Interpretable time series kernel analytics by pre-image estimation

Thi Phuong Thao Tran

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Thi Phuong Thao TRAN

Thèse dirigée par **Ahlame DOUZAL**
et codirigée par **Paul HONEINE**, Professeur d'université, Université de Rouen
et **Saeed VARASTEH YAZDI**, UGA
préparée au sein du **Laboratoire Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Interprétabilité de l'analyse de séries temporelles à noyaux par l'estimation de la pré-image

Interpretable time series kernel analytics by pre-image estimation

Thèse soutenue publiquement le **18 septembre 2020**,
devant le jury composé de :

Madame AHLAME DOUZAL
MAITRE DE CONFERENCES HDR, UNIVERSITE GRENOBLE ALPES, Directrice de thèse
Monsieur MOHAMED NADIF
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE PARIS, Rapporteur
Monsieur CHRISTOPHE MARSALA
PROFESSEUR DES UNIVERSITES, SORBONNE UNIVERSITE - PARIS, Rapporteur
Monsieur PATRICK GALLINARI
PROFESSEUR DES UNIVERSITES, SORBONNE UNIVERSITE - PARIS, Examinateur
Madame SIHEM AMER-YAHIA
DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES, Présidente
Monsieur PAUL HONEINE
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE ROUEN-NORMANDIE, Co-directeur de thèse

Abstract

Kernel methods are known to be effective to analyse complex objects by implicitly embedding them into some feature space. To interpret and analyse the obtained results, it is often required to restore in the input space the results obtained in the feature space by using pre-image estimation methods. This work proposes a pre-image estimation method for time series kernel analytics that consists of two steps. In the first step, a time warp function, driven by distance constraints in the feature space, is defined to embed time series in a metric space where analytics can be performed conveniently. In the second step, the time series pre-image estimation is cast as learning a linear (or a nonlinear) transformation that ensures a local isometry between the time series embedding space and the feature space. The proposed method is compared to state of the art through three major tasks that require pre-image estimation: 1) time series averaging, 2) time series reconstruction and denoising, and 3) time series representation learning. The extensive experiments conducted son 33 publicly-available datasets show the benefits of the pre-image estimation for time series kernel analytics.

Keywords: Time series, temporal kernel, pre-image estimation, representation learning, dimensionality reduction, dictionary learning.

Résumé

Les méthodes à noyaux sont connues pour être efficaces pour l'analyse d'objets complexes en les plongeant implicitement dans un espace de caractéristiques (feature-space). Pour interpréter et analyser les résultats obtenus, il est souvent nécessaire de restaurer dans l'espace d'entrée les résultats obtenus dans l'espace des caractéristiques à l'aide de méthodes d'estimation de la pré-image. Ce travail propose une méthode d'estimation de la pré-image pour rendre interprétable les méthodes d'analyse de séries temporelles à base de noyaux. Dans la première étape, une fonction de déformation temporelle, supervisée par des contraintes de distances, est définie pour plonger les séries dans un espace métrique où des analyses pratiques peuvent être menées. Dans la deuxième étape, l'estimation de la pré-image des séries temporelles est obtenue par l'apprentissage d'une transformation linéaire (ou non linéaire) assurant une isométrie locale entre le nouvel espace métrique des séries et l'espace des caractéristiques. La méthode proposée est comparée aux méthodes de l'état de l'art au travers de trois tâches principales requérant l'estimation de la pré-image: 1) le centrage des séries temporelles, 2) la reconstruction et le débruitage des séries temporelles et 3) l'apprentissage de représentations pour des séries temporelles.

Mots-clés : Séries temporelles , noyau temporel , estimation de pré-images, apprentissage de représentation, réduction de dimension, apprentissage de dictionnaire.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Prof. Ahlame Douzal, for her support, patience. She has spent a lot of her time and her energy to guide and teach me. It is great luck to be her Ph.D. student.
I am thankful to my co-supervisor, Prof. Paul Honeine, for following each stage of my Ph.D. I also thank my co-supervisor, Dr. Saeed Varasteh Yazdi, to accompany and help me during my Ph.D.
I would like to thank all members my jury for their interest in my work. I also thank Prof. Mohamed Nadif and Prof. Christophe Marsala for reviewing my thesis and providing helpful comments and suggestions. I thank Prof. Patrick Gallinari and Dr. Sihem Amer-Yahia for participating in my Ph.D. defense committee. I thank them all for their valuable time and useful feedback.
I thank the AMA team for giving me the professional and intimate environment to work. I also thank my colleagues for encouraging me to overcome the difficulties of my Ph.D.
Finally, I would like to thank my parents for always standing beside me. They have been my inspiration and motivation for continuing my education.

Contents

List of Figures

List of Tables

Abbreviations

DTAK	Dynamic Time Alignment Kernel
DTW	Dynamic Time Warping
KGA	Kernel Global Alignment
KDTW	Kernel Dynamic Time Warping
k-SVD	k-Singular Value Decomposition
PCA	Principal Component Analysis
OMP	Orthogonal Matching Pursuit
SVMs	Support Vector Machines
p.d	positive definite
TsPrima	Time series Pre-image
MDS	Multidimensional Scaling
RKHS	Reproducing Kernel Hilbert space
6DMG	6 Dimensional Motion Gesture

1

Introduction

Over the past two decades, machine learning has become one of the fastest-growing areas involved with computer science and statistics. Machine learning plays a vital role in the revolution of science and technology in multiple fields as biomedical informatics, computer vision and natural language processing. Kernel methods [53] are among the major machine learning approaches, known to be effective in dealing with nonlinear problems and complex data as time series, sequences and graphs. The main idea of kernel methods is to map the data in the input space $\mathcal{X}$ to a feature space $\mathcal{H}$ via a nonlinear mapping Φ, where the data can be processed conveniently by linear approaches, as illustrated in Figure 1.1. *Kernel trick*, defines the main concept behind kernel machines, that consists to process all data computations by implicitly using the inner products $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle = \kappa(\boldsymbol{x}, \boldsymbol{x}')$, where κ is a valid kernel. Since then, many nonlinear algorithms have been developed for

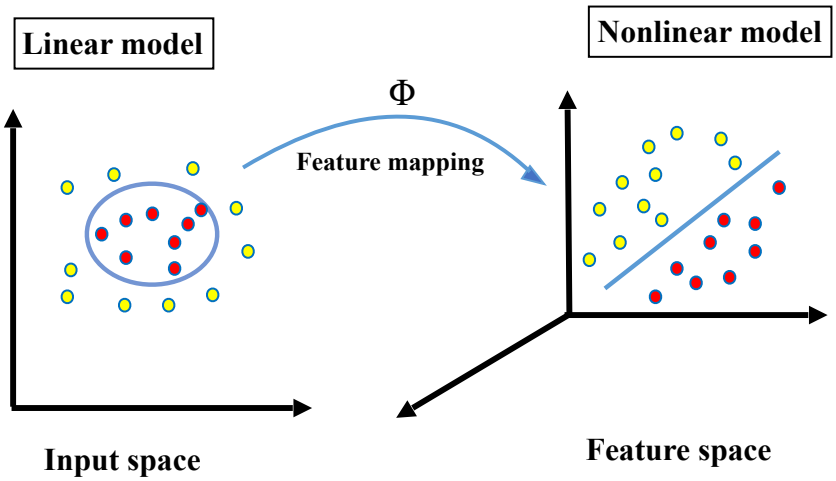


Figure 1.1: Illustration of kernel methods

several machine learning tasks as for classification with kernel Fisher discriminant analysis [43] and support vector machines [57], for dimensionality reduction with kernel principal component analysis [46], for sparse coding with kernel k-SVD [5] and for clustering with kernel k-means [32]. The price that one should pay for the efficient kernel machinery is that the solutions are only obtained as expansions in terms of the mapped input samples into the feature spaces. However, in many situations, for analysis and interpretation purposes, there is a need of the reverse mapping of the obtained results from the feature space back to the input space, called pre-image estimation problem. For instance, given some noisy samples, kernel PCA first applies linear PCA on the mapped samples in the feature space, then perform denoising by projecting them onto the subspace defined by the leading eigenvectors. The projections have then to be mapped back to the input space to recover the denoised samples. The pre-image estimation is of a high interest in many other kernel tasks to obtain, for instance, the reverse mapping of the centroids of a kernel clustering or the pre-image of the atoms and the sparse representations of a kernel dictionary learning, among others. In view of the importance of the pre-image estimation issue and of its benefits in machine learning, several major propositions have been developed.

The first study proposed by Schölkopf [40] gives the exact pre-image solution based

on the inner product between the pre-image and a set of vectors of an orthonormal basis into the input space. This method underlines two strict conditions: the existence of the pre-image solution and the inversion of the function f_κ, where $\kappa(\boldsymbol{x}, \boldsymbol{y}) = f_\kappa(\langle \boldsymbol{x}, \boldsymbol{y} \rangle)$, that are unsatisfied in general. Thus, several pre-image estimation solutions have been proposed. First, in Mika [46], the problem is formalised as a nonlinear optimization problem, which for the particular case of the Gaussian kernel allows to estimate the reverse mapping based on a fixed-point iterative process. To avoid numerical instabilities of the latter approach, in Kwok [55], the relationship between the distances in the feature and the input spaces is established for isotropic kernels and then is used to approximate pre-images by multidimensional scaling. In Bakir [51], the pre-image estimation problem is cast as a regression problem between the input and the mapped data and the learned regression model is used to predict pre-images. Honeine and Richard proposed in [48] an approach in which the main idea is to estimate, from the mapped data, a coordinate system that ensures an isometry with the input space. This approach has the advantage to provide a closed-form solution, to be independent of the kernel nature and to involve only linear algebra.

All the proposed methods for pre-image estimation are either based on optimization schemas, such as gradient descent or fixed-point iterative solution or based on ideas borrowed from dimension reduction methods. In particular, these methods are developed for Euclidean input spaces, as derivations are straightforward owing to linear algebra (see [48] for a survey on the resolution of the pre-image problems in machine learning). A major challenge arises when dealing with non-Euclidean input spaces, which are often used to represent structured data as sequences, time series or graphs. In particular, for time series data, thanks to temporal kernels ([36],[28]), kernel machinery has been increasingly investigated with success for time series kernel analytics ([63], [60], [17], [39, 11]), the pre-image problem for temporal data remains unaddressed. In addition, time series data, that include temporal dependency and time delays, are naturally lying in a non-Euclidean input space, preventing the application of the traditional approaches for pre-image estimation.

