is a set of indices corresponding to the atoms of the c^{th} class.

It is worth noting to mention, since the sub-dictionaries are learned independently from each other, these class-specific dictionaries usually share some common atoms that may degrade the classification accuracy. An optional step to improve the algorithm, as suggested by [Ramirez 10], is pruning the sub-dictionaries before putting them together. The pruning can be done easily, as if the similarity between a pair of atoms from two different sub-dictionaries exceeds a predefined threshold, the both atoms are discarded from the global dictionary.

3.3.3 Time warp invariant dictionary learning by gradient descend (TWI-GDDL)

We provide an alternative solution to the dictionary learning problem in Eq. 3.8 which is based on a gradient descent optimization. Let us recall the dictionary learning formalization under time warp as:

$$\begin{aligned} \min_D \mathcal{J}(D) \quad &= \quad \sum_{i=1}^{N} \|\boldsymbol{x}_i - \sum_{j=1}^{K} \Delta_{ij} \, \boldsymbol{d}_j \, \alpha_{ji}\|_2^2 \\ s.t. \quad &\forall j \;\; \|\boldsymbol{d}_j\|_2 = 1 \end{aligned} \tag{3.16}$$

where, the sparse coefficients $A = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_N]$ and alignment matrices $\boldsymbol{\Delta} = \{\Delta_{ij}\}$, $i = 1, ..., N; j = 1, ..., K$ are assumed to be learned by TWI-OMP (Algorithm 6). The above problem is equivalent to:

$$\begin{aligned} \min_D \mathcal{J}(D) \quad &= \quad \sum_{i=1}^{N} \sum_{t=1}^{q_i} (x_{it} - \sum_{j=1}^{K} \alpha_{ji} \sum_{(t,t') \in \boldsymbol{\pi}_{ij}^*} d_{jt'})^2 \\ s.t. \quad &\forall j \;\; \|\boldsymbol{d}_j\|_2 = 1 \end{aligned} \tag{3.17}$$

where x_{it} is the t^{th} time instant of $\boldsymbol{x}_i$ and $\boldsymbol{\pi}_{ij}^*$ denotes the optimal alignment path between $\boldsymbol{x}_i$ and $\boldsymbol{d}_j$. To resolve the Eq. 3.17, We propose a gradient descend method

Algorithm 9 Time Warp Invariant Gradient Descend Dictionary Learning (TWI-GDDL)

Input: $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$ ($\boldsymbol{x}_i \in \mathbb{R}^{q_i}$), $D = \{\boldsymbol{d}_j\}_{j=1}^{K}$ ($\boldsymbol{d}_j \in \mathbb{R}^{p_j}$), τ.
Output: A, $\boldsymbol{\Delta}$, D.
repeat
{Sparse coding step:}
  for $i = 1, ..., N$ **do**
    $(\boldsymbol{\alpha}_i, \{\Delta_{ij}\}_{j=1}^{K}\}) =$TWI-OMP$(\boldsymbol{x}_i, D, \tau)$
  end for
{Dictionary update step:}
  for $k = 1, ..., K$ **do**
    Set $\boldsymbol{\omega}_k = \{i \,|\, \alpha_{ki} \neq 0,\ i = 1, ..., N\}$ (the set of samples involving $\boldsymbol{d}_k$)
    Update $\boldsymbol{d}_k$ using Eqs. 3.18 and 3.19.
  end for
until Convergence (stopping rule)

based on the following update rule at iteration m, for the atom $\boldsymbol{d}_k$:

$$\begin{aligned} \forall t', 1 \leq t' \leq p_k : \ d_{kt'}^{m+1} &\leftarrow d_{kt'}^{m} - \eta^m \frac{\partial \mathcal{J}}{\partial d_{kt'}^{m}} \\ \boldsymbol{d}_k^{m+1} &\leftarrow \frac{\boldsymbol{d}_k^{m+1}}{\|\boldsymbol{d}_k^{m+1}\|_2} \end{aligned} \tag{3.18}$$

where η^m is the learning rate at iteration m. With the partial derivative equation $\forall t', 1 \leq t' \leq p_k$:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial d_{kt'}} = \sum_{\substack{\boldsymbol{x}_i \\ i \in \boldsymbol{\omega}_k}} \sum_{t=1}^{q_i} -2\alpha_{ki}(x_{it} - \alpha_{ki} d_{kt'} - \alpha_{ki} \sum_{\substack{(t,t'') \in \boldsymbol{\pi}_{ik}^* \\ (t'' \neq t')}} d_{kt''} \\ - \sum_{j \neq k} \alpha_{ji} \sum_{(t,t'') \in \boldsymbol{\pi}_{ij}^*} d_{jt''}) \end{aligned} \tag{3.19}$$

where $\boldsymbol{\omega}_k = \{i \,|\, \alpha_{ki} \neq 0,\ i = 1, ..., N\}$ is the set of samples involving $\boldsymbol{d}_k$. The proof of the above proposition is given in Appendix B. Algorithm 9 summarize the entire sparse coding and TWI-GDDL dictionary learning procedure.

3.3.4 Dictionary learning for time series clustering

In this section we address the problem of time series clustering under sparse coding and dictionary learning framework, where both input samples and atoms define

time series that may involve varying delays and be of different lengths. Time series clustering aims to find homogeneous groups of input data where the within group similarity is minimized and the between group similarity maximized. Clustering of unlabeled time series provides useful information on the underlying structure of the groups.

Denote by $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$ as before, a set of N input time series, we formalize the problem of time series clustering under the sparse coding and dictionary learning setting as the estimation of: a) the partition $C = \{C_l\}_{l=1}^{K}$ of X into K clusters and b) the K sub-dictionaries $\{D_l\}_{l=1}^{K}$, to minimize the inertia goodness criterion and the error of reconstruction as:

$$\min_{C,D} \sum_{l=1}^{K} \sum_{\boldsymbol{x}_i \in C_l} E(\boldsymbol{x}_i, D_l) \tag{3.20}$$

where $D_l = \{\boldsymbol{d}_j^l\}_{j=1}^{K_l}$ the sub-dictionary of C_l is composed of K_l time series atoms $\boldsymbol{d}_j^l \in \mathbb{R}^{p_j}$. $E(\boldsymbol{x}_i, D_l)$ the error of the reconstruction, under time warp, of $\boldsymbol{x}_i$ based on the sub dictionary $D_l = \{\boldsymbol{d}_j^l\}_{j=1}^{K_l}$ is formalized as:

$$\begin{aligned} E(\boldsymbol{x}_i, D_l) = \quad & \min_{\boldsymbol{\alpha}_i} \|\boldsymbol{x}_i - \mathcal{F}_i(D_l)\boldsymbol{\alpha}_i^l\|_2^2 \\ s.t. \quad & \|\boldsymbol{\alpha}_i^l\|_0 \leq \tau. \end{aligned} \tag{3.21}$$

where $\mathcal{F}_i(D_l) = [f_i(\boldsymbol{d}_1^l), ..., f_i(\boldsymbol{d}_{K_l}^l)] \in \mathbb{R}^{q_i \times K_l}$ is the transformation of D_l to a new dictionary composed of warped atoms $f_i(\boldsymbol{d}_j^l) \in \mathbb{R}^{q_i}$ aligned to $\boldsymbol{x}_i$ to deal with the involved delays w.r.t $\boldsymbol{x}_i$. $\boldsymbol{\alpha}_i^l = (\alpha_{1i}^l, ..., \alpha_{K_l i}^l)^t$ is the sparse codes of $\boldsymbol{x}_i$ under D_l and τ the sparsity factor under the l_0 norm.

To resolve the clustering problem defined in Eq. 3.20, we use a two steps iterative refinement process, as in standard K-means clustering 1) cluster assignment 2) dictionary update. In the *cluster assignment* step, D_l's are assumed fixed and the problem remains to resolve the sparse coding based on the warped dictionaries $\mathcal{F}_i(D_l)$ defined in Eq. 3.21 which can be done based on TWI-OMP Algorithm 6. The cluster assignments are then obtained by assigning each $\boldsymbol{x}_i$ to the cluster C_l whose sub dictionary D_l minimizes the reconstruction error.

In the *dictionary update* step, the cluster assignments C_l, the learned sparse codes α_i^l and the alignments Δ_i^l are fixed and the problem in Eq. 3.20 defines a dictionary learning problem to minimize the clustering inertia criterion and represent sparsely samples within clusters:

$$\begin{aligned} &\min_{D} \sum_{l=1}^{K} \sum_{\boldsymbol{x}_i \in C_l} \|\boldsymbol{x}_i - \sum_{j=1}^{K_l} \Delta_{ij}^l \, \boldsymbol{d}_j^l \, \alpha_{ji}^l\|_2^2 \\ &s.t. \quad \|\boldsymbol{d}_j^l\|_2 = 1. \end{aligned} \tag{3.22}$$

This problem is then resolved as K single dictionary learning problems of the form:

$$\min_{D_l} \mathcal{J}_l \;=\; \sum_{\boldsymbol{x}_i \in C_l} \|\boldsymbol{x}_i - \sum_{j=1}^{K_l} \Delta_{ij}^l \, \boldsymbol{d}_j^l \, \alpha_{ji}^l\|_2^2 \tag{3.23}$$

to learn each sub-dictionary D_l that minimizes the inertia of the cluster C_l using TWI-GDDL update rules in Eqs. 3.18 and 3.19. Algorithm 10 summarized the entire clustering routine.

In the initialization step, a clustering (e.g., spectral clustering, affinity propagation) algorithm is performed on the COSTW Gram matrix S to determine an initial partition $\{C_l\}_{l=1}^K$ of X (line 1-3 of Algorithm 10). A sparse coding and a dictionary learning are then performed on the samples of each cluster to initialize the sub-dictionaries $\{D_l\}_{l=1}^K$ (line 4-10). Based on the initial partition $\{C_l\}_{l=1}^K$ and sub-dictionaries $\{D_l\}_{l=1}^K$, the cluster assignment step consists to perform a sparse coding of each input sample based on each dictionary D_l, then to assign it the cluster whose dictionary minimizes its reconstruction error (line 12-15). Subsequently, in the dictionary update step, the atoms $\boldsymbol{d}_j^l$ of each dictionary D_l are updated.

3.4 Summary

In this chapter, the time warp invariant sparse coding and dictionary learning problems are formalized and respective solutions are presented. First, we investigate the problem of time warp invariant kSVD (TWI-kSVD) where both input samples and dictionary atoms may have different lengths while involving varying delays. For the sparse coding problem, we propose a time warp invariant orthogonal matching pursuit based on a new cosine maximization time warp operator. For the dictionary

Algorithm 10 Time Warp Invariant Dictionary Learning Clustering (TWI-DLCLUST)

Input: $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$, the number of clusters K, τ.
Output: $\{C_1, ..., C_K\}$, $\{D_1, ..., D_K\}$
{Clustering Initialization:}
Define the affinity matrix $S \in \mathbb{R}^{N \times N}$ of general term:
$s_{ii'} = \text{COSTW}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$
Apply the affinity propagation (or spectral clustering) algorithm to cluster S into K clusters: $C_1, ..., C_K$.
{Sub-dictionary initialization:}
for $l = 1, ..., K$ **do**
  Initialize D_l randomly from C_l.
  repeat
    Sparse code each $\boldsymbol{x}_i \in C_l$: $[\boldsymbol{\alpha}_i^l, \Delta_i^l] = \text{TWI-OMP}(\boldsymbol{x}_i, D_l, \tau)$.
    Update D_l using TWI-GDDL update rules in Eqs. 3.18 and 3.19.
  until Convergence (stopping rule)
end for
repeat
{Cluster assignment:}
  Sparse code each $\boldsymbol{x}_i \in X$ based on each D_l $(l = 1, ..., K)$:
    $[\boldsymbol{\alpha}_i^l, \Delta_i^l] =$TWI-OMP$(\boldsymbol{x}_i, D_l, \tau)$
  Assign $\boldsymbol{x}_i$ to the cluster C_l whose D_l minimizes $E(\boldsymbol{x}_i, D_l)$:
    $C_l = \{\boldsymbol{x}_i \,/\, l = \min_{l'} \|\boldsymbol{x}_i - \sum_{l'=1}^{K_{l'}} \Delta_{ij}^{l'} \boldsymbol{d}_j^{l'} \; \alpha_{ji}^{l'}\|_2^2\}$
{Dictionaries update:}
  for $l = 1, ..., K$ **do**
    Update D_l using TWI-GDDL update rules in Eqs. 3.18 and 3.19.
  end for
until Convergence (no changes in cluster assignments)

learning, thanks to a rotation transformation between each atom and its sibling atoms, a singular value decomposition is used to jointly approximate the coefficients and update the dictionary. In the second part, a solution for the dictionary learning problem by time warp invariant gradient descent (TWI-GDDL) is studied. Finally, under the framework of sparse coding and dictionary learning, two algorithms are proposed for time series classification and clustering (TWI-DLCLUST).