This thesis aims to fill this gap by proposing a pre-image estimation approach for time series kernel analytics. The main idea of the proposed method consists of two steps. In the first step, a time warp function, driven by distance constraints in the feature space, is defined to embed time series into a metric space where analytics can be performed conveniently. In the second step, the time series pre-image estimation is cast as learning a linear (or a nonlinear) transformation that ensures a local isometry between the time series embedding space and the feature space. The relevance of the proposed time series pre-image estimation is studied through three major tasks :

- time series averaging,
- time series reconstruction and denoising under kernel PCA,
- time series sparse representation under kernel dictionary learning.

The benefits of the proposed method are assessed through extensive experiments conducted on 33 publicly-available time series datasets, including univariate and multivariate time series that may include varying delays and be of the different lengths. The main contributions of this thesis are:

1. We propose a time warp function, driven by distance constraints in the feature space, that embeds time series into an Euclidean space.
2. We cast the time series pre-image estimation approach as learning linear or nonlinear transformations in the feature space.
3. We propose a tractable solution that ensures a local isometry between the temporal embedded space and the feature space.
4. We conduct wide experiments to compare the proposed approach to the major alternative pre-image estimation methods under three crucial tasks: 1) time series averaging, 2) time series reconstruction and denoising, and 3) dictionary learning and sparse representations for time series.

In the thesis, the remainder is summarised as follows. Chapter 2 gives a brief introduction to kernel PCA, kernel k-SVD and kernel regression, three important

machine learning methods that crucially require pre-image estimation. Then we present the major related works for pre-image estimation in Chapter 3. In Chapter 4, we formalise the pre-image estimation problem for time series and develop the proposed method as well as the corresponding solution. In Chapter 5, we detail the experiments conducted and discuss the obtained results. Finally, in Chapter 6, we conclude this thesis and point out some perspectives for this work.

Notations

$\mathcal{X}$	an input space
X	a set of input samples/ a matrix
N	number of input samples
p	number of eigenvectors
n	number of neighbors
$\boldsymbol{x}$	a vector/ a time series
$\| . \|_2$	Euclidean distance
$\| . \|_{\mathcal{F}}$	Frobenius distance
X' or X^T	transpose of matrix X
X^{-1}	inverse of matrix X
$\mathcal{H}$	a Hilbert space/ a feature space
$\langle ., . \rangle$	an inner product
$\kappa(., .)$	a kernel function
$\Phi(.)$	a feature mapping
K	a Gram matrix
$K_{ii'}$	element in i^{th} row and j^{th} column of matrix K
H	a centering matrix
λ	a regularity term
Λ	a diagonal matrix
I_N	identity matrix of size N
$\mathbf{1}_N$	column vector of size N with entries 1
$\mathbb{1}_N$	square matrix of size N with entries $1/N$

$diag(\lambda_1, ..., \lambda_N)$	diagonal matrix of $(\lambda_1, \ldots, \lambda_N)$
$\boldsymbol{\alpha}_j$	an eigenvector of Gram matrix
λ_j	an eigenvalues of Gram matrix
$\boldsymbol{\pi}$	an alignment between two time series
$t_{\boldsymbol{x}}$	length of time series $\boldsymbol{x}$
$\Phi(X)$	row vector of $\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N)$
$P(\Phi(\boldsymbol{x}))$	projection of $\Phi(x)$ on subspace of the largest eigenvectors
$\boldsymbol{\varphi}$	a feature vector
$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_N)^T$	coefficient vector of $\boldsymbol{\varphi}$ with respect to $\{\Phi(\boldsymbol{x}_i)\}_{i=1}^N$
$\boldsymbol{k}_{\boldsymbol{x}} = (\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x}))$	row vector of similarities between $\boldsymbol{x}_i$ and $\boldsymbol{x}$
$\boldsymbol{x}$	pre-image of $\boldsymbol{\varphi}$
$\boldsymbol{x}^*$	an approximation of $\boldsymbol{x}$/ an estimation of the pre-image of $\boldsymbol{\varphi}$
D	a dictionary
$\boldsymbol{d}_j$	an atom of dictionary D
L	number of atoms in dictionary D
A	a matrix of sparse coding
$\boldsymbol{a}_i$	a sparse coding
f_r	a temporal embedding function

2

Importance of pre-image in kernel machinery

This chapter gives a brief introduction to kernel PCA [42], kernel k-SVD [5] and kernel regression [62], three methods largely used in machine learning tasks where the pre-image estimation is highly required. Let $\mathcal{X}$ be a compact set in $\mathbb{R}^d$. The positive definite (reproducing) kernel $\kappa(.,.)$ is a function on $\mathcal{X}^2 \to \mathbb{R}$, which for all sets of input samples $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathcal{X}$ gives positive matrices K with entries $K_{ii'} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$. By [44], there exists a unique reproducing kernel Hilbert space $\mathcal{H}$ (RKHS) that is associated with kernel κ via feature mapping $\Phi : \mathcal{X} \to \mathcal{H}$. That means kernel κ can be evaluated as inner product in $\mathcal{H}$:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'}) \rangle$$

Kernel methods [53] rely on embedding samples $\boldsymbol{x} \in \mathcal{X}$ with $\Phi(\boldsymbol{x})$ into a feature space $\mathcal{H}$, of arbitrary large and possibly infinite dimension. The map function Φ needs not to be explicitly defined, since computations conducted in $\mathcal{H}$ can be carried out by a kernel function that measures the inner product in that space, namely $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'}) \rangle$ for all $\boldsymbol{x}_i, \boldsymbol{x}_{i'}$.

Given a set of input samples $\{\boldsymbol{x}_i\}_{i=1}^N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, let K be the Gram matrix related to the kernel κ. With some abuse of notation, let $\Phi(X)$ be the row vector of entries $\Phi(\boldsymbol{x}_1), ..., \Phi(\boldsymbol{x}_N)$. Note that kernel κ is supposed to be positive definite, then K is a positive matrix whose all of eigenvalues are positive.

In the following, the two first Sections describe kernel PCA and kernel k-SVD, as nonlinear extensions of the well-known PCA and k-SVD. While both methods estimate a linear combination for optimal reconstruction of the input samples, the former forces the orthogonality of the atoms that leads to an orthonormal basis basis, and the latter forces the sparsity while relaxing the orthogonality condition.

2.1 Kernel PCA

Principal Component Analysis (PCA) is a powerful technique for extracting structure from possibly high-dimensional datasets. PCA is an orthogonal transformation of the coordinate system in which we describe data. The new coordinate system is obtained by projection onto the so-called principal components of the data. A small number of principal components can be sufficient to account for most of the structure in the data.

Given a set of samples $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^d$ which, for clarity reasons, are assumed centered, namely $\sum_{i=1}^N \boldsymbol{x}_i = \boldsymbol{0}$, PCA finds the principal components by diagonalizing the covariance matrix:

$$C = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T. \tag{2.1}$$

C is positive definite that can thus be diagonalized (Problem (2.1)) by solving the eigendecomposition:

$$\lambda_j \boldsymbol{u}_j = C\boldsymbol{u}_j. \tag{2.2}$$

with eigenvalues $\lambda_j \geq 0$ and nonzero eigenvectors $\boldsymbol{u}_j \in \mathbb{R}^d \setminus \{\mathbf{0}\}$. Substituting Eq. (2.1) into the expression (2.2)

$$\lambda_j \boldsymbol{u}_j = C\boldsymbol{u}_j = \frac{1}{N}\sum_{i=1}^{N} \langle \boldsymbol{x}_i, \boldsymbol{u}_j \rangle \boldsymbol{x}_i.$$

we see that all solutions $\boldsymbol{u}_j$ with $\lambda_j \neq 0$ lie in the span of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, hence for the solutions Eq. (2.2) is equivalent to

$$\lambda_j \langle \boldsymbol{x}_i, \boldsymbol{u}_j \rangle = \langle \boldsymbol{x}_i, C\boldsymbol{u}_j \rangle, \quad \forall i = 1, \ldots, N.$$

The ratio of eigenvalues λ_j is the ratio of explanatory importance of the principal components with respect to the variables. If a principal component has a low eigenvalue, then it is contributing little to the explanation of variances in the variables and may be ignored as redundant with more important principal components. Hence, for dimensionality reduction or compression data, one can choose the number p of principal components such that

$$p = \underset{p \in \mathbb{N}^*}{\arg\min} \frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{N} \lambda_j} \geq 0.95, \tag{2.3}$$

where 0.95 is the proportion of information extracted from the input data.

Standard PCA only allows linear dimensionality reduction. However, if the data has more complicated structures which cannot be well represented in a linear space, standard PCA will not be very helpful. Fortunately, based on kernel trick, kernel PCA extends standard PCA to find principal components that are nonlinearly related to the input variables (illustrated in Figure 2.1). For that, the principal components are rather determined in the feature space. Similarly, for the sake of clarity, we assume for now that we are dealing with centered mapped data, namely $\sum_{i=1}^{N} \Phi(\boldsymbol{x}_i) = \mathbf{0}$. The covariance matrix in the feature space takes

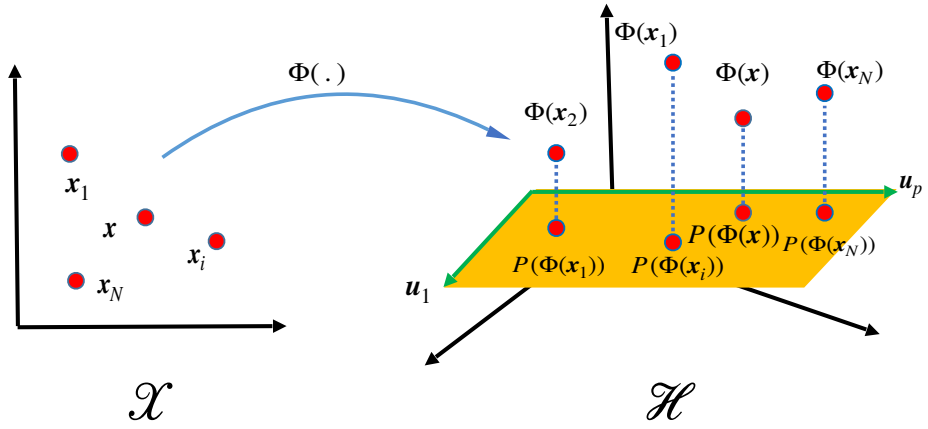


Figure 2.1: Illustration of kernel PCA

then the form of

$$C = \frac{1}{N}\sum_{i=1}^{N}\Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_i)^T. \tag{2.4}$$

Similarly to standard PCA, the objective comes to find the eigenvalues $\lambda_j \geq 0$ and eigenvectors $\boldsymbol{u}_j \in \mathcal{H} \setminus \{\boldsymbol{0}\}$ that satisfies

$$\lambda_j \boldsymbol{u}_j = C\boldsymbol{u}_j. \tag{2.5}$$

As each $\boldsymbol{u}_j$ lie in the span of $\Phi(\boldsymbol{x}_1), ..., \Phi(\boldsymbol{x}_N)$, there exist coefficients $\alpha_{1j}, \ldots, \alpha_{Nj}$ such that

$$\boldsymbol{u}_j = \sum_{i=1}^{N}\alpha_{ij}\Phi(\boldsymbol{x}_i), \tag{2.6}$$

and for each $\Phi(\boldsymbol{x}_{i'})$

$$\lambda_j < \boldsymbol{u}_j, \Phi(\boldsymbol{x}_{i'}) >=< C\boldsymbol{u}_j, \Phi(\boldsymbol{x}_{i'}) > . \tag{2.7}$$

Combining Eq. (2.6) and Eq. (2.7), we get

$$\begin{aligned}
\lambda_j\langle\sum_{i=1}^{N}\alpha_{ij}\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'})\rangle &= \langle C\sum_{i=1}^{N}\alpha_{ij}\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'})\rangle, \\
\lambda_j \sum_{i=1}^{N}\alpha_{ij}\langle\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'})\rangle &= \Big\langle \frac{1}{N}\sum_{i=1}^{N}\Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_i)^T\sum_{i=1}^{N}\alpha_{ij}\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'})\Big\rangle,
\end{aligned} \tag{2.8}$$

$$\lambda_j \sum_{i=1}^{N} \alpha_{ij} \langle \Phi(\boldsymbol{x}_j), \Phi(\boldsymbol{x}_{i'}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \alpha_{ij} \Big\langle \sum_{j=1}^{N} \Phi(\boldsymbol{x}_j) \Phi(\boldsymbol{x}_j)^T \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'}) \Big\rangle.$$

In terms of the Gram matrix $K = (K_{ii'})_{ii'}$ related to the kernel κ:

$$K_{ii'} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) =< \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{i'}) >,$$

and $\boldsymbol{\alpha}_j = [\alpha_{1j}, \ldots, \alpha_{Nj}]$, we have:

$$\lambda_j K \boldsymbol{\alpha}_j = \frac{1}{N} K^2 \boldsymbol{\alpha}_j. \tag{2.9}$$

The problem (2.6) remains to find the solution of the eigendecomposition problem:

$$\lambda_j \boldsymbol{\alpha}_j \;\; = \;\; \frac{1}{N} K \boldsymbol{\alpha}_j. \tag{2.10}$$

Let $\lambda_1 \geq \ldots \geq \lambda_p$ ($N\lambda_j$ in Eq. (2.10)) be the p non-zero eigenvalues of $\frac{1}{N}K$ and $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p$ their corresponding eigenvectors. The principal components in the feature space are then given by computing the projections $P_j(\Phi(\boldsymbol{x}))$ of the sample $\boldsymbol{x}$ onto the eigenvector $\boldsymbol{u}_j = \Phi(X)\,\boldsymbol{\alpha}_j$:

$$P_j(\Phi(\boldsymbol{x})) =< \boldsymbol{u}_j, \Phi(\boldsymbol{x}) >= \sum_{i=1}^{N} \alpha_{ij} < \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) >= \boldsymbol{k}_{\boldsymbol{x}} \boldsymbol{\alpha}_j, \tag{2.11}$$

with $\boldsymbol{k}_{\boldsymbol{x}} = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x})]$. By denoting $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p]$, the description $P(\Phi(\boldsymbol{x}))$ of $\Phi(\boldsymbol{x})$ into the sub-space of the p first principal components is then

$$P(\Phi(\boldsymbol{x})) \;\; = \;\; (\boldsymbol{k}_{\boldsymbol{x}} \boldsymbol{\alpha})^T. \tag{2.12}$$

Two considerations should be taken in the above results. First, the eigenvectors $\boldsymbol{u}_j$ should be normalised by:

$$\begin{aligned} 1 &= \langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle = \langle \Phi(X)\,\boldsymbol{\alpha}_j, \Phi(X)\,\boldsymbol{\alpha}_j \rangle \\ &= \boldsymbol{\alpha}_j^T \Phi(X)^T \Phi(X) \boldsymbol{\alpha}_j = \boldsymbol{\alpha}_j^T K \boldsymbol{\alpha}_j \\ &= \boldsymbol{\alpha}_j^T \lambda_j \boldsymbol{\alpha}_j = \lambda_j \langle \boldsymbol{\alpha}_j, \boldsymbol{\alpha}_j \rangle. \end{aligned}$$

Secondly, as $\Phi(X)$ should be centered by considering $\widetilde{\Phi}(X) = \Phi(X) - \frac{1}{N}\Phi(X)\mathbf{1}_N\mathbf{1}_N^T$ with $\mathbf{1}_N = (1, \ldots, 1)^T \in \mathbb{R}^N$ the unit vector. The Gram matrix K in Eq. (2.10) and $\boldsymbol{k}_{\boldsymbol{x}}$ in Eq. (2.11) need to be substituted with their centered counterparts,

namely $\widetilde{K}$ and $\widetilde{\boldsymbol{k}}_{\boldsymbol{x}}$, as follows:

$$\begin{aligned}
\widetilde{K} &= \widetilde{\Phi}(X)^T\widetilde{\Phi}(X) \\
&= \Big(\Phi(X) - \frac{1}{N}\Phi(X)\mathbf{1}_N\mathbf{1}_N^T\Big)^T\Big(\Phi(X) - \frac{1}{N}\Phi(X)\mathbf{1}_N\mathbf{1}_N^T\Big) \\
&= \Phi(X)^T\Phi(X) - \frac{1}{N}\Phi(X)^T\Phi(X)\mathbf{1}_N\mathbf{1}_N^T - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T\Phi(X)^T\Phi(X) + \\
&\quad + \frac{1}{N^2}\mathbf{1}_N\mathbf{1}_N^T\Phi(X)^T\Phi(X)\mathbf{1}_N\mathbf{1}_N^T \\
&= K - \frac{1}{N}K\mathbf{1}_N\mathbf{1}_N^T - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^TK + \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^TK\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T \\
&= K - K\mathbb{1}_N - \mathbb{1}_NK + \mathbb{1}_NK\mathbb{1}_N, \qquad (2.13)
\end{aligned}$$

with $(\mathbb{1}_N)_{ij} = 1/N$ for all i, j, and I_N the identity matrix.
Similarly, $\widetilde{\boldsymbol{k}}_{\boldsymbol{x}}$ is defined by:

$$\begin{aligned}
\widetilde{\boldsymbol{k}}_{\boldsymbol{x}} &= \Big[\langle\widetilde{\Phi}(\boldsymbol{x}_1), \widetilde{\Phi}(\boldsymbol{x})\rangle, ..., \langle\widetilde{\Phi}(\boldsymbol{x}_N), \widetilde{\Phi}(\boldsymbol{x})\rangle\Big] \\
&= \Big[\langle\Phi(\boldsymbol{x}_1) - \frac{1}{N}\Phi(X)\mathbf{1}_N, \Phi(\boldsymbol{x}) - \frac{1}{N}\Phi(X)\mathbf{1}_N\rangle, \\
&\qquad ..., \langle\Phi(\boldsymbol{x}_N) - \frac{1}{N}\Phi(X)\mathbf{1}_N, \Phi(\boldsymbol{x}) - \frac{1}{N}\Phi(X)\mathbf{1}_N\rangle\Big] \\
&= \Big[\langle\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x})\rangle - \frac{1}{N}\Phi(\boldsymbol{x})\Phi(X)\mathbf{1}_N - \Phi(\boldsymbol{x}_i)\frac{1}{N}\Phi(X)\mathbf{1}_N + \\
&\qquad + \frac{1}{N^2}\mathbf{1}_N^T\Phi(X)^T\Phi(X)\mathbf{1}_N\rangle\Big]_{i=1}^N \\
&= \Big[\kappa(\boldsymbol{x}_i, \boldsymbol{x}) - \frac{1}{N}\boldsymbol{k}_{\boldsymbol{x}}\mathbf{1}_N - \frac{1}{N}\boldsymbol{k}_{\boldsymbol{x}_i}\mathbf{1}_N + \frac{1}{N^2}\mathbf{1}_N^TK\mathbf{1}_N\Big]_{i=1}^N \\
&= \big(\boldsymbol{k}_{\boldsymbol{x}} - \frac{1}{N}\boldsymbol{k}_{\boldsymbol{x}}\mathbf{1}_N\mathbf{1}_N^T\big) - \big(\frac{1}{N}\mathbf{1}_N^TK - \frac{1}{N^2}\mathbf{1}_N^TK\mathbf{1}_N\mathbf{1}_N^T\big) \\
&= \boldsymbol{k}_{\boldsymbol{x}}(I_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T) - \frac{1}{N}\mathbf{1}_N^TK(I_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T) \\
&= (\boldsymbol{k}_{\boldsymbol{x}} - \frac{1}{N}\,\mathbf{1}_N^T\,K)\,(I_N - \mathbb{1}_N).
\end{aligned}$$