Nothing ever becomes real 'til it is experienced.

— John Keats

4 Experimental Results

Contents

This chapter provides the experimental studies on sparse coding and dictionary learning for time series reconstruction, classification and clustering. The proposed approaches are evaluated and compared to major alternative methods on several public datasets, with a real application to DEEZER music data stream clustering. The dataset description is provided in Section 4.1. The evaluation of the proposed TWI-kSVD for time series classification and reconstruction is presented in Section 4.2. The proposed GDDL and TWI-DLCLUST for time series clustering are evaluated in Section 4.3.

4.1 Data description

We have considered in Table 4.1 two groups of datasets. The first group is composed of the top 14 datasets for which the classes, as well as the training and test sets are given. The four first datasets are composed of public multivariate time series that

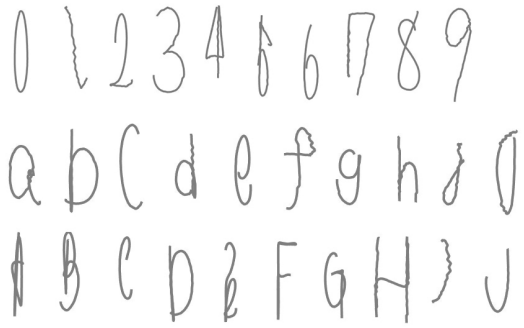

Figure 4.1: 6DMG time series behaviour. DIGIT (top), LOWER (middle), UPPER (bottom) classes.

have different lengths and involve varying delays. In particular, DIGITS, LOWER, and UPPER datasets give the description of 2-D air-handwritten motion gesture of digits, upper and lower case letters performed on a Nintendo (R) Wii device by several writers [Chen 12]. The CHAR-TRAJ dataset gives the 2-dimensional handwritten character trajectory performed on a Wacom tablet by the same user [A. Frank 10]. The ECG-MIT dataset was obtained from the MIT-BIH Arrhythmia [Goldberger 00] database where the heartbeats represented by QRS complexes. The 7 remaining datasets are composed of univariate time series of the same lengths that involve significant delays [Keogh 06]. Furthermore, we consider two challenging synthetic datasets BME and UMD [Soheily-Khah 16][1], where time series share local temporal features within the classes while being of distinctive global behaviour. Figure 4.1, 4.2 and 4.3 illustrate the time series behaviour of some considered time series. For instance, Figure 4.3 shows that time series may have variable lengths and delays within a same class ("e"), they may have different global behaviours as in the "UP" class of UMD dataset, while sharing only local events ("small bell") that may arise at different time stamps, or even have time series of different classes "0" and "6" that may share similar global behaviours.

The last two datasets are provided by DEEZER[2], the online music streaming service that offers access to the music content of nearly 40 million licensed tracks. DEEZER data, for which we have no ground truth, give the description of streaming data of music albums, randomly selected among 10^5 French user streams and

[1] http://ama.liglab.fr/~douzal/tools.html
[2] https://www.deezer.com/fr/

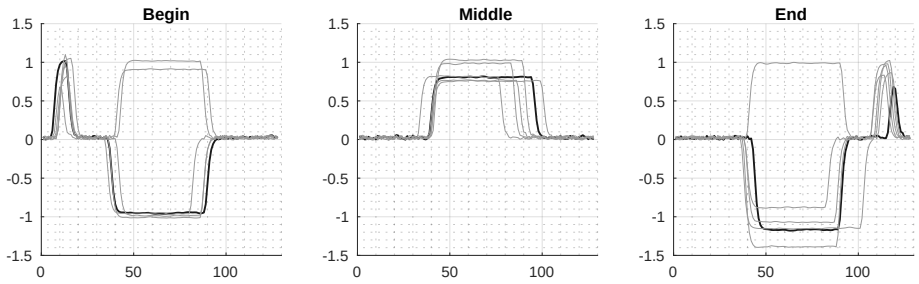

Figure 4.2: BME time series behaviour: "Begin", "Middle" and "End" classes.

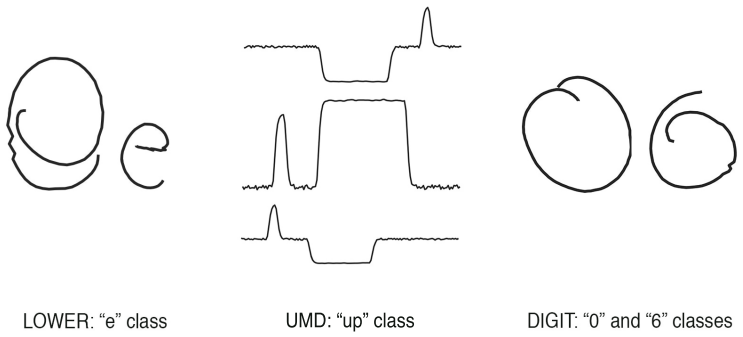

Figure 4.3: Time series characteristics within and between classes

recorded from October 2016 to September 2017. They are composed of univariate time series that give the daily total number of streams per album from its release date to September 2017; this study consider only the streams of a duration ≥ 30 seconds. In particular, DEEZER15 and DEEZER30 are provided for the streams analysis over the crucial early period after the album release date. They give the description of the prefix time series on the early period covering a cumulative number of 10^3 streams (in red in Figure 4.4b), where 10^3 is the median of the total streams for the top 25% of the albums (Figure 4.4a). In addition, for the pertinence of the analysis, the prefix time series of length < 7 days are extended to 15 days in DEEZER15 and to 30 days in DEEZER30.

4.2 TWI-kSVD for time series classification

In this section, we evaluate the relevance of the proposed TWI-kSVD approach in a classification context. For that, the sparse representation based classification (SRC) schedule (Sections 2.2.4 and 3.3.2) is used. For a given dictionary learning

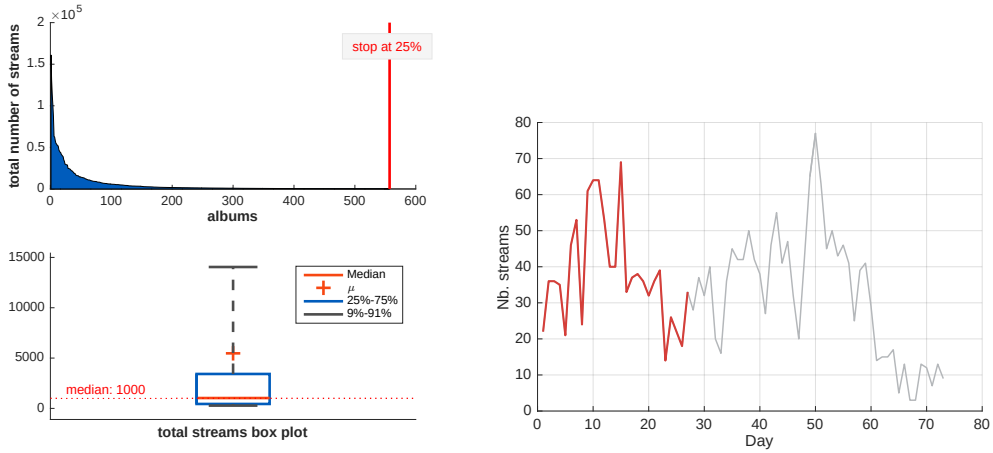

Figure 4.4: DEEZER dataset (a) Albums versus total number of streams (top), box plot of the values (bottom). (b) An album streaming time series, in red the prefix time series covering a cumulative number of 10^3 streams.

method, SRC process consists first to learn one dictionary per class, then to form one global dictionary by concatenating the dictionaries learned for all the classes. The global dictionary is then used to sparse code test samples. Finally, based on both the estimated sparse coefficients and the dictionaries learned for the classes, the test samples are assigned to the class whose dictionary yields to the minimum reconstruction error.

The proposed approach TWI-kSVD is compared to five major dictionary learning methods. The considered alternative dictionary learning methods deploy different strategies to address varying lengths and delays issues in time series data. First, we consider the Shift Invariant Sparse Coding (SISC) [Grosse 07], a convolved dictionary learning method, that learns basis functions as well as the time offsets and shifts to sparse code input time series. Then we consider kSVD [Aharon 06], MOD as LASSO-DL$_1$ [Engan 99] and Metaface Learning as LASSO-DL$_2$ [Yang 10] where time series are zero padded to render them of the same length, while the delay aspect is simply ignored. For LASSO-DL$_1$ and LASSO-DL$_2$, the sparse coding stage uses l_1 instead of l_0 norm, whereas in the dictionary learning step, all the atoms are learned at once by solving a pseudo-inverse problem in LASSO-DL$_1$ and updated one atom at a time by using Lagrangian solver in LASSO-DL$_2$. Finally, we consider the kernel kSVD (κ-kSVD) [Chen 15], where both varying length and delays are circumvent by using a Gaussian DTW kernel. The description of all these methods are given in Chapter 2.

Table 4.1: Data description

Dataset	Nb. class	Train size	Test size	Length range	Domain
DIGITS	10	100	300	29-218	Sensor
LOWER	26	260	430	27-163	Sensor
UPPER	26	260	520	27-412	Sensor
CHAR-TRAJ	20	200	200	109-205	Sensor
ECG-MIT	4	40	160	541	Medical
CBF	3	30	900	128	Synthetic
FACEFOUR	4	24	88	350	Image
LIGHTNING2	2	60	61	637	Sensor
LIGHTNING7	7	70	73	319	Sensor
CC	6	300	300	60	Synthetic
TRACE	4	100	100	275	Sensor
ECG200	2	100	100	96	Medical
UMD	3	36	144	150	Synthetic
BME	3	30	150	128	Synthetic
DEEZER15	-	-	281	15-301	Music
DEEZER30	-	-	278	30-301	Music

Table 4.2: Table of parameters - classification methods

	Para.	Range Val.	Description
LASSO-DL$_1$	λ	$[0.1, 1]$ lag of 0.1	Regularization
LASSO-DL$_2$	λ	$[0.1, 1]$ lag of 0.1	Regularization
SISC	β	$[0.05, 5]$ lag of 0.05	Regularization
	p	$\{50, 70, 90\}$	patch size
	q	$\{20, 40, 60\}$	atom size
TWI-kSVD	sc	$[0, 100]$ lag of 10	Sakoe-Chiba band

Validation protocol

For each method, the related parameters Table 4.2 are learned by a grid search on a validation set for handwritten characters of 6DMG and CHAR-TRAJ datasets. For the small datasets BME and UMD, a 1-fold cross-validation is deployed. Our method does not require the setting of many parameters and the only thing that we need to fix is the warping band (SC). The best configuration of parameters is then used to estimate the accuracy on the test set. This process is reiterated 10-times and the obtained average error-rates are provided in Table 4.3. Note that, we have developed the algorithms of kSVD and LASSO-DL$_1$ and use the codes

Table 4.3: Classification error rates

	LASSO-DL$_1$	LASSO-DL$_2$	kSVD	SISC	κ-kSVD	TWI-kSVD
DIGIT	0.63	0.31	0.12	0.75	*0.02*	**0.01**
LOWER	0.64	0.43	0.29	0.79	**0.02**	0.07
UPPER	0.69	0.26	0.26	0.86	**0.06**	*0.09*
CHAR-TRAJ	0.34	0.05	**0.02**	0.78	0.09	*0.03*
CBF	0.29	0.13	0.20	0.75	**0.00**	*0.01*
FACEFOUR	0.39	0.14	0.35	0.75	**0.11**	**0.11**
LIGHTNING2	0.43	0.26	0.38	0.45	0.23	**0.16**
LIGHTNING7	0.50	0.39	0.47	0.71	**0.21**	**0.21**
CC	0.67	0.50	0.52	0.54	**0.01**	*0.04*
TRACE	0.33	0.23	0.27	0.30	*0.01*	**0.00**
ECG200	0.62	**0.10**	0.15	0.34	0.18	0.18
UMD	0.77	0.42	0.57	0.45	0.04	**0.01**
BME	0.80	0.24	0.59	0.42	0.03	**0.00**
Nb. Best	0	1	1	0	*6*	**7**
Avg. Rank	5.38	3.04	3.58	5.46	*1.88*	**1.65**

provided for LASSO-DL$_2$[3], SISC[4] and κ-kSVD[5]. For all the methods, a dictionary of size $K = 10 \times$ *the number of classes* is initialized randomly from the training set expect for SISC that is initialized in the provided code as Haar basis functions. Table 4.3 gives, for ($\tau = 2$) sparsity level, the error-rates obtained by using each dictionary learning method to classify the time series datasets. The best result is indicated in bold and the italic values reference the performances that are not significantly different from the best value (t-test at 5% risk). In addition, for each dataset, the performances obtained by the methods are first ranked, then the average ranking of each method is reported at the end of the Table (Avg. Rank). The best ranking (i.e., the lowest one) is indicated in bold, and the italic values show the average ranking non significantly different from the best (Wilcoxon matched-pairs ranks test at 5% risk).

[3] https://goo.gl/B5sKMc
[4] https://goo.gl/HirU4P
[5] https://goo.gl/j6nrzz

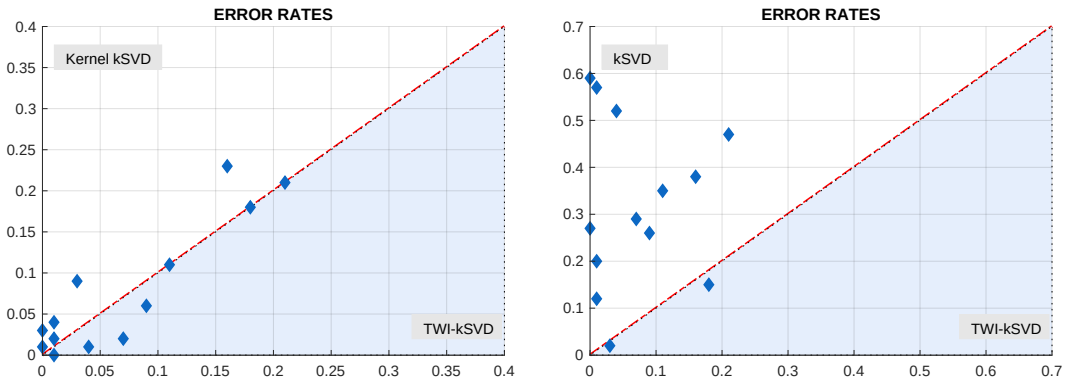

Figure 4.5: Error rates comparison. TWI-kSVD versus kernel-kSVD (Left) and TWI-kSVD versus kSVD (Right).

Results and discussion

From Table 4.3, we can see that the good performances are obtained by TWI-kSVD followed closely by κ-kSVD. In particular, the best results are reached by TWI-kSVD with a total number of best values (Nb. Best) of 7 and an average ranking (Avg.Rank) of 1.65. Reasonable results are obtained for LASSO-DL$_2$ and kSVD, with good performances for CHAR-TRAJ and ECG200 datasets and slightly better results for LASSO-DL$_2$ than kSVD for most of the datasets. The weak results are obtained for LASSO-DL$_1$ and SISC. The error rates comparison plots also provided in Figure 4.5 that compares two-by-two the error rates of the alternative methods κ-kSVD and kSVD to the proposed TWI-kSVD. From the right sub-figure, we can see that the TWI-kSVD suppresses the kSVD in almost all datasets except two. The left sub-figure, on the other hand, shows that κ-kSVD leads to very close results to our method except for six datasets where the proposed method brings better results than κ-kSVD.

We first must note that the methods LASSO-DL$_1$, LASSO-DL$_2$ and kSVD that use zero-padding, to circumvent the problem of variable lengths and delays for the first group of datasets (DIGIT, LOWER, UPPER, CHAR-TRAJ), lead to lower performances than TWI-kSVD and κ-kSVD that use appropriate approaches to deal with time series under time warp. For SISC, the problem of time warp is addressed by segmenting each input time series into small parts of the same length, then a dictionary is learned to sparse represent the small parts. Although the learned dictionary shows a good ability to reconstruct time series of each class, it fails under SRC classification. The main reason is due to the learned dictionary, that is composed of basis atoms

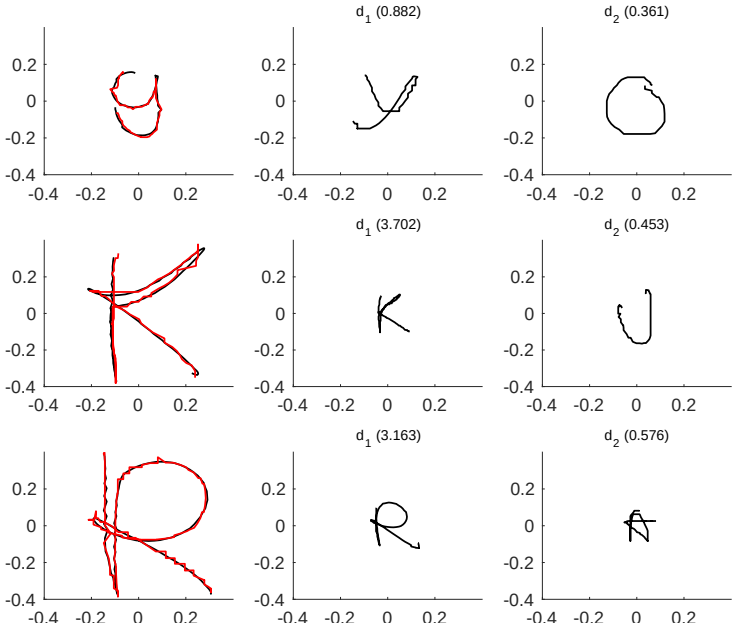

Figure 4.6: TWI-kSVD: The first column gives the input time series (in black) and their reconstruction (in red). The second and third columns show the two first atoms used for the reconstruction.

that are not discriminative nor specific to each class and instead are commonly shared by the dictionaries of all the classes. Thus, SISC method seems appropriate to reconstruct time series under time warp not for their classification.

For κ-kSVD, the variable lengths and delays are addressed by using a time warp kernel. It leads, similarly to TWI-kSVD, to good classification results. As a kernel-based machine, input samples are mapped into a higher dimensional space (Hilbert feature space), where the dictionary is learned and samples sparse coded as a non linear combination of the basis atoms. Sparse coding based on non linear combination functions allows a more precise representation than when relying on linear combination functions, as used by the other studied methods. However, as the sparse codes and the learned dictionary are processed into the feature space, there is no explicit description of the sparse representations nor of the learned atoms into the input space, which constitutes a major limitation for κ-kSVD. κ-kSVD seems to be a powerful and efficient method for time series classification under time warp, but not as a dictionary learning method with a main purpose to extract the basic primitives that discriminate the classes, to be used for any further learning tasks into the input space.

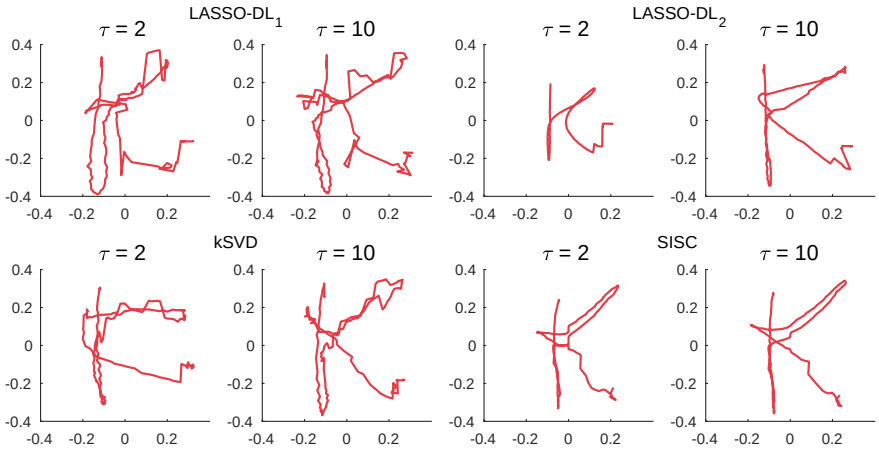

Figure 4.7: The reconstruction of the letter "K" at ($\tau = 2$) and ($\tau = 10$).

Reconstruction, convergence and τ effect

Let us underline that TWI-kSVD relies on a linear combination of the basis atoms for data reconstruction and beside having the best classification performances on almost all datasets, it is also a powerful reconstruction method. For instance, Figure 4.6 shows for several time series, their reconstruction (in red) based on the two first learned atoms. In particular, we can see that the first learned atom reveals the latent character that characterizes the class, whereas the second atom contributes to reconstruct the residual to best fit the input sample. In addition, we can see that when TWI-kSVD requires only two atoms ($\tau = 2$) to reconstruct precisely the letter "K" (Figure 4.6), the alternative methods need 10 atoms ($\tau = 10$) to reach a reasonable reconstruction (Figure 4.7). Note that, there is no visualization for κ-kSVD as there is no explicit descriptions into the input space.