For centered data, the p principal components $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$ are then:

$$\boldsymbol{u}_j = \sum_{i=1}^N \alpha_{ij}\widetilde{\Phi}(\boldsymbol{x}_i),$$

where $\widetilde{\Phi}(\boldsymbol{x}_j) = \Phi(\boldsymbol{x}_j) - \bar{\Phi}$ and $\bar{\Phi} = \frac{1}{N}\sum_{i=1}^N \Phi(\boldsymbol{x}_i)$.
The projection $P(\widetilde{\Phi}(\boldsymbol{x}))$ of $\widetilde{\Phi}(\boldsymbol{x})$ onto the subspace of the p eigenvectors is defined

by:

$$\begin{aligned} P(\tilde{\Phi}(\boldsymbol{x})) &= \sum_{i=1}^{p} \langle \tilde{\Phi}(\boldsymbol{x}), \boldsymbol{u}_i \rangle \boldsymbol{u}_i = \sum_{i=1}^{p} \langle \tilde{\Phi}(\boldsymbol{x}), \tilde{\Phi}(X)\boldsymbol{\alpha}_i \rangle \tilde{\Phi}(X)\boldsymbol{\alpha}_i \\ &= \sum_{i=1}^{p} \langle \tilde{\Phi}(\boldsymbol{x}), \tilde{\Phi}(X) \rangle \boldsymbol{\alpha}_i \tilde{\Phi}(X)\boldsymbol{\alpha}_i = \sum_{i=1}^{p} \tilde{\boldsymbol{k}}_{\boldsymbol{x}} \boldsymbol{\alpha}_i \tilde{\Phi}(X)\boldsymbol{\alpha}_i \\ &= \sum_{i=1}^{p} \tilde{\Phi}(X)\boldsymbol{\alpha}_i (\tilde{\boldsymbol{k}}_{\boldsymbol{x}} \boldsymbol{\alpha}_i)^T = \sum_{i=1}^{p} \tilde{\Phi}(X)\boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T \tilde{\boldsymbol{k}}_{\boldsymbol{x}}^T \\ &= \tilde{\Phi}(X)\boldsymbol{\alpha}\boldsymbol{\alpha}^T \tilde{\boldsymbol{k}}_{\boldsymbol{x}}^T. \end{aligned}$$

From that, we obtain the approximation of $\Phi(\boldsymbol{x})$ as

$$\begin{aligned} \Phi(\boldsymbol{x}) &= \tilde{\Phi}(\boldsymbol{x}) + \bar{\Phi} \\ &\approx P(\tilde{\Phi}(\boldsymbol{x})) + \bar{\Phi} \\ &= \tilde{\Phi}(X)\boldsymbol{\alpha}\boldsymbol{\alpha}^T \tilde{\boldsymbol{k}}_{\boldsymbol{x}}^T + \bar{\Phi} \\ &= \Phi(X)(I_N - \mathbb{1}_N)\boldsymbol{\alpha}\boldsymbol{\alpha}^T \tilde{\boldsymbol{k}}_{\boldsymbol{x}}^T + \frac{1}{N}\Phi(X)\mathbf{1}_N \\ &= \Phi(X)\big[(I_N - \mathbb{1}_N)\boldsymbol{\alpha}\boldsymbol{\alpha}^T \tilde{\boldsymbol{k}}_{\boldsymbol{x}}^T + \frac{1}{N}\mathbf{1}_N\big]. \end{aligned} \tag{2.14}$$

Note that $\Phi(\boldsymbol{x})$ is expressed as a linear combination of the mapped set $\Phi(X)$.

Kernel PCA first maps the data into the feature space via a nonlinear feature mapping Φ, then preforms linear PCA on the mapped data. This method is convenient to detect nonlinear structure in a given data [27] and highly used for data compression, reconstructions and denoising. However, the results obtained by kernel PCA live in high dimensional feature space $\mathcal{H}$. To make the compressed, denoised results expressed into the input space, a pre-image problem should be solved.

2.2 Kernel k-SVD

Sparse coding and dictionary learning become popular methods in machine learning and pattern recognition for a variety of tasks as feature extraction

([56],[35],[39]), reconstruction, denoising, compressed sensing ([14],[9]) and classification ([33],[64]). The aim of sparse coding methods is to represent input samples as a linear combination of few basis functions called atoms composing a given dictionary. Sparse coding is generally formalised as an optimisation problem that minimises the error of the reconstruction under l_0 or l_1 sparsity constraint. The l_0 constraint, that control the maximum number of involved atoms, leads to a nonconvex and NP-hard problem. This problem can however be solved efficiently by using the matching pursuit method ([18]) or its orthogonal variant ([66]). Relaxing the sparsity constraint from l_0 to l_1 norm yields a convex sparse coding problem, also known a a LASSO problem ([54]). In sparse coding, the used dictionary may be selected among pre-specified familly of basis functions as, among Fourier, Wavelets ([34]), Curvelets ([10]), Contourlets ([61]) and Gabor functions ([37]). Although these dictionaries allow fast transforms, their reconstribution potential is tightly related to the nature of the data. For instance, Wavelets show efficient reconstruction for natural images and textures ([13]), Curvelets for edges ([37]) and Gabor for sounds ([26]). The alternative to the above basis functions is to use a dictionary learning approach to learn, from the input data, a set of atoms to sparse represent the input samples. To solve that dictionary learning problem most approaches alternate between two steps: 1) keep the dictionary fixed and find the sparse representation using a sparse approximation algorithm, e.g., orthogonal matching pursuit (OMP), 2) keep the representation fixed and update the dictionary, either all the atoms at once by using for instance MOD (method of optimal directions) ([21]) or one atom at a time as in k-SVD([1]). In particular, k-SVD uses a singular value decomposition to learn jointly the dictionary as well as the sparse coefficient. k-SVD can be viewed as a generalisation of k-means algorithm that relaxes the assignment constrain to represent each input sample by a linear combination of few representative atoms (i.e., the centroids) instead by using only one centroid. In the following,we formalise the standard sparse coding and dictionary learning problem and present the standard efficient solution k-SVD.

Given an input sample $\boldsymbol{x} \in \mathbb{R}^d$ and a dictionary $D \in \mathbb{R}^{d \times L}$: $D = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_L]$ composed of L atoms $\boldsymbol{d}_j \in \mathbb{R}^d$, the objective of sparse coding problem is to represent

sparsely $\boldsymbol{x}$ by a linear combination of a few atoms of D. This problem is formalised as minimising the error of reconstruction of $\boldsymbol{x}$ under a sparsity constraint:

$$\min_{\boldsymbol{a}} \|\boldsymbol{x} - D\boldsymbol{a}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{a}\|_0 \leq \tau \tag{2.15}$$

where $\boldsymbol{a} = (a_1, \ldots, a_L)^T \in \mathbb{R}^L$ is the sparse code of $\boldsymbol{x}$ and D is in general a predefined (e.g., Fourier, Wavelet, Gabor basis functions) and overcomplete (i.e., $d << L$) dictionary. The l_0 sparsity constraint in Eq. (2.15) ensures to limit the maximum number of involved atoms to τ. Although the l_0-norm renders the problem formalised in Eq. (2.15) nonconvex and NP-hard, it can be efficiently solved via matching pursuit ([18]) or its orthogonal variant OMP ([66]). The main idea of OMP method is to select at each iteration the atoms $\boldsymbol{d}_j$ that is highly correlated to the input sample or to its residual part. The coefficient a_j, obtained by an orthogonal projection on the sub-space defined by the yet selected atoms, defines the contribution of $\boldsymbol{d}_j$ to reconstruct $\boldsymbol{x}$. The process is reiterated until the maximum number τ of atoms is reached. Algorithm 1 gives the main steps of the OMP method. It is worth noting that although initialising the dictionary with a given family of basis functions (e.g., Fourrier, Wavelet) hastens the process, the sparse coding result remain in general more precise when the dictionary are proposed in the literature, among them the k-SVD [14] method that we detail in the following.

Let $X \in \mathbb{R}^{d \times N}$: $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ be the matrix giving the description of N input samples, with $\boldsymbol{x}_i \in \mathbb{R}^d$. The dictionary learning problem, that generalises the sparse coding given in Eq. (2.15), can be formalised as learning both the sparse coding and the dictionary D to minimise the error of reconstruction of a set input samples:

$$\min_{A,D} \|X - DA\|_{\mathcal{F}}^2 \quad s.t. \quad \forall i \ \|\boldsymbol{a}_i\|_0 \leq \tau, \quad \forall j \ \|\boldsymbol{d}_j\|_2 = 1 \tag{2.16}$$

where $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{L \times N}$ gives the sparse coding $\boldsymbol{a}_i \in \mathbb{R}^L$ of samples $\boldsymbol{x}_i$ and $\boldsymbol{d}_j$ the jth atom of unit l_2-norm. The above optimisation problem is not convex in both A and D, that is resolved in general by using a block-coordinate-descent method. This method consists of alternating two steps: 1) keep the dictionary D fixed and learn the sparse codes A and 2) keep the sparse codes A fixed and

Algorithm 1 OMP

Input: $\boldsymbol{x} \in \mathbb{R}^d$, D, τ.
Output: $\boldsymbol{a}$.

1: Initialization: $\boldsymbol{r} = \boldsymbol{x}$, $\Omega = \{\emptyset\}$
2: **while** $|\Omega| \leq \tau$ **do**
3: Select the atom $\boldsymbol{d}_j$ $(j \notin \Omega)$:

$$\boldsymbol{d}_j = \arg\max \frac{\boldsymbol{r}^T \boldsymbol{d}_j}{\|\boldsymbol{r}\|_2 \|\boldsymbol{d}_j\|_2}$$

4: Update the set of selected atoms:

$$\Omega = \Omega \cup j$$

5: Update the coefficients:

$$\boldsymbol{a}_\Omega = (D_\Omega^T D_\Omega)^{-1} (D_\Omega^T \boldsymbol{x})$$

{where D_Ω is the subs-dictionary of the selected atoms and $\boldsymbol{a}_\Omega$ the related coefficients}
6: Estimate the residual:

$$\boldsymbol{r} = \boldsymbol{x} - D_\Omega \boldsymbol{a}_\Omega$$

7: **end while**
8: **Return** $\boldsymbol{a}$.

learn the dictionary D. The algorithm k-SVD uses a singular value decomposition (SVD) to learn jointly the dictionary as well as the sparse codes as follows.