We are also evaluating the convergence behaviour of the proposed algorithm on a few datasets. The goal of this experiment is to show how the objective function value is decreasing at each iteration and if it is become steady. As the objective function is based on the reconstruction of the samples, we compute the average reconstruction error $\sum_{i=1}^{N_c} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$ where N_c is the number of samples in each class and $\hat{\boldsymbol{x}}$ is the reconstructed instance of $\boldsymbol{x}$, at each epoch. We show in Figure 4.8 the convergence curves of the proposed TWI-kSVD algorithm on four data sets (CBF, FACEFOUR, CC, TRACE) as examples. From this figure, it can be

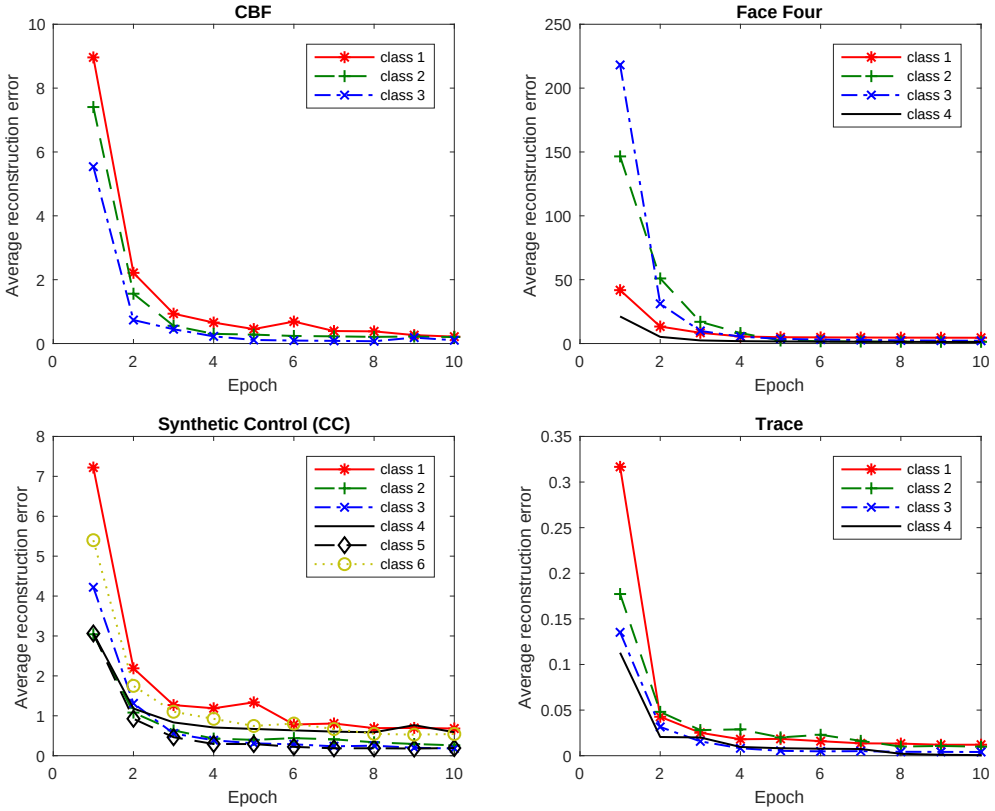

Figure 4.8: the convergence curves of the proposed TWI-kSVD learning algorithm.

concluded that the TWI-kSVD learning algorithm, for all the four dataset and each class of the data, can converge within less than ten epochs. A similar behaviour is observed for the rest of the datasets.

In the final experiment, the effect of the sparsity value (τ) on the input reconstruction is shown. Since the proposed method is an extension of the standard kSVD, this method is used for a comparison in the experiment. We evaluate, in this experiment, how higher sparsity level yields to better reconstruction or probably better classification accuracy. We use the synthetic control dataset (CC) as an example. Note that, this time the average reconstruction error is calculated on the Test dataset. Figure 4.9 (Left) shows for several sparsity levels the average reconstruction error for both kSVD and TWI-kSVD. It can be seen that the proposed method has very good reconstruction ability even with high sparsity level. An example time series and the related reconstructed outputs based on the both algorithms are provided in Figure 4.9 (Right). It is clear that, at the sparsity level

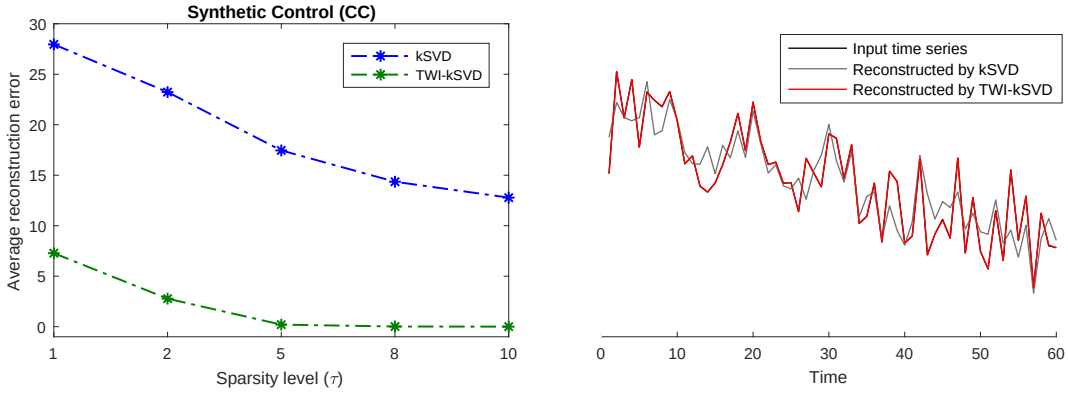

Figure 4.9: Left: The average reconstruction error versus the sparsity level (τ), Right: An example time series from CC dataset and its reconstruction ($\tau = 10$).

($\tau = 10$), our method reach to nearly perfect reconstruction while kSVD still cannot produce a reasonable output. The very poor reconstruction performance of kSVD with ($\tau = 2$) clarifies also its weak classification performances in Table 4.3.

We also show the reconstruction evolution of a sample letter (letter "a") from LOWER dataset in Figure 4.10. It shows for different learning iterations the reconstructed time series. It can be seen that after 9 iterations a reasonable reconstruction is achieved in this case.

4.3 TWI-DLCLUST for time series clustering

In the following section, we study the proposed TWI-DLCLUST clustering method ability to identify the underlying structure in different classes of time series even with varying lengths. We conduct several experiments to evaluate the efficiency of the proposed clustering framework in a comparison to the major alternative methods namely, the subspace sparse clustering (SSC) (Section 2.2.5) and the dictionary learning with structured incoherence (DLSI) (Section 2.3.4). For SSC, two variants SSC-BP [Elhamifar 13] and SSC-OMP [You 16] are studied for a sparse coding under l_0 and l_1 norms, where an orthogonal matching pursuit and a basis pursuit methods are used respectively. For DLSI, both sample-based (DLSI-S) and atom-based (DLSI-A) affinity matrix initialization proposed in [Ramirez 10] are studied. The Matlab codes of these methods are available online [6].

[6] SSC-OMP: https://goo.gl/8QrfBm, SSC-BP: https://goo.gl/bck6tS and DLSI: https://goo.gl/APbSwA.

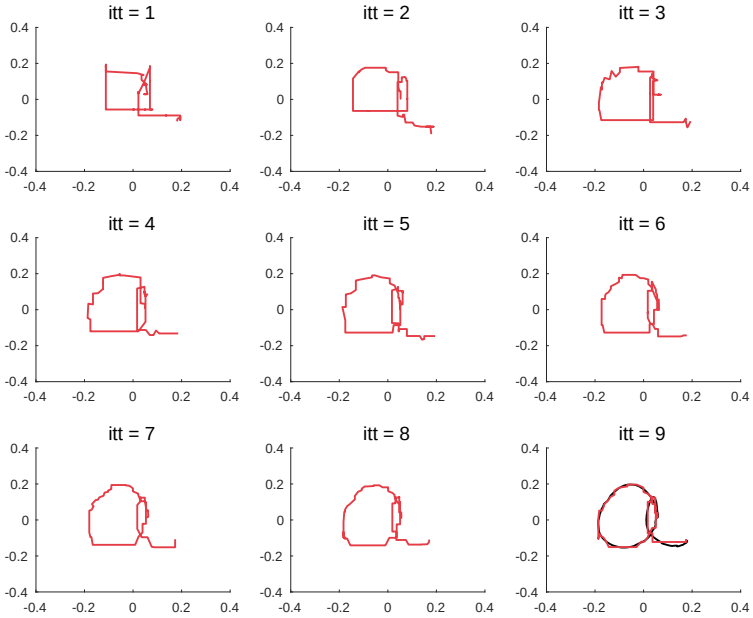

Figure 4.10: The reconstruction of the letter "a" at different learning iterations of the TWI-kSVD.

Table 4.4: Parameter Line/Grid values

Method	Line / Grid values	Desc.
SSC-OMP	$\tau \in \{1, 2, 3, 4, 5\}$	l_0 sparsity threshold
SSC-BP	$\lambda \in \{0.001, 0.01\}$, lag of 0.01	l_1 sparsity regularization
DLSI	$\lambda \in \{0.001, 0.01\}$, lag of 0.01	l_1 sparsity regularization
	$\eta \in \{0, 0.1, 0.01\}$	dictionary incoherence regularization
	$K_l = 5, \forall\, l \in \{1, ..., K\}$	Sub-dictionary D_l size
TWI-DLCLUST	$sc \in [0, 100]$, lag of 10	Sakoe-Chiba band width
	$\tau \in \{1, 2, 3, 4, 5\}$	l_0 sparsity threshold
	$K_l = 5, \forall\, l \in \{1, ..., K\}$	Sub-dictionary D_l size

Validation protocol

For the top 14 datasets in Table 4.1, for which the ground truth partition is known, the proposed method TWI-DLCLUST as well as the alternative clustering approaches are applied to cluster the data. For alternative methods, time series of different lengths are *a priori* zero padded. The adjusted Rand index [Rand 71] is then used to evaluate the goodness of the obtained clusters. The Rand index lies between 0 and 1, it measures the agreement between the obtained clusters and the ground truth ones.

Table 4.5: Adjusted Rand index

Dataset	SSC-OMP (τ)	SSC-BP	DLSI-S	DLSI-A	TWI-DLCLUST (τ)
DIGITS	0.839 (2)	0.856	0.854	0.841	**0.940** (1)
LOWER	0.935 (3)	0.943	0.937	0.934	**0.970** (1)
UPPER	*0.940* (2)	**0.942**	*0.940*	*0.938*	**0.942** (1)
CHAR-TRAJ	0.947 (5)	*0.977*	**0.978**	*0.971*	*0.965* (3)
ECG-MIT	0.327 (2)	*0.789*	*0.772*	*0.773*	**0.792** (2)
CBF	0.558 (2)	0.668	0.599	0.601	**0.770** (2)
FACEFOUR	**0.810** (5)	0.722	0.767	0.769	0.776 (3)
LIGHTNING2	**0.559** (2)	**0.559**	**0.559**	0.519	**0.559** (2)
LIGHTNING7	*0.793* (2)	*0.808*	0.724	0.747	**0.814** (3)
CC	0.736 (5)	0.630	0.813	0.791	**0.910** (1)
TRACE	0.680 (5)	0.752	0.755	0.753	**0.805** (1)
ECG200	0.547 (4)	0.631	**0.689**	*0.664*	*0.653* (3)
Nb. Best	2	2	3	0	**9**
Avg. Rank	4.00	2.83	2.92	3.58	**1.67**

The higher the index, the better the agreement is. In particular, the maximum value of Rand index, 1, is reached when the obtained partitions and the ground truth ones are identical. For DEEZER datasets, the ground truth being unknown, a DTW-based within-class W_r ratio [7] is used. The lower the within-class ratio W_r, the better the clustering is. W_r is as well used to select the optimal number of clusters. Finally, the set of parameters related to each studied method, indicated in Table 4.4, are learned by line/grid search on the validation set, the best parameters are then used to perform the clustering on the evaluation set. The process is iterated over 10 runs and the averaged performances are reported in Tables 4.5 and 4.6.

Results and discussion

Table 4.5 gives for the datasets the obtained adjusted Rand index values. The best values are indicated in bold, the non significantly different ones from the best (t-test at 5% risk) are in italic and the remaining results are significantly different from the bold values. For the two l_0 sparse coding methods SSC-OMP and TWI-DLCLUST, the learned sparsity coefficient τ is given between brackets. The

[7] $W_r = \frac{\sum_{l=1}^{K} \sum_{\boldsymbol{x},\boldsymbol{y} \in C_l} \mathrm{DTW}(\boldsymbol{x},\boldsymbol{y})}{\sum_{\boldsymbol{x},\boldsymbol{y} \in X} \mathrm{DTW}(\boldsymbol{x},\boldsymbol{y})}$

two last rows give, over all the datasets, the number of times a method reaches the best value as well as its average ranking.