Let us denote $\boldsymbol{a}_{j.} = (\boldsymbol{a}_{j1}, \ldots, \boldsymbol{a}_{jN})$ is the jth row of matrix A, it provides the contribution of atom $\boldsymbol{d}_j$ to reconstruct N input samples. To update $\boldsymbol{d}_k$, the objective function in Eq. (2.16) can be formulated as:

$$\begin{aligned} \min_{A,D} \|X - DA\|_{\mathcal{F}}^2 &= \min_{A,D} \|X - \sum_{j=1}^{L} \boldsymbol{d}_j \boldsymbol{a}_{j.}\|_{\mathcal{F}}^2 \\ &= \min_{A,D} \|(X - \sum_{j \neq k} \boldsymbol{d}_j \boldsymbol{a}_{j.}) - \boldsymbol{d}_k \boldsymbol{a}_{k.}\|_{\mathcal{F}}^2 \\ &= \min_{A,D} \|E_k - \boldsymbol{d}_k \boldsymbol{a}_{k.}\|_{\mathcal{F}}^2, \end{aligned} \tag{2.17}$$

where DA is expressed as the sum of L rank-1 matrices, each one giving the sparse representation of X involving one atom. The matrix $E_k \in \mathbb{R}^{d \times N}$ stands for the error of reconstruction for the N samples excluding the kth atom. An SVD rank-1 approximation on E_k can be used to find $\boldsymbol{d}_k$ and $\boldsymbol{a}_{k.}$. However, the new $\boldsymbol{a}_{k.}$ may not be sparse anymore. To preserve the sparsity of $\boldsymbol{a}_{k.}$, the residual matrix

Algorithm 2 k-SVD

Input: $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^{d \times N}$, D,τ.
Output: D, A.

repeat
  for $i = 1, \ldots, N$ **do**

$$\boldsymbol{a}_i = OMP(\boldsymbol{x}_i, D, \tau)$$

  end for
  for $k = 1, \ldots, L$ **do**
    Estimate $E_k = X - \sum_{j \neq k} \boldsymbol{d}_j \boldsymbol{a}_{j.}$

$$\Omega_k = \{i / a_{ki} \neq 0, i = 1, \ldots, N\}$$

    Define E_k^R as the restriction of E_k to Ω_k
    Apply an SVD on $E_k^R = U \Sigma V^T$
    Update $\boldsymbol{d}_k = \boldsymbol{u}_1$ and $\boldsymbol{a}_{k.}^R = \sigma_1 \boldsymbol{v}_1$
  end for
until convergence
Return D, A.

E_k is limited to only samples that involve atom $\boldsymbol{d}_k$. Denote ω_k as the set of index where $\boldsymbol{a}_{k.}$ is not zero and Ω_k as a matrix of size $N \times |\omega_k|$ with 1 on the $(\omega_k(i), i)$ entries and zeros elsewhere. $\boldsymbol{a}_{k.}^R$ and E_k^R are discarded the zeros columns of $\boldsymbol{a}_{k.}$ and E_k by multiplying to Ω_k. Subsequently, SVD is used to estimate the closest rank-1 matrix that approximate E_k^R annd the first column $\boldsymbol{u}_1$, 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 $\boldsymbol{a}_{k.}$.

k-SVD is an iterative algorithm (Algorithm 2) that alternates between sparse coding of the input samples based on the current dictionary and a process of updating the dictionary atoms to better fit the given data. The update of the dictionary columns is combined with an update of the sparse representations, thereby accelerating convergence. The k-SVD algorithm is flexible and can work with any pursuit method (e.g., basis pursuit, Focuss, or matching pursuit).

When dealing with complex data, kernel k-SVD may be required to learn in the feature space the dictionary and the sparse representations of the mapped samples as a nonlinear combination of the dictionary atoms [5]. This method improves the separating margin between dictionaries and allow better tolerance against different

types of degradation. Moreover, kernel k-SVD can provide better discrimination than the standard k-SVD and kernel PCA. Let us introduce a brief description of kernel k-SVD.

Let $D = [\boldsymbol{d}_1, ..., \boldsymbol{d}_L] \in \mathbb{R}^{d \times L}$ be the dictionary composed of L atoms $\boldsymbol{d}_j \in \mathbb{R}^d$. The embedded dictionary $\Phi(D) = \Phi(X)\mathcal{B}$ is defined as a linear representation of $\Phi(X)$, since the atoms lie in the subspace spanned by the $\Phi(X)$. The kernel dictionary learning problem takes the form

$$\begin{aligned} &\min_{\mathcal{B},\mathcal{A}} \|\Phi(X) - \Phi(X)\mathcal{B}\,\mathcal{A}\|_{\mathcal{F}}^2, \\ &\text{s.t. } \|\boldsymbol{a}_i\|_0 \leq \tau \ \forall i = 1, ..., N, \end{aligned} \tag{2.18}$$

where the matrix $\mathcal{B} = [\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_L] \in \mathbb{R}^{N \times L}$ gives the representation of the embedded atoms into the base $\Phi(X)$ and $\mathcal{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_N] \in \mathbb{R}^{L \times N}$ gives the sparse representations of $\Phi(X)$, with the sparsity level τ imposed by the above constraint.

The kernel k-SVD algorithm iteratively cycles between two stages. In the first stage, the dictionary is assumed fixed with $\mathcal{B}$ known and a kernel orthogonal matching pursuit (KOMP) technique [31] is deployed to estimate $\mathcal{A}$ as

$$\begin{aligned} &\min_{\boldsymbol{a}_i} \|\Phi(\boldsymbol{x}_i) - \Phi(X)\mathcal{B}\boldsymbol{a}_i\|_2^2, \\ &\text{s.t.} \|\boldsymbol{a}_i\|_0 \leq \tau, \quad \forall i = 1, \ldots, N. \end{aligned} \tag{2.19}$$

As in standard OMP, given a sample $\boldsymbol{x}$, we select the atoms that best reconstructs $\Phi(\boldsymbol{x})$ by using the iterative proceduce. We denote that:

$$\Phi(\boldsymbol{x}) = \Phi(X)\widehat{\boldsymbol{x}} + \boldsymbol{r},$$

where $\widehat{\boldsymbol{x}}$ and $\boldsymbol{r}$ are respectively the current estimation of $\boldsymbol{x}$ and the current residual based on the selected atoms. Let Ω be the set of indices of selected atoms. The residual $\boldsymbol{r}$ is projected on the remaining dictionary atoms:

$$\begin{aligned} \langle \boldsymbol{r}, \Phi(X)\boldsymbol{\beta}_i \rangle &= \langle \Phi(\boldsymbol{x}) - \Phi(X)\widehat{\boldsymbol{x}}, \Phi(X)\boldsymbol{\beta}_i \rangle, \\ &= (\boldsymbol{k}_{\boldsymbol{x}} - \widehat{\boldsymbol{x}}^T K)\boldsymbol{\beta}_i, \quad \forall i \notin \Omega. \end{aligned} \tag{2.20}$$

The method selects a new dictionary atom in the remaining set that gives largest projection coefficient in Eq. (2.20). This selection guarantees the biggest reduction

of approximation error:

$$\Omega = \Omega \cup \arg\max_i \langle \boldsymbol{r}, \Phi(X)\boldsymbol{\beta}_i \rangle$$

The sparse representation of $\Phi(\boldsymbol{x})$ on the selected dictionary atoms are obtained by using the least square solution:

$$\begin{aligned}
\boldsymbol{a}_{\boldsymbol{x}} &= \arg\min_{\boldsymbol{a}_{\boldsymbol{x}}} \|\Phi(\boldsymbol{x}) - \Phi(X)\boldsymbol{\beta}_\Omega \boldsymbol{a}_{\boldsymbol{x}}\|_2^2, \\
\boldsymbol{a}_{\boldsymbol{x}} &= \Big((\Phi(X)\boldsymbol{\beta}_\Omega)^T(\Phi(X)\boldsymbol{\beta}_\Omega)\Big)^{-1}(\Phi(X)\boldsymbol{\beta}_\Omega)^T\Phi(\boldsymbol{x}), \\
&= (\boldsymbol{\beta}_\Omega^T K \boldsymbol{\beta}_\Omega)^{-1}(\boldsymbol{k}_{\boldsymbol{x}}\boldsymbol{\beta}_\Omega)^T.
\end{aligned}$$

The estimation $\widehat{\boldsymbol{x}}$ is updated by

$$\widehat{\boldsymbol{x}} = \boldsymbol{\beta}_\Omega \boldsymbol{a}_{\boldsymbol{x}}$$

The procedure is reiterate until the selection of τ atoms. Once the sparse codes $\mathcal{A}$ of the N samples estimated, the second stage of the kernel k-SVD is performed to update the dictionary $\mathcal{B}$ and sparse coding $\mathcal{A}$. For that, the reconstruction error is defined as

$$\begin{aligned}
\min_{\mathcal{B},\mathcal{A}} \|\Phi(X) - \Phi(X)\mathcal{B}\mathcal{A}\|_{\mathcal{F}}^2 &= \min_{\mathcal{B},\mathcal{A}} \|\Phi(X) - \Phi(X)\sum_{j=1}^{L} \boldsymbol{\beta}_j \boldsymbol{a}_{j.}\|_{\mathcal{F}}^2, \\
&= \min_{\mathcal{B},\mathcal{A}} \|\Phi(X)(I_N - \sum_{j\neq k} \boldsymbol{\beta}_j \boldsymbol{a}_{j.}) - \Phi(X)\boldsymbol{\beta}_k \boldsymbol{a}_{k.}\|_{\mathcal{F}}^2, \\
&= \min_{\mathcal{B},\mathcal{A}} \|\Phi(X)E_k - \Phi(X)\boldsymbol{\beta}_k \boldsymbol{a}_{k.}\|_{\mathcal{F}}^2, \quad \forall k = 1, \ldots, L,
\end{aligned} \tag{2.21}$$

with $\boldsymbol{a}_{j.} \in \mathbb{R}^N$ referencing the j-th row of $\mathcal{A}$ and $E_k = I_N - \sum_{j\neq k} \boldsymbol{\beta}_j \boldsymbol{a}_{j.}$ the error of reconstruction matrix when removing the k-th atom. An eigendecomposition is then preformed to get

$$(\Phi(X)E_k^R)^T(\Phi(X)E_k^R) = (E_k^R)^T K E_k^R = V\Sigma V^T, \tag{2.22}$$

where E_k^R is the error of reconstruction restricted to the samples that have involved the k-th atom. The dictionary $\boldsymbol{\beta}_k$ and sparse coding $\boldsymbol{a}_{k.}^R$ are updated by using the

first eigenvector $\boldsymbol{v}_1$ with

$$\boldsymbol{a}_{k.}^R = \sigma_1 \boldsymbol{v}_1^T \text{ and } \boldsymbol{\beta}_k = \sigma_1^{-1} E_k^R \boldsymbol{v}_1. \tag{2.23}$$

Similarly to kernel PCA, the obtained dictionary $\Phi(X)\mathcal{B}$ of kernel k-SVD, that is a linear combination of $\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N)$, lives in the high dimensional feature space. To restitute the learned sparse representation as well as the learned dictionary into the input space, a pre-image estimation problem should be solved.