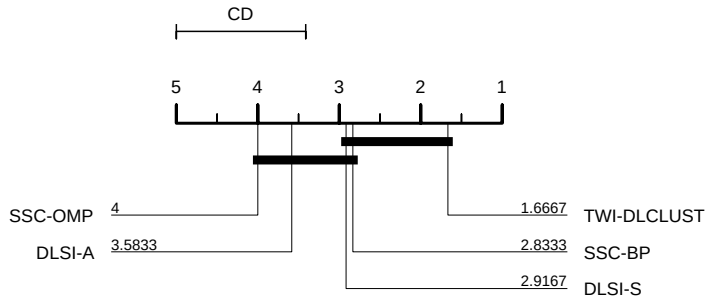

Figure 4.11: Comparison of the clustering methods against each other with the Nemenyi test.

From Table 4.5, we can see that the proposed TWI-DLCLUST reaches the best clustering results with 9 times (9 out of 12) as the best values, 2 times as significantly non different from the best and obtained the lowest average ranking. The second best results are obtained by SSC-BP and DLSI-S, followed by SSC-OMP. Figure 4.11 shows the statistically significant differences in the average rankings of the methods from Table 4.5 in the critical difference diagram for Nemenyi test [Nemenyi 62]. Methods that are not significantly different are connected. We can see that TWI-DLCLUST has the best performances with an average rank of 1.67 among the alternatives.

Although the l_1 sparse coding models (here SSC-BP and DLSI-S) are known to be more efficient than the l_0 models, TWI-DLCLUST even involving an l_0 sparse coding leads to the best results. While TWI-DLCLUST and DLSI-S involve smaller size sub-dictionaries ($K_l = 5$), SSC-OMP and SSC-BP are based on larger dictionary of the size of the evaluation set. Finally, by comparing the two l_0 sparse coding methods SSC-OMP and TWI-DLCLUST, we can see that TWI-DLCLUST leads for all datasets to sparser solutions with a lower or equal sparsity coefficient τ than SSC-OMP. Finally, note that SSC-OMP and SSC-BP lead to the lowest performances with a slightly better results for SSC-BP as using an l_1 norm sparse coding. These results may be partly explained by the fact that both SSC-OMP and SSC-BP are purely sparse coding methods based on one global dictionary fixed beforehand, unlike DLSI and TWI-DLCLUST that learn one sub-dictionary per cluster.

For DEEZER data we have performed each clustering method for several number of clusters and the within-class ratio of the obtained partitions reported in Table

Table 4.6: Within-class ratio W_r per number of clusters K.

Dataset	K	SSC-OMP ($\tau = 5$)	SSC-BP	DLSI-S	TWI-DLCLUST ($\tau = 2$)
DEEZER15	2	*0.266*	0.310	**0.245**	*0.262*
	3	0.201	0.253	0.177	**0.145**
	4	0.212	0.146	*0.114*	**0.112**
	5	0.188	*0.118*	*0.112*	**0.096**
	6	0.133	0.106	*0.074*	**0.069**
DEEZER30	2	0.339	0.348	0.322	**0.303**
	3	0.226	0.292	0.273	**0.173**
	4	0.175	0.241	*0.133*	**0.127**
	5	0.154	*0.119*	*0.110*	**0.096**
	6	*0.100*	**0.080**	*0.085*	*0.085*
Nb. Best		0	1	1	**8**
Avg. Rank		3.40	3.30	2.05	**1.25**

4.6. For simplicity, the DLSI approach is conducted only with DLSI-S variant, DLSI-A being highly equivalent in Table 4.5. We can see easily that, for both datasets and almost all the number of clusters, the best values are reached by TWI-DLCLUST, followed by the l_1 sparse code approaches SSC-BP and DLSI-S, then by SSC-OMP. In the second study, we analyze more closely the obtained clusters. For instance, based on Figure 4.12 that displays the progression of the within-class ratio w.r.t the number of clusters, a partitioning into four clusters is performed on DEEZER30. Accordingly, Figure 4.13, shows for each of the four clusters (each row), the profile of the medoids (in the first column), the closest albums to the medoid in the second column and at the third column, the atom that most contributes to sparse represent the cluster's samples.

The analysis of some characteristics of the albums within the clusters, allows us to bring additional explanations about the revealed clusters (Figure 4.13). These characteristics are whether the album is a "Full" (composed of several tracks) or a "Single" (just one track), a "Deluxe" edition (an old track which is updated with additional content) or not and the artist popularity before and after the album release date.

The first cluster is composed of 71% of "Full" albums and 15% of "Deluxe" editions. It corresponds to album releases with flat stream profiles. Such behaviour usually occurs when the content has already been published ("Deluxe" versions) or

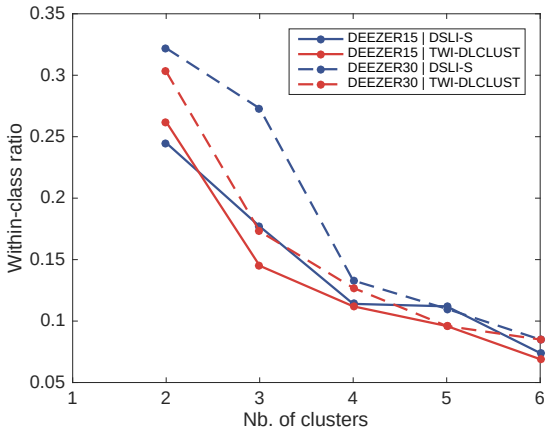

Figure 4.12: Number of clusters K vs. Within-class ratio W_r.

for not well-known artists, as assessed by the cluster medoid "Empereur du Sale" album of the rapper "Lorenzo" that released several singles a few weeks before the album release date and although not highly popular has still a steady fan base.

In the second cluster, 75% of the albums are "Single". The fast decrease stream profile just after the release date is not surprising for short albums (composed of 1-4 tracks). Indeed, a "Full" album is generally released shortly after the "Single" release, inducing a decrease of streams for the "Single" few weeks after its release. The cluster medoid "Ethologie" is produced by the rapper "Dehmo" that has not released albums since a long period, that may explain the burst of streams for the new content just after its release.

The third cluster is composed of 69% of "Full" albums mainly produced by artists that became popular after their album release. This is reflected by the stream profiles that initially evolve at low level then increase significantly several days/weeks after the album release. This is confirmed by the medoid album "Be Mine" a single produced by "Ofenbach" that was in fact revealed to the public with that album.

Finally, the cluster 4 comprises a majority of "Single" albums (84%) produced by popular artists with a huge fan base and immediate success. The medoid album "Divide" produced by "Ed Sheeran" was one of the biggest hits of 2017. Although the stream profiles of the clusters 4 and 2 seem similar, albums of the cluster 4 concern more established artists in their second/third albums while the cluster 2 is more related to emerging works and first successes.

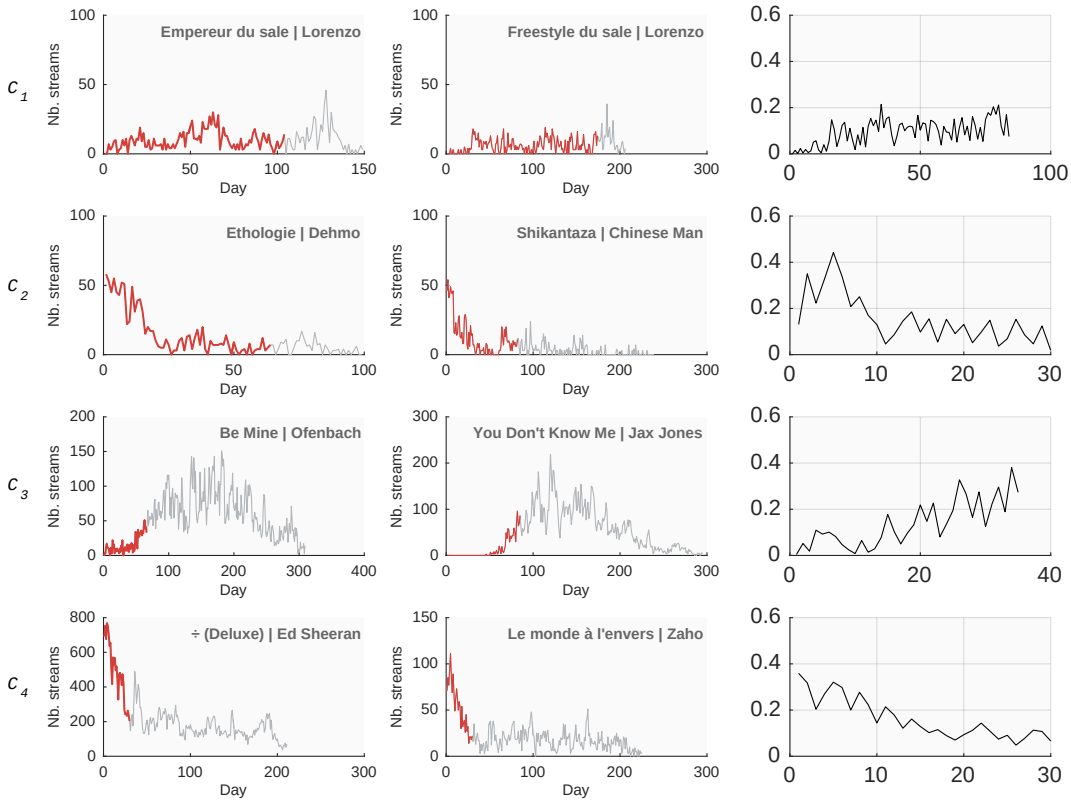

Figure 4.13: Four clusters partitioning of DEEZER30: Medoid profile (left column), Nearest album to the medoid (middle column), the most contributing atom to the cluster (right column).

The aim of the last study is to analyze the pertinence of the learned sub-dictionaries $\{D_1, ..., D_K\}$ for both DLSI and TWI-DLCLUST; the dictionary for the other methods SSC-OMP and SSC-BP is not learned but fixed beforehand. For that, for each method DLSI and TWI-DLCLUST, the atoms of the learned sub-dictionaries are first gathered together to built one global dictionary $\cup_{l=1}^{K} D_l$. Let us denote D_{G1} and D_{G2} the global dictionary obtained respectively for DLSI and TWI-DLCLUST. Subsequently, the samples in X are sparse coded, by using first an l_1 norm regularization based on D_{G1}, then a TWI-OMP based on D_{G2}.