2.3 Kernel regression

The linear regression problem has been widely used in statistics and machine learning. The non linear kernel regression was proposed later [58], it performs a linear regression in the kernel feature space, which represents a nonlinear regression in the input space. To avoid a large number of parameter estimations and computational difficulties, a dual version of ridge regression is proposed in [62], closely related to Vapnik's kernel method [59].

Given two sets $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ and $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ with $\boldsymbol{x}_i \in \mathbb{R}^p$ and the repesctive $\boldsymbol{y}_i \in \mathbb{R}^q$. The linear regression problem aims to learn a linear transformation $L \in \mathbb{R}^{q \times p}$ such that:

$$L = \arg\min_L \|Y - LX\|_{\mathcal{F}}^2,$$

then predict $\boldsymbol{y}$ with respect to a new sample $\boldsymbol{x}$, based on the obtained transformation:

$$\boldsymbol{y} = L\boldsymbol{x}.$$

The ridge regression adds a regularisation term to the linear regression problem as:

$$\|Y - LX\|_{\mathcal{F}}^2 + \lambda \|L\|_{\mathcal{F}}^2. \tag{2.24}$$

with the regularisation parameter $\lambda \geq 0$. The constrained formulation of Eq. (2.24) is then:

$$\|R\|_{\mathcal{F}}^2 + \lambda \|L\|_{\mathcal{F}}^2 \quad \text{s.t.} \quad R = Y - LX \tag{2.25}$$

Using Lagrange multipliers $A \in \mathbb{R}^{N \times q}$, we can replace the above constrained optimisation problem by

$$\lambda \|L\|_{\mathcal{F}}^2 + \|R\|_{\mathcal{F}}^2 + trace\big(A(Y - LX - R)\big) \tag{2.26}$$

In the Kuhn Tucker theorem, there exist values of Lagrange A for which, the problem (2.25) is equivalent to the problem (2.26). First, to minimise (2.26), we make the differential in L equal to 0:

$$\begin{aligned} 2\lambda L - A^T X^T &= 0 \\ L &= \frac{1}{2\lambda} A^T X^T \end{aligned} \tag{2.27}$$

Substituting Eq. (2.27) into Eq. (2.26), we have

$$\begin{aligned} &\lambda \|\frac{1}{2\lambda} A^T X^T\|_{\mathcal{F}}^2 + \|R\|_{\mathcal{F}}^2 + trace\big(A(Y - \frac{1}{2\lambda} A^T X^T X - R)\big) \\ &= \frac{1}{4\lambda} trace\big(A^T X^T (A^T X^T)^T\big) + \|R\|_{\mathcal{F}}^2 - \frac{1}{2\lambda} trace(AA^T X^T X) + trace(AY) - trace(AR) \\ &= \frac{1}{4\lambda} trace\big(A^T X^T X A\big) + \|R\|_{\mathcal{F}}^2 - \frac{1}{2\lambda} trace(AA^T X^T X) + trace(AY) - trace(AR) \\ &= -\frac{1}{4\lambda} trace(AA^T X^T X) + \|R\|_{\mathcal{F}}^2 + trace(AY) - trace(AR) \end{aligned} \tag{2.28}$$

Set the derivative of (2.28) with respect to R equal to 0, we obtain

$$\begin{aligned} 2R - A^T &= 0 \\ R &= \frac{1}{2} A^T \end{aligned} \tag{2.29}$$

Substituting Eq. (2.29) into (2.28), it leads:

$$\begin{aligned} &-\frac{1}{4\lambda} trace(AA^T X^T X) + \|\frac{1}{2} A^T\|_{\mathcal{F}}^2 + trace(AY) - trace(A\frac{1}{2} A^T) \\ &= -\frac{1}{4\lambda} trace(AA^T X^T X) - \frac{1}{4} AA^T + trace(AY) \end{aligned} \tag{2.30}$$

Make the derivative of (2.30) with respect to A equal to 0, we have

$$\begin{aligned} -\frac{1}{2\lambda}X^TXA - \frac{1}{2}A + Y^T &= 0 \\ A &= 2\lambda(X^TX + \lambda I_N)^{-1}Y^T \end{aligned} \tag{2.31}$$

Given a new sample $\boldsymbol{x}$, from Eq. (2.27) the prediction $\boldsymbol{y}$ can be given by

$$\begin{aligned} \boldsymbol{y} = L\boldsymbol{x} &= \frac{1}{2\lambda}A^TX^T\boldsymbol{x} \\ &= \frac{1}{2\lambda}(2\lambda(X^TX + \lambda I_N)^{-1}Y^T)^TX^T\boldsymbol{x} \\ &= \lambda Y(X^TX + \lambda I_N)^{-1}X^T\boldsymbol{x}. \end{aligned} \tag{2.32}$$

To make nonlinear regression between X and Y, [62] embed the input data X into a high dimensional feature space $\mathcal{H}$ via the feature mapping Φ, then construct a linear regression in the feature space between $\Phi(X)$ and Y. This linear regression problem can then be formulated as learning a transformation L:

$$L = \arg\min_L \|Y - L\Phi(X)\|_{\mathcal{F}}^2$$

Similar to the dual linear regression in the input space, given a new sample $\boldsymbol{x}$, the prediction of $\boldsymbol{y}$ is then obtained by using Eq. (2.32) and substituting X^TX and $X^T\boldsymbol{x}$ by respectively $\Phi(X)^T\Phi(X)$ and $\Phi(X)^T\Phi(\boldsymbol{x})$ as:

$$\boldsymbol{y} = \lambda Y(\Phi(X)^T\Phi(X) + \lambda I_N)^{-1}\Phi(X)^T\Phi(\boldsymbol{x}). \tag{2.33}$$

Using the property $\langle\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')\rangle = \kappa(\boldsymbol{x}, \boldsymbol{x}')$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p$, we have:

$$\begin{aligned} \Phi(X)^T\Phi(X) &= [\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N)]^T[\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N)] \\ &= (\langle\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)\rangle)_{ij} \\ &= (\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j))_{ij}, \end{aligned}$$

and

$$\begin{aligned}\Phi(X)^T\Phi(\boldsymbol{x}) &= [\Phi(\boldsymbol{x}_1),\ldots,\Phi(\boldsymbol{x}_N)]^T\Phi(\boldsymbol{x}) \\ &= (\langle\Phi(\boldsymbol{x}_1),\Phi(\boldsymbol{x})\rangle,\ldots,\langle\Phi(\boldsymbol{x}_1),\Phi(\boldsymbol{x})\rangle)^T \\ &= (\kappa(\boldsymbol{x}_1,\boldsymbol{x}),\ldots,\kappa(\boldsymbol{x}_N,\boldsymbol{x}))^T.\end{aligned}$$

Hence, Eq. (2.33) is rewritten as:

$$\boldsymbol{y} = \lambda Y(K+\lambda I_N)^{-1}\boldsymbol{k}_{\boldsymbol{x}}^T. \tag{2.34}$$

with $K = (\kappa(\boldsymbol{x}_i,\boldsymbol{x}_{i'}))_{i,i'} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{k}_{\boldsymbol{x}}^T = (\kappa(\boldsymbol{x}_1,\boldsymbol{x}),\ldots,\kappa(\boldsymbol{x}_N,\boldsymbol{x}))^T \in \mathbb{R}^N$.