For instance, for DEEZER30, DIGIT and CC, Figure 4.14 shows the learned sparse codes based on D_{G1} (on left) and on D_{G2} (on right), organized for interpretation purpose per clusters and per sub-dictionaries. It emerges from Figure 4.14, that sparse codes based on D_{G2} highlight clearly a block structure that reflects the

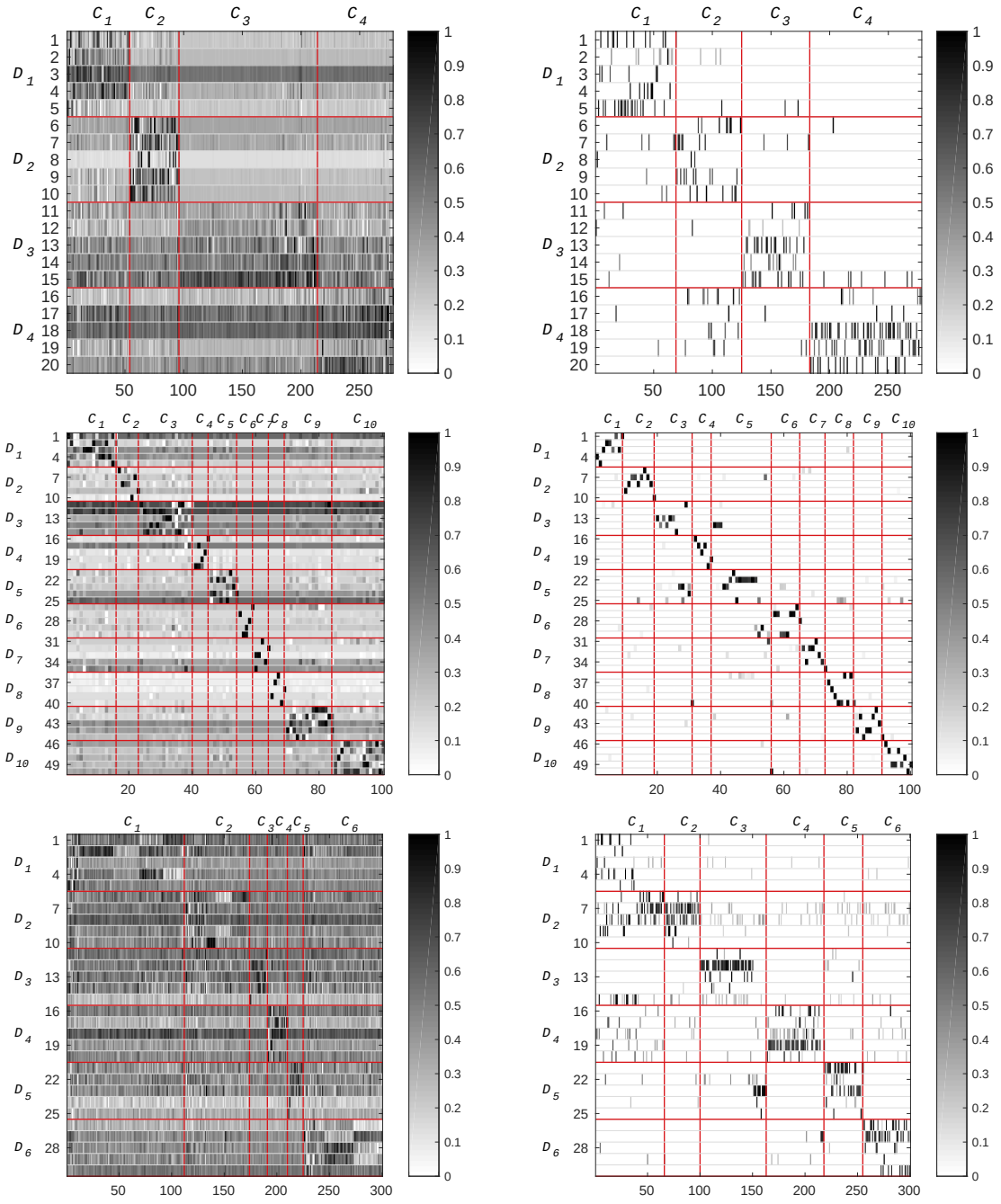

Figure 4.14: Sparse representations based on: D_{G1} learned by DLSI (left) and D_{G2} learned by TWI-DLCLUST (right). From top top bottom: DEEZER30, DIGIT and CC datasets.

discriminative performance of the sub-dictionaries composing D_{G2}; namely learned by TWI-DLCLUST. Indeed, sparse codes show that sub-dictionary D_l are mainly

involved to reconstruct samples of C_l. On the other hand, the structure of the sparse codes based on D_{G1} (learned by DLSI) seems much less sparser and less discriminative. In particular, for DEEZER30 we can note that atoms $d_3^1, d_{15}^3, d_{17}^4$ and d_{18}^4 reveal common primitives that are involved to reconstruct samples of all the clusters. Same structure can be seen in the learned sparse codes on D_{G1} (learned by DLSI) for the other datasets.

4.4 Summary

The performance of the proposed methods on several real and synthetic datasets were investigated. The first experiment was an application of TWI-kSVD for time series classification where the results are compared to the popular supervised dictionary learning methods. The conducted experiments and obtained results show that the proposed method lead to promising results compared to the state of the art dictionary learning methods. One can note that kernel-kSVD is a competitive method, in terms of classification accuracy. The proposed method has a major advantage over this method which is the ability to visualize the learned atoms. The dictionary for the kernel-kSVD is learned in the kernel space where there is no explicit information of the learned atom into the input space.

In the second experiment, we studied the performance of the suggested TWI-DLCLUST clustering approach. We discovered that the proposed dictionary learning for clustering method is capable of learning the discriminative information of the clusters and outperformed the state of the art methods. We also evaluated the performance of the proposed clustering method on DEEZER music streams data where the ground truth partitions are unknown. We validated the clustering results by a depth looking of the cluster members properties. The qualitative analysis is also done and interesting results obtained that assesses the pertinence of the proposed method.

If you want to make God laugh, tell him about your plans.

— Woody Allen

5 Conclusions and Future Works

5.1 Conclusion

The first part of the thesis presents a sparse representation framework for time series where the inputs and the dictionary atoms may have different lengths and including varying delays. The initial study was to investigate the performance of the existing sparse coding and dictionary learning methods on time series data. The experiments showed that the popular existing methods are unable to produce the same good reconstruction or classification results on temporal data, as for the static data, in the presence of delay. The main reason is that the objective function of such methods defined in the Euclidean space which is sensitive to the time warped.

Based on the above observation, a time warped invariant sparse coding and a dictionary learning problem is formalized. For the sparse coding of time series, a variant of the orthogonal matching pursuit (TWI-OMP), based on a new COSTW similarity measure, is proposed. For two time series, COSTW finds the alignment between the two that maximize their cosine. A dynamic programming strategy is developed to find the COSTW similarity and the related alignment. In the next step, for updating the dictionary atoms. we proposed two solutions. One is inspired by the kSVD algorithm where each atom is updated sequentially by applying an SVD on the projected residuals vectors and the other one is a gradient descend minimization strategy.

A time series classification framework is investigated based on the purposed dictionary learning strategy. The classification problem is formulated similar to the ones of SRC [Wright 09] and Meta-face learning [Yang 10] where class-specific sub dictionaries are learned and then used to encode the arriving test samples. The experiments showed that the method is outperformed the standard methods where the inputs are zero-padded and the delay aspect is ignored. It is also shown that with a very small dictionary size and at a low-level of sparsity, it can provide reasonable reconstruction.

In the second part of the thesis, an extension of the proposed dictionary learning algorithm for time series clustering is formalized. The related objective function minimizes the intra-cluster and maximizes the inter-cluster variations by learning the sub cluster dictionaries. The clustering results are compared to the other state-of-the-art methods that we discussed in Chapter 2 and the proposed method offers the best performance under a more rigid condition. While subspace spectral clustering methods require the whole training set for dictionary, the proposed method only uses a small size dictionary per cluster.

Let us back to the big picture, we were looking for an unsupervised feature learning method for time series representation. In this thesis, various dictionary learning methods are investigated and an efficient learning strategies for time series data are suggested. In the following section we express the possible research directions that we will focus on in the future works.

5.2 Future works

Unsupervised feature learning is an active research domain in machine learning community in the past years. Artificial Neural Network (ANN), dictionary learning and recently deep learning have been widely studied and become popular due to their impressive results in image classification, speech recognition and natural language processing among other emerging applications.

In this thesis, we focus on the dictionary learning methods as an unsupervised feature learning technique for time series representation under time warp. We also discussed the limitations of the standard algorithms for time series data. Although, prior investigations have been done for learning dictionaries for time series such as shift invariant dictionary learning, the proposed methods shown promising and

competitive results. In the future studies, first we intend to generalize the proposed method for handling multivariate time series. The multivariate data are vectorized in the experiments which is not the ideal case. Similar to [Soheily-Khah 16], a weighting vector may be learned for each atom to involve the most pertinent regions for a given task (e.g. reconstruction). Secondly, we target to model the sub-dictionaries relationships as coherence and discrimination. Above all, we are looking for more challenging datasets and we are searching for different speed up techniques and better implementations of the proposed methods. The next objective is to explore another unsupervised learning model, particularly deep learning for time series analysis.

Appendices

A

Rotation for n-dimensional vectors

In Euclidean space to perform a rotation, a rotation matrix is needed. For example R_θ is a $2D$ counter-clockwise rotation matrix around point $(0, 0)$ with angle θ.

$$R_\theta = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

To perform the rotation using the rotation matrix R_θ on a column vector $\boldsymbol{x} \in \mathbb{R}^2$, we need a matrix multiplication as $R_\theta \boldsymbol{x}$. In three dimensional space, rotations are more complex since there are many possible rotations. Rotations can perform about one of the axis of a coordinate system using right-hand or left-hand rule whereas for high dimensional vector we need to rotate about an $n-2$ dimensional subspace.

Denote by $\boldsymbol{x}$ and $\boldsymbol{y}$ two vectors in $\mathbb{R}^n$, we want to find the rotation matrix that aligns vector $\boldsymbol{x}$ to vector $\boldsymbol{y}$. One way to do this is to find two orthonormal vectors in the plane generated by the two vectors using Gramm-Schmidt procedure, and then extend this 2-dimensional basis to an orthonormal basis of $\mathbb{R}^n$. Then with respect to this basis, consider the rotation by angle θ in the plan generated by the first two vectors, and the identity on the space generated by the rest of the orthonormal basis. Let

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{x}/||\boldsymbol{x}|| \\ &\boldsymbol{v} = \boldsymbol{y} - (\boldsymbol{u}'\boldsymbol{y})\boldsymbol{u},\ \boldsymbol{v} = \boldsymbol{v}/||\boldsymbol{v}|| \end{aligned}$$

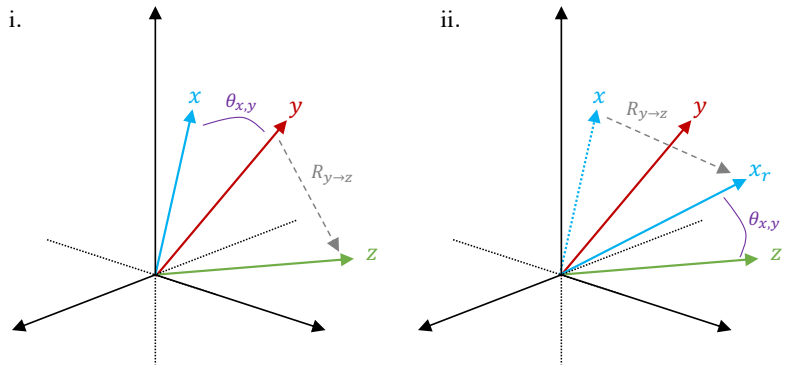

Figure A.1: A valid rotation to preserve the angle for three dimensional vectors.

where $\boldsymbol{u}$ is unit vector into the direction of $\boldsymbol{x}$ and $\boldsymbol{v}$ is unit vector into the direction of $\boldsymbol{y}$. Then $\boldsymbol{u}\boldsymbol{u}^T + \boldsymbol{v}\boldsymbol{v}^T$ is a projection onto the space generated by $\boldsymbol{x}$ and $\boldsymbol{y}$ and $I - \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{v}\boldsymbol{v}^T$ is the projection onto the $n-2$ dimensional complemented subspace. Thus, the whole rotation becomes

$$R_{x\to y} = I - \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{v}\boldsymbol{v}^T + [\boldsymbol{u}\ \boldsymbol{v}]R_\theta[\boldsymbol{u}\ \boldsymbol{v}]^T$$

where R_θ is the same as before and $cos(\theta) = \boldsymbol{x}^T\boldsymbol{y}/(||\boldsymbol{x}||.||\boldsymbol{y}||)$.