While kernel regression learns a nonlinear transformation between X and Y by embedding X into the feature space, kernel dependency [23] learns a nonlinear transformation between X and Y by embedding both sets into different feature spaces. In the first step, Y is embeded into the feature space $\mathcal{H}_{\hat{\kappa}}$ associated kernel $\hat{\kappa}$, where a kernel PCA is performed to obtain the representation of $\Phi_{\hat{\kappa}}(Y)$. The p principal components are denoted $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_p$ in $\mathcal{H}_{\hat{\kappa}}$. By kernel PCA, the coefficient of any feature vector $\Phi_{\hat{\kappa}}(\boldsymbol{\varphi})$ can be defined by

$$P(\Phi_{\hat{\kappa}}(\boldsymbol{\varphi})) = (\langle\Phi_{\hat{\kappa}}(\boldsymbol{\varphi}),\boldsymbol{u}_1\rangle,\ldots,\langle\Phi_{\hat{\kappa}}(\boldsymbol{\varphi}),\boldsymbol{u}_p\rangle)^T. \tag{2.35}$$

Similarly, X is embedded in the feature space $\mathcal{H}_\kappa$ related to the Gram matrix $K \in \mathbb{R}^{N\times N}$ with entries $K_{ii'} = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_{i'})$. Subsequently, the problem is to learn a transformation L between $P(\Phi_{\hat{\kappa}}(\boldsymbol{Y}))$ and $\Phi_\kappa(X)$ as

$$L = \arg\min_L \|P(\Phi_{\hat{\kappa}}(\boldsymbol{Y})) - L\Phi_\kappa(X)\|_{\mathcal{F}}^2$$

In the second step, given a new sample $\boldsymbol{x}$ and by using Eq. (2.34), the prediction of $P(\Phi_{\hat{\kappa}}(\boldsymbol{y}))$ is obtained as:

$$P(\Phi_{\hat{\kappa}}(\boldsymbol{y})) = \lambda P(\Phi_{\hat{\kappa}}(\boldsymbol{Y}))(K+\lambda I_N)^{-1}(\boldsymbol{k}_{\boldsymbol{x}})^T. \tag{2.36}$$

Let's denote $\boldsymbol{\gamma}_1 = \lambda P(\Phi_{\hat{\kappa}}(\boldsymbol{Y}))(K + \lambda I_N)^{-1}(\boldsymbol{k_x})^T$, from Eq. (2.35) and the orthogonal PCA systems obtained in Eq. (2.6), we have

$$\begin{aligned}(\langle \Phi_{\hat{\kappa}}(\boldsymbol{y}), \boldsymbol{u}_1 \rangle, \ldots, \langle \Phi_{\hat{\kappa}}(\boldsymbol{y}), \boldsymbol{u}_p \rangle)^T &= \boldsymbol{\gamma}_1 \\ \Phi_{\hat{\kappa}}(\boldsymbol{y}) &\approx [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p]\boldsymbol{\gamma}_1 \\ \Phi_{\hat{\kappa}}(\boldsymbol{y}) &\approx [\Phi_{\hat{\kappa}}(Y)\boldsymbol{\alpha}_1, \ldots, \Phi_{\hat{\kappa}}(Y)\boldsymbol{\alpha}_p]\boldsymbol{\gamma}_1 \\ \Phi_{\hat{\kappa}}(\boldsymbol{y}) &\approx \Phi_{\hat{\kappa}}(Y)\boldsymbol{\alpha}\boldsymbol{\gamma}_1 \end{aligned} \tag{2.37}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p]$ are eigenvectors of Gram matrix $\hat{K}$ with entries $\hat{K}_{ii'} = \hat{\kappa}(\boldsymbol{y}_i, \boldsymbol{y}_{i'})$. For the prediction of $\boldsymbol{y}$, the pre-image estimation of $\Phi_{\hat{\kappa}}(\boldsymbol{y})$, that is a combination of $\Phi(\boldsymbol{y}_1), \ldots, \Phi(\boldsymbol{y}_N)$ in Eq. (2.37), should be solved:

$$\boldsymbol{y} = \arg\min_{\boldsymbol{y}^*} \|\Phi_{\hat{\kappa}}(\boldsymbol{y}^*) - \Phi_{\hat{\kappa}}(\boldsymbol{y})\|^2_{\mathcal{F}}. \tag{2.38}$$

In this chapter, we introduce three well-known kernel methods that are kernel PCA, kernel SVD and kernel regression. These methods have been used commonly for the analysis of complex and unstructured data by embedding the data into a feature space via a kernel mapping. The main trick behind these methods is to learn nonlinear structures in the input space by learning linear models in the feature space. While such approaches are fruitful and have widely proven their efficiency, the results obtained are lying in the kernel feature space, limiting further interpretations and analysis. The pre-image problem in then crucial to complement the kernel approaches and allows for any result obtained in the feature space to be restored into the initial space.

3

Related works for pre-image estimation

In this chapter, we will report the state of the art of the pre-image estimation problem. The pre-image estimation problem is formalised and the major related works on pre-image estimation on static and unstructured data are presented. From the representer theorem [50], any feature vector $\boldsymbol{\varphi} \in \mathcal{H}$ obtained by some kernel method may be expressed in the following form:

$$\boldsymbol{\varphi} = \sum_{i=1}^{N} \gamma_i \, \Phi(\boldsymbol{x}_i), \tag{3.1}$$

that is as a linear combination of the mapped input samples $\{\Phi(\boldsymbol{x}_i)\}_{i=1}^{N}$, and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_N)^T$ is called the coefficient vector of $\boldsymbol{\varphi}$ with respect to the set $\{\Phi(\boldsymbol{x}_i)\}_{i=1}^{N}$. The pre-image problem aims to find out a sample $\boldsymbol{x} \in \mathcal{X}$ such that $\Phi(\boldsymbol{x}) = \boldsymbol{\varphi}$. Since the feature space $\mathcal{H}$ has a much higher dimension than the input space $\mathcal{X}$, the pre-image $\boldsymbol{x}$ may not exist. The pre-image problem is an

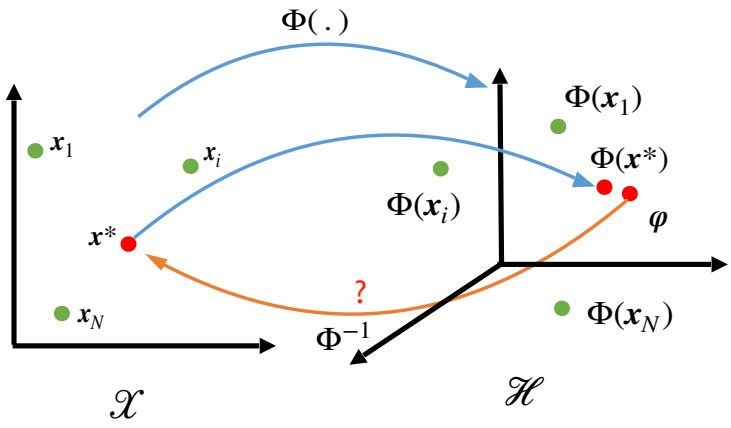


Figure 3.1: Illustration of the pre-image problem

ill-posed problem (illustrated in Figure 3.1), that is often addressed by providing an approximate solution, namely by estimating $\boldsymbol{x}^*$ such that $\Phi(\boldsymbol{x}^*) \approx \boldsymbol{\varphi}$. In this section, we will discuss the major proposed approaches for pre-image estimation problem. First, in [41] an exact solution for pre-image problem is proposed under two assumptions: the existence of the pre-image and the inversion of the kernel mapping. However, the exact pre-image of a feature vector may not exist and the invertibility of Φ is only hold for some kernels. Second, [46] cast the pre-image problem as a nonlinear optimisation problem, which for particular choices of kernels, can be solved by the fixed-point iteration method. However, this method suffers from local minimum and numerical instabilities. Latter, [55] determine a relationship between the distances in the feature space and the distances in the input space, then apply a multidimensional scaling technique (MDS) to obtain the pre-image estimation. Unlike the method proposed in [46], the [55] method is non-iterative, involves only linear algebra and does not suffer from numerical instabilities or the local minimum problem. However, the obtained pre-image estimation is reconstructed by using only local information. The method proposed by [48] learns a new coordinate system in $\mathcal{H}$ making an isometry between the feature space and the input space. By representing $\boldsymbol{\varphi}$ in this coordinate system, its

pre-image estimation can be obtained thanks to the inner product between its pre-image and the input samples. This method provides a natural pre-image technique, requires only linear algebra, and is universal in the sense of being independent of the type of adopted kernels. The role of the kernel in the pre-image solution may vanish under some regularisation specifications. Recently, [51] propose a method based on kernel PCA and kernel regression or kernel dependency [23] to reconstruct pre-image estimation. In the following, we describe these five major methods to estimate the pre-image $\boldsymbol{x}$ of a given feature vector $\boldsymbol{\varphi} = \sum_{i=1}^{N} \gamma_i \, \Phi(\boldsymbol{x}_i) \in \mathcal{H}$.

3.1 Exact pre-image solution

In Schölkopf [41], according to the theorem 3.1, it is shown that if an exact pre-image of $\boldsymbol{\varphi}$ exists, it would be easy to compute $\boldsymbol{x}$ such that $\Phi(\boldsymbol{x}) = \boldsymbol{\varphi}$.

Theorem 3.1. *Given a feature vector* $\boldsymbol{\varphi} = \sum_{i=1}^{N} \gamma_i \, \Phi(\boldsymbol{x}_i)$. *If there exists a sample* $\boldsymbol{x} \in \mathbb{R}^d$ *such that*

$$\Phi(\boldsymbol{x}) = \boldsymbol{\varphi},$$

and if κ *is an invertible kernel e.i* $\kappa(\boldsymbol{x}, \boldsymbol{y}) = f_\kappa(\langle \boldsymbol{x}, \boldsymbol{y} \rangle)$ *with invertible function* f_κ, *then we can compute* $\boldsymbol{x}$ *as*

$$\boldsymbol{x} = \sum_{i=1}^{d} f_\kappa^{-1} \Big(\sum_{j=1}^{N} \gamma_j \, \kappa(\boldsymbol{x}_j, \boldsymbol{e}_i) \Big) \, \boldsymbol{e}_i,$$

with $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$ *is any orthonormal basis of input space.*

Let $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$ be any orthonormal basis of input space $\mathcal{X} = \mathbb{R}^d$. The pre-image $\boldsymbol{x}$ is represented by:

$$\boldsymbol{x} = \sum_{i=1}^{d} \langle \boldsymbol{x}, \boldsymbol{e}_i \rangle \, \boldsymbol{e}_i.$$

As f_κ is invertible, the inner products are rewritten by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = f_\kappa^{-1}\big(\kappa(\boldsymbol{x}, \boldsymbol{y})\big) = f_\kappa^{-1}\big(\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle\big), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}.$$

Hence, $\boldsymbol{x}$ may be expandes as:

$$\boldsymbol{x} = \sum_{i=1}^{d} f_{\kappa}^{-1} \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{e}_i) \rangle \, \boldsymbol{e}_i. \tag{3.2}$$

As $\boldsymbol{\varphi} = \Phi(\boldsymbol{x}) = \sum_{i=1}^{N} \gamma_i \, \Phi(\boldsymbol{x}_i)$, $\boldsymbol{x}$ is then obtained as

$$\begin{aligned} \boldsymbol{x} &= \sum_{i=1}^{d} f_{\kappa}^{-1} \langle \sum_{j=1}^{N} \gamma_j \, \Phi(\boldsymbol{x}_j), \Phi(\boldsymbol{e}_i) \rangle \, \boldsymbol{e}_i \\ &= \sum_{i=1}^{d} f_{\kappa}^{-1} \left(\sum_{j=1}^{N} \gamma_j \, \kappa(\boldsymbol{x}_j, \boldsymbol{e}_i) \right) \boldsymbol{e}_i. \end{aligned} \tag{3.3}$$