Now, assume we have three vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ and the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ ($\theta_{x,y}$) is known. To rotate $\boldsymbol{x}$ such that the angle between $\boldsymbol{x}_r$ (the rotated vector) and $\boldsymbol{z}$ becomes $\theta_{x,y}$, we need to first find the rotation matrix from $\boldsymbol{y}$ to $\boldsymbol{z}$ (denote as $R_{y\to z}$ in Figure A.1) and then apply the same rotation matrix on $\boldsymbol{x}$ to obtain $\boldsymbol{x}_r$. Figure A.1 illustrates these steps for a three dimensional vectorial space.

Zhelezov [Zhelezov 17] has also proposed a method to generate an n-dimensional rotation matrix, which rotates a given n-dimensional vector $\boldsymbol{x}$ to the direction of another vector $\boldsymbol{y}$ with the same dimension. Their algorithm generates two rotation matrices M_x and M_y, which rotates two given vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to the direction of an arbitrarily axis $\boldsymbol{x}_1$. The whole rotation matrix M is then obtained as the multiplication of matrix M_x and the inverse of matrix M_y. Throughout our experiments, we find the first method more robust and accurate.

B

Proof of Solution 3.19

We present here the proof for the proposed solution 3.19. Let us recall the dictionary learning optimization problem as:

$$\begin{aligned} \mathcal{J}(D) = \quad & \min_{D} \sum_{i=1}^{N}\sum_{t=1}^{q_i}(x_{it} - \sum_{j=1}^{K}\alpha_{ji}\sum_{(t,t')\in\boldsymbol{\pi}^*_{ij}} d_{jt'})^2 \\ \textit{subject to} \quad & \forall j \ \ \|\boldsymbol{d}_j\|_2 = 1 \end{aligned} \tag{B.1}$$

where $\boldsymbol{x}_i$, $i = 1, .., N$, is an input time series $\boldsymbol{x}_i = (x_{i1}, ..., x_{it}, ..., x_{iq_i})^T \in \mathbb{R}^{q_i}$ and $\boldsymbol{d}_j$, $j = 1, ..., K$, is the dictionary time series atom $\boldsymbol{d}_j = (d_{j1}, ..., d_{jt'}, ..., d_{jp_j})^T \in \mathbb{R}^{p_j}$. We assume the sparse coefficients $A = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_N]$ with $\boldsymbol{\alpha}_i = (\alpha_{1i}, ..., \alpha_{Ki})^T$ and the optimal alignments $\boldsymbol{\pi}^*_{ij}$, $i = 1, ..., N; j = 1, ..., K$ are known (learned by TWI-OMP).

To find the updating rule for the atom $\boldsymbol{d}_k$, we take the partial derivative of the above function with respect to the time stamp t' of $\boldsymbol{d}_k$. For that, the reconstruction sum is decomposed and the associated terms to $\boldsymbol{d}_k$ and $d_{kt'}$ are extracted:

$$\begin{aligned} \mathcal{J}(D) \ &= \\ &= \min_{D} \sum_{i=1}^{N}\sum_{t=1}^{q_i}(x_{it} - \alpha_{ki}\sum_{(t,t'')\in\boldsymbol{\pi}^*_{ik}} d_{kt''} - \sum_{j\neq k}^{K}\alpha_{ji}\sum_{(t,t'')\in\boldsymbol{\pi}^*_{ij}} d_{jt''})^2 \qquad \text{(B.2)} \\ &= \min_{D} \sum_{i=1}^{N}\sum_{t=1}^{q_i}(x_{it} - \alpha_{ki}d_{kt'} - \alpha_{ki}\sum_{\substack{(t,t'')\in\boldsymbol{\pi}^*_{ik} \\ (t''\neq t')}} d_{kt''} - \sum_{j\neq k}^{K}\alpha_{ji}\sum_{(t,t'')\in\boldsymbol{\pi}^*_{ij}} d_{jt''})^2 \qquad \text{(B.3)} \end{aligned}$$

By taking the partial derivative of Eq. B.3 with respect to $d_{kt'}$, we get:

$$\frac{\partial \mathcal{J}}{\partial d_{kt'}} = \sum_{i=1}^{N}\sum_{t=1}^{q_i} -2\alpha_{ki}(x_{it} - \alpha_{ki}d_{kt'} - \alpha_{ki}\sum_{\substack{(t,t'')\in\boldsymbol{\pi}_{ik}^{*}\\(t''\neq t')}} d_{kt''} - \sum_{j\neq k}\alpha_{ji}\sum_{(t,t'')\in\boldsymbol{\pi}_{ij}^{*}} d_{jt''}) \tag{B.4}$$

Note that since $\alpha_{ki} = 0, \forall i \notin \boldsymbol{\omega}_k$ with $\boldsymbol{\omega}_k$ be the set of sample indices involving atom $\boldsymbol{d}_k$, we can utilize $\sum_{i=1}^{|\boldsymbol{\omega}_k|}$ instead of $\sum_{i=1}^{N}$.□

The mind is everything. What you think you become.

— Buddha

Bibliography

[A. Frank 10] A. Asuncion A. Frank. *UCI machine learning repository.* http://archive.ics.uci.edu/ml/, 2010. [Online access].

[Agrawal 93] Rakesh Agrawal, Christos Faloutsos & Arun Swami. *Efficient similarity search in sequence databases.* In International Conference on Foundations of Data Organization and Algorithms, pages 69–84. Springer, 1993.

[Aharon 06] Michal Aharon, Michael Elad & Alfred Bruckstein. *k-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation.* Signal Processing, IEEE Transactions on, vol. 54, no. 11, pages 4311–4322, 2006.

[Aharon 08] Michal Aharon & Michael Elad. *Sparse and redundant modeling of image content using an image-signature-dictionary.* SIAM Journal on Imaging Sciences, vol. 1, no. 3, pages 228–247, 2008.

[Arfken 99] George B Arfken & Hans J Weber. Mathematical methods for physicists. AAPT, 1999.

[Barthélemy 12] Quentin Barthélemy, Anthony Larue, Aurélien Mayoue, David Mercier & Jérôme I Mars. *Shift & 2D rotation invariant sparse coding for multivariate signals.* Signal Processing, IEEE Transactions on, vol. 60, no. 4, pages 1597–1611, 2012.

[Bengio 09] Samy Bengio, Fernando Pereira, Yoram Singer & Dennis Strelow. *Group sparse coding.* In Advances in neural information processing systems, pages 82–89, 2009.

[Box 15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel & Greta M Ljung. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[Bradley 00] Paul S Bradley & Olvi L Mangasarian. *K-plane clustering.* Journal of Global Optimization, vol. 16, no. 1, pages 23–32, 2000.

[Bristow 13] Hilton Bristow, Anders Eriksson & Simon Lucey. *Fast convolutional sparse coding.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 391–398, 2013.

[Bro 08] Rasmus Bro & Tamara G Kolda. *Resolving the sign ambiguity in the singular value decomposition.* Journal of Chemometrics, vol. 22, no. 2, pages 135–140, 2008.

[Bryt 08] Ori Bryt & Michael Elad. *Compression of facial images using the K-SVD algorithm.* Journal of Visual Communication and Image Representation, vol. 19, no. 4, pages 270–282, 2008.

[Candes 00] Emmanuel J Candes & David L Donoho. *Curvelets: A surprisingly effective nonadaptive representation for objects with edges.* Rapport technique, Stanford Univ Ca Dept of Statistics, 2000.

[Chan 99] Kin-Pong Chan & Ada Wai-Chee Fu. *Efficient time series matching by wavelets.* In Data Engineering, 1999. Proceedings., 15th International Conference on, pages 126–133. IEEE, 1999.

[Chen 01] Scott Shaobing Chen, David L Donoho & Michael A Saunders. *Atomic decomposition by basis pursuit.* SIAM review, vol. 43, no. 1, pages 129–159, 2001.

[Chen 12] Mingyu Chen, Ghassan AlRegib & Biing-Hwang Juang. *6dmg: A new 6d motion gesture database.* In Proceedings of the 3rd Multimedia Systems Conference, pages 83–88. ACM, 2012.

[Chen 15] Zhihua Chen, Wangmeng Zuo, Qinghua Hu & Liang Lin. *Kernel sparse representation for time series classification.* Information Sciences, vol. 292, pages 15–26, 2015.

[Chu 95] Chia-Shang James Chu. *Time series segmentation: A sliding window approach.* Information Sciences, vol. 85, no. 1-3, pages 147–173, 1995.

[Chu 09] Selina Chu, Shrikanth Narayanan & C-C Jay Kuo. *Environmental sound recognition with time–frequency audio features.* IEEE Transactions on Audio, Speech, and Language Processing, vol. 17, no. 6, pages 1142–1158, 2009.

[Cuturi 11] Marco Cuturi. *Fast global alignment kernels.* In Proceedings of the 28th international conference on machine learning (ICML-11), pages 929–936, 2011.

[Do 17] Cao-Tri Do, Ahlame Douzal-Chouakria, Sylvain Marié, Michèle Rombaut & Saeed Varasteh. *Multi-modal and multi-scale temporal metric learning for a robust time series nearest neighbors classification.* Information Sciences, vol. 418, pages 272–285, 2017.

[Elad 06] Michael Elad & Michal Aharon. *Image denoising via sparse and redundant representations over learned dictionaries.* IEEE Transactions on Image processing, vol. 15, no. 12, pages 3736–3745, 2006.

[Elhamifar 13] Ehsan Elhamifar & Rene Vidal. *Sparse subspace clustering: Algorithm, theory, and applications.* IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 11, pages 2765–2781, 2013.

[Engan 99] Kjersti Engan, Sven Ole Aase & J Hakon Husoy. *Method of optimal directions for frame design.* In Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, volume 5, pages 2443–2446. IEEE, 1999.

[Fancourt 98] Craig L Fancourt & José Carlos Principe. *Competitive principal component analysis for locally stationary time series.* IEEE Transactions on Signal Processing, vol. 46, no. 11, pages 3068–3081, 1998.

[Fu 06] Tak-chung Fu, Fu-lai Chung & Chak-man Ng. *Financial Time Series Segmentation based on Specialized Binary Tree Representation.* DMIN, vol. 2006, pages 26–29, 2006.

[Gao 10] Shenghua Gao, Ivor Wai-Hung Tsang & Liang-Tien Chia. *Kernel sparse representation for image classification and face recognition.* In European Conference on Computer Vision, pages 1–14. Springer, 2010.

[Goldberger 00] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng & H Eugene Stanley. *Physiobank, physiotoolkit, and physionet.* Circulation, vol. 101, no. 23, pages e215–e220, 2000.

[Grosse 07] Roger Grosse, Rajat Raina, Helen Kwong & Andrew Y Ng. *Shift-invariant sparse coding for audio classification.* In Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, pages 149–158. AUAI Press, 2007.

[Guha 12] Tanaya Guha & Rabab K Ward. *Learning sparse representations for human action recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 8, pages 1576–1588, 2012.

[Guralnik 99] Valery Guralnik & Jaideep Srivastava. *Event detection from time series data.* In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 33–42. ACM, 1999.

[Hamilton 94] James Douglas Hamilton. Time series analysis, volume 2. Princeton university press Princeton, 1994.

[Huang 06] Ke Huang & Selin Aviyente. *Sparse representation for signal classification.* In Advances in neural information processing systems, pages 609–616, 2006.

[Huang 12] Po-Sen Huang, Jianchao Yang, Mark Hasegawa-Johnson, Feng Liang & Thomas S Huang. *Pooling Robust Shift-Invariant Sparse Representations of Acoustic Signals.* In INTERSPEECH, pages 2518–2521, 2012.

[Jeni 14] László A Jeni, András Lőrincz, Zoltán Szabó, Jeffrey F Cohn & Takeo Kanade. *Spatio-temporal event classification using time-series kernel based structured sparsity.* In Computer Vision–ECCV 2014, pages 135–150. Springer, 2014.

[Jiang 11] Zhuolin Jiang, Zhe Lin & Larry S Davis. *Learning a discriminative dictionary for sparse coding via label consistent K-SVD.* In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1697–1704. IEEE, 2011.

[Jost 06] Philippe Jost, Pierre Vandergheynst, Sylvain Lesage & Rémi Gribonval. *MoTIF: An efficient algorithm for learning translation invariant dictionaries.* In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 5, pages V–V. IEEE, 2006.

[Juang 91] Biing Hwang Juang & Laurence R Rabiner. *Hidden Markov models for speech recognition.* Technometrics, vol. 33, no. 3, pages 251–272, 1991.

[Kavukcuoglu 10] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu & Yann L Cun. *Learning convolutional feature hierarchies for visual recognition.* In Advances in neural information processing systems, pages 1090–1098, 2010.

[Keogh 01] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani & Sharad Mehrotra. *Dimensionality reduction for fast similarity search in large time series databases.* Knowledge and information Systems, vol. 3, no. 3, pages 263–286, 2001.

[Keogh 06] Eammon Keogh. *The UCR time series data mining archive.* http://www. cs. ucr. edu/˜eamonn/, 2006. [Online access].

[Lewicki 99] Michael S Lewicki & Terrence J Sejnowski. *Coding time-varying signals using sparse, shift-invariant representations.* Advances in neural information processing systems, pages 730–736, 1999.

[Li 17] Jun Li, Yu Kong & Yun Fu. *Sparse subspace clustering by learning approximation l0 codes.* In Proceedings of the AAAI Conference on Artificial Intelligence, pages 2189–2195, 2017.

[Lin 03] Jessica Lin, Eamonn Keogh, Stefano Lonardi & Bill Chiu. *A symbolic representation of time series, with implications for streaming algorithms.* In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pages 2–11. ACM, 2003.

[Lütkepohl 05] Helmut Lütkepohl. New introduction to multiple time series analysis. Springer Science & Business Media, 2005.

[Mailhé 08] Boris Mailhé, Sylvain Lesage, Rémi Gribonval, Frédéric Bimbot & Pierre Vandergheynst. *Shift-invariant dictionary learning for sparse representations: extending K-SVD*. In Signal Processing Conference, 2008 16th European, pages 1–5. IEEE, 2008.

[Mailhé 12] Boris Mailhé & Mark Plumbley. *Dictionary learning with large step gradient descent for sparse representations*. Latent Variable Analysis and Signal Separation, pages 231–238, 2012.

[Mairal 08] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro & Andrew Zisserman. *Discriminative learned dictionaries for local image analysis*. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[Mairal 09] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman & Francis R Bach. *Supervised dictionary learning*. In Advances in neural information processing systems, pages 1033–1040, 2009.

[Majumdar 09] Angshul Majumdar & Rabab K Ward. *Fast group sparse classification*. Canadian Journal of Electrical and Computer Engineering, vol. 34, no. 4, pages 136–144, 2009.

[Mallat 93] Stéphane G Mallat & Zhifeng Zhang. *Matching pursuits with time-frequency dictionaries*. Signal Processing, IEEE Transactions on, vol. 41, no. 12, pages 3397–3415, 1993.

[Nemenyi 62] Peter Nemenyi. *Distribution-free multiple comparisons*. Biometrics, vol. 18, no. 2, page 263, 1962.

[Ng 02] Andrew Y Ng, Michael I Jordan & Yair Weiss. *On spectral clustering: Analysis and an algorithm*. In Advances in neural information processing systems, pages 849–856, 2002.

[Olshausen 96] Bruno A Olshausen & David J Field. *Emergence of simple-cell receptive field properties by learning a sparse code for natural images*. Nature, vol. 381, no. 6583, page 607, 1996.

[Ophir 11] Boaz Ophir, Michael Lustig & Michael Elad. *Multi-scale dictionary learning using wavelets*. IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 5, pages 1014–1024, 2011.

[Pati 93] Yagyensh Chandra Pati, Ramin Rezaiifar & P. Sambamurthy Krishnaprasad. *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*. Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on, pages 40–44, 1993.

[Pham 08] Duc-Son Pham & Svetha Venkatesh. *Joint learning and dictionary construction for pattern recognition*. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[Poularakis 13] Stergios Poularakis, Grigorios Tsagkatakis, Panagiotis Tsakalides & Ioannis Katsavounidis. *Sparse representations for hand gesture recognition*. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 3746–3750. IEEE, 2013.

[Rabiner 86] Lawrence R Rabiner & Biing-Hwang Juang. *An introduction to hidden Markov models*. ieee assp magazine, vol. 3, no. 1, pages 4–16, 1986.

[Ramírez 09] Ignacio Ramírez, Federico Lecumberry & Guillermo Sapiro. *Sparse modeling with universal priors and learned incoherent dictionaries*. In University of Minnesota. Institute for Mathematics and Its Applications, 2009.

[Ramirez 10] Ignacio Ramirez, Pablo Sprechmann & Guillermo Sapiro. *Classification and clustering via dictionary learning with structured incoherence and shared features*. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 3501–3508. IEEE, 2010.

[Rand 71] William M Rand. *Objective criteria for the evaluation of clustering methods*. Journal of the American Statistical association, vol. 66, no. 336, pages 846–850, 1971.

[Rubinstein 10] Ron Rubinstein, Alfred M Bruckstein & Michael Elad. *Dictionaries for sparse representation modeling*. Proceedings of the IEEE, vol. 98, no. 6, pages 1045–1057, 2010.

[Sakoe 78] Hiroaki Sakoe & Seibi Chiba. *Dynamic programming algorithm optimization for spoken word recognition*. IEEE transactions on acoustics, speech, and signal processing, vol. 26, no. 1, pages 43–49, 1978.

[Smith 05] Evan Smith & Michael S Lewicki. *Efficient coding of time-relative structure using spikes*. Neural Computation, vol. 17, no. 1, pages 19–45, 2005.

[Soheily-Khah 16] Saeid Soheily-Khah, Ahlame Douzal-Chouakria & Eric Gaussier. *Generalized k-means-based clustering for temporal data under weighted and kernel time warp*. Pattern Recognition Letters, vol. 75, pages 63–69, 2016.

[Stiefmeier 07] Thomas Stiefmeier, Daniel Roggen & Gerhard Tröster. *Gestures are strings: efficient online gesture spotting and classification using string matching*. In Proceedings of the ICST 2nd international conference on Body area networks, page 16. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[Tibshirani 96] Robert Tibshirani. *Regression shrinkage and selection via the lasso*. Journal of the Royal Statistical Society. Series B (Methodological), pages 267–288, 1996.

[Tropp 04] Joel A Tropp. *Greed is good: Algorithmic results for sparse approximation*. Information Theory, IEEE Transactions on, vol. 50, no. 10, pages 2231–2242, 2004.

[Tseng 00] Paul Tseng. *Nearest q-flat to m points*. Journal of Optimization Theory and Applications, vol. 105, no. 1, pages 249–252, 2000.

[Van Nguyen 12] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi & Rama Chellappa. *Kernel dictionary learning*. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2021–2024. IEEE, 2012.

[Van Nguyen 13] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi & Rama Chellappa. *Design of non-linear kernel dictionaries for object recognition*. IEEE Transactions on Image Processing, vol. 22, no. 12, pages 5123–5135, 2013.

[Wang 10] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang & Yihong Gong. *Locality-constrained linear coding for image classification*. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 3360–3367. IEEE, 2010.

[Wang 14] Donghui Wang & Shu Kong. *A classification-oriented dictionary learning model: Explicitly learning the particularity and commonality across categories*. Pattern Recognition, vol. 47, no. 2, pages 885–898, 2014.

[Wei 13a] Chia-Po Wei, Yu-Wei Chao, Yi-Ren Yeh & Yu-Chiang Frank Wang. *Locality-sensitive dictionary learning for sparse representation based classification*. Pattern Recognition, vol. 46, no. 5, pages 1277–1287, 2013.

[Wei 13b] Chia-Po Wei, Yu-Wei Chao, Yi-Ren Yeh & Yu-Chiang Frank Wang. *Locality-sensitive dictionary learning for sparse representation based classification*. Pattern Recognition, vol. 46, no. 5, pages 1277–1287, 2013.

[Wright 09] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry & Yi Ma. *Robust face recognition via sparse representation*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 31, no. 2, pages 210–227, 2009.

[Yang 10] Meng Yang, Lei Zhang, Jian Yang & Dejing Zhang. *Metaface learning for sparse representation based face recognition*. In Image Processing (ICIP), 2010 17th IEEE International Conference on, pages 1601–1604. IEEE, 2010.

[Yang 11] Meng Yang, Lei Zhang, Xiangchu Feng & David Zhang. *Fisher discrimination dictionary learning for sparse representation*. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 543–550. IEEE, 2011.

[Yazdi 18a] Saeed Varasteh Yazdi & Ahlame Douzal-Chouakria. *Time warp invariant kSVD: Sparse coding and dictionary learning for time series under time warp*. Pattern Recognition Letters, vol. 112, pages 1–8, 2018.

[Yazdi 18b] Saeed Varasteh Yazdi, Ahlame Douzal-Chouakria, Patrick Gallinari & Manuel Moussallam. *Time warp invariant dictionary learning for time series clustering: application to music data stream analysis*. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 356–372. Springer, 2018.

[Yin 12] Jun Yin, Zhonghua Liu, Zhong Jin & Wankou Yang. *Kernel sparse representation based classification*. Neurocomputing, vol. 77, no. 1, pages 120–128, 2012.

[You 16] Chong You, Daniel Robinson & René Vidal. *Scalable sparse subspace clustering by orthogonal matching pursuit*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3918–3927, 2016.

[Yuan 06] Ming Yuan & Yi Lin. *Model selection and estimation in regression with grouped variables*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 68, no. 1, pages 49–67, 2006.

[Yuan 18] Jidong Yuan, Ahlame Douzal-Chouakria, Saeed Varasteh Yazdi & Zhihai Wang. *A large margin time series nearest neighbour classification under locally weighted time warps*. Knowledge and Information Systems, pages 1–19, 2018.

[Zhang 10] Qiang Zhang & Baoxin Li. *Discriminative K-SVD for dictionary learning in face recognition*. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 2691–2698. IEEE, 2010.

[Zhang 12] Li Zhang, Wei-Da Zhou, Pei-Chann Chang, Jing Liu, Zhe Yan, Ting Wang & Fan-Zhang Li. *Kernel sparse representation-based classifier*. IEEE Transactions on Signal Processing, vol. 60, no. 4, pages 1684–1695, 2012.

[Zhelezov 17] Ognyan Ivanov Zhelezov. *N-dimensional Rotation Matrix Generation Algorithm*. American Journal of Computational and Applied Mathematics, vol. 7, no. 2, pages 51–57, 2017.

[Zhou 13] Yin Zhou, Kai Liu, Rafael E Carrillo, Kenneth E Barner & Fouad Kiamilev. *Kernel-based sparse representation for gesture*

recognition. Pattern Recognition, vol. 46, no. 12, pages 3208–3222, 2013.