This method provides the exact solution of pre-image problem. However, the assumption of the existence of the pre-image $\boldsymbol{x}$ is not satisfied in many situations. For instance, we consider the feature mapping that is formulated by

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \tag{3.4}$$

$$\boldsymbol{x} \mapsto \kappa(., \boldsymbol{x}) \tag{3.5}$$

Only feature vectors $\boldsymbol{\varphi}$ in $\mathcal{H}$, that can be written as $\kappa(., \boldsymbol{x})$, have an exact pre-image solution under the mapping Φ. In the case of Gaussian kernels, if a feature vector $\boldsymbol{\varphi}$ has an exact pre-image solution $\boldsymbol{x}$, that means that the Gaussian function $\kappa(., \boldsymbol{x})$ is as a linear combination of the Gaussian functions $\kappa(., \boldsymbol{x}_i)$ as:

$$\begin{aligned} \boldsymbol{\varphi} &= \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i) \\ \Phi(\boldsymbol{x}) &= \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i) \\ \kappa(., \boldsymbol{x}) &= \sum_{i=1}^{N} \gamma_i \kappa(., \boldsymbol{x}_i) \end{aligned} \tag{3.6}$$

However, in [38], it is shown that any Gaussian function can not be written as a linear combination of Gaussian functions of the other samples, the exact pre-image $\boldsymbol{x}$ of feature vector $\boldsymbol{\varphi}$ obtained by Eq. (3.6) may not exist. Furthermore, the inversion of f_{κ} is only satisfied for some kernels as polynomial and sigmoid kernels. Without the existence of an exact solution, other methods focus on finding an approximation for the pre-image solution, denoted by $\boldsymbol{x}^*$ in the following.

3.2 Pre-image estimation by fixed-point iteration

In [46], a new approach is investigated for the pre-image estimation under the Gaussian kernel. This method proposes an iterative scheme to estimate the pre-image $\boldsymbol{x}^*$ of a given feature vector φ, where under particular Gaussian kernels, the fixed-point iteration method can be used. The problem is formalised as a nonlinear optimisation:

$$\begin{aligned}\boldsymbol{x}^* &= \arg\min_{\boldsymbol{z}\in\mathcal{X}} \parallel \Phi(\boldsymbol{z}) - \boldsymbol{\varphi} \parallel^2_{\mathcal{H}} \\ &= \arg\min_{\boldsymbol{z}\in\mathcal{X}} \Big(\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle - 2 \langle \Phi(\boldsymbol{z}), \boldsymbol{\varphi} \rangle + \langle \boldsymbol{\varphi}, \boldsymbol{\varphi} \rangle \Big). \end{aligned} \tag{3.7}$$

As $\boldsymbol{\varphi} = \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i)$, we have

$$\langle \boldsymbol{\varphi}, \boldsymbol{\varphi} \rangle = \langle \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i), \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i) \rangle = \boldsymbol{\gamma}^T K \boldsymbol{\gamma}. \tag{3.8}$$

Hence, $\langle \boldsymbol{\varphi}, \boldsymbol{\varphi} \rangle$ is independent of $\boldsymbol{z}$. After the kernel κ normalisation and for all $\boldsymbol{z}$, $\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle = \kappa(\boldsymbol{z}, \boldsymbol{z})$ is constant. Rather than minimising Eq. (3.7), we can consider the maximisation problem:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{z}\in\mathcal{X}} \langle \Phi(\boldsymbol{z}), \boldsymbol{\varphi} \rangle. \tag{3.9}$$

Substituting Eq. (3.1) into Eq. (3.9), we deduce

$$\begin{aligned}\boldsymbol{x}^* &= \arg\max_{\boldsymbol{z}\in\mathcal{X}} \langle \Phi(\boldsymbol{z}), \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i) \rangle = \arg\max_{\boldsymbol{z}\in\mathcal{X}} \sum_{i=1}^{N} \gamma_i \langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{x}_i) \rangle, \\ &= \arg\max_{\boldsymbol{z}\in\mathcal{X}} \sum_{i=1}^{N} \gamma_i \kappa(\boldsymbol{z}, \boldsymbol{x}_i) = \arg\max_{\boldsymbol{z}\in\mathcal{X}} F(\boldsymbol{z}) \end{aligned} \tag{3.10}$$

The optimisation problem in Eq. (3.10) can be solved by using gradient descent. In particular, for a Gaussian kernel $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{||\boldsymbol{x}-\boldsymbol{y}||^2}{2\sigma^2})$, we can use fixed-point iterative method. We take the derivative of $F(\boldsymbol{z})$ with respect to $\boldsymbol{z}$ and set

it to zero, then we have

$$\frac{\partial F}{\partial \boldsymbol{z}} = \frac{\partial}{\partial z} \sum_{i=1}^{N} \gamma_i \, \kappa(\boldsymbol{z}, \boldsymbol{x}_i) = 0,$$
$$\Rightarrow \boldsymbol{z} = \frac{\sum_{i=1}^{N} \gamma_i \, \exp\Big(- ||\boldsymbol{z} - \boldsymbol{x}_i||^2 / (2\,\sigma^2) \Big) \, \boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma_i \, \exp\Big(- ||\boldsymbol{z} - \boldsymbol{x}_i||^2 / (2\,\sigma^2) \Big)}.$$

By the fix point iteration theorem, $\boldsymbol{x}^*$ is a convergent point of the sequences $\{\boldsymbol{z}_n\}$

$$\Rightarrow \boldsymbol{z}_{n+1} = \frac{\sum_{i=1}^{N} \gamma_i \, \exp\Big(- ||\boldsymbol{z}_n - \boldsymbol{x}_i||^2 / (2\,\sigma^2) \Big) \boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma_i \exp\Big(- ||\boldsymbol{z}_n - \boldsymbol{x}_i||^2 / (2\,\sigma^2) \Big)}. \tag{3.11}$$

This method gives an approximate solution. But even this is nontrivial as the dimensionality of the feature space can be infinite. [46] cast this as a nonlinear optimisation problem, which, for particular choices of kernels (such as the Gaussian kernel), can be solved by a fixed-point iteration method. However, the considered optimisation problem is highly non-convex, this method suffers from numerical instabilities. Moreover, as in any nonlinear optimisation problem, one can get trapped in a local minimum an the obtained pre-image estimation is thus sensitive to the initial guess.

3.3 Pre-image estimation by distance constraints

Kwok et al. [55] address the problem of finding the pre-image of a given feature vector in the feature space induced by a kernel. Unlike the method proposed in [46] which relies on nonlinear optimisation, the Kwok et al. [55] method directly finds the location of the pre-image based on the distance constraints in the feature space. It is non-iterative, involves only linear algebra, and does not suffer from numerical instability or local minimum problems. The distances between $\boldsymbol{\varphi}$ and $\Phi(\boldsymbol{x}_i)$ and their relation to the distances between the pre-image $\boldsymbol{x}^*$ and $\boldsymbol{x}_i$ are used to estimate $\boldsymbol{x}^*$, as illustrated in Figure 3.2. The main steps of the proposed

approach are detailed in the following.

Let $\tilde{\boldsymbol{d}}^2(\boldsymbol{\varphi}, \Phi(\boldsymbol{x}_j))$ be the distances squared into the feature space between $\boldsymbol{\varphi}$ and $\Phi(\boldsymbol{x}_j)$ defined as

$$\begin{aligned}\tilde{d}^2(\boldsymbol{\varphi}, \Phi(\boldsymbol{x}_j)) &= \|\boldsymbol{\varphi} - \Phi(\boldsymbol{x}_j)\|_{\mathcal{F}}^2 \\ &= \langle \boldsymbol{\varphi}, \boldsymbol{\varphi} \rangle - 2\langle \boldsymbol{\varphi}, \Phi(\boldsymbol{x}_j) \rangle + \langle \Phi(\boldsymbol{x}_j), \Phi(\boldsymbol{x}_j) \rangle \\ &= \langle \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i), \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i) \rangle - 2\langle \sum_{i=1}^{N} \gamma_i \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j) \\ &= \boldsymbol{\gamma}^T K \boldsymbol{\gamma} - 2\boldsymbol{\gamma}^T K_{i.} + K_{ij}. \end{aligned} \tag{3.12}$$

where $K_{i.}$ and K_{ij} are respectively the ith row and the ij entry of Gram matrix K.

Let $\Phi(\dot{\boldsymbol{x}}_1), \ldots, \Phi(\dot{\boldsymbol{x}}_n)$ denote the n-th closest elements to $\boldsymbol{\varphi}$:

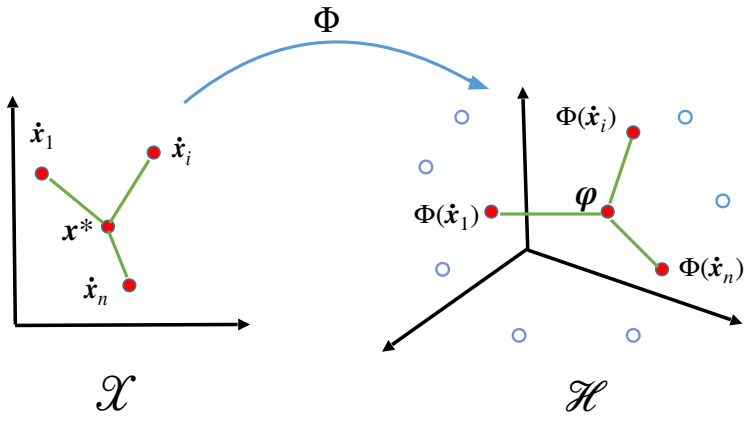


Figure 3.2: The pre-image estimation by distance constraints

$$\{\Phi(\dot{\boldsymbol{x}}_1), \ldots, \Phi(\dot{\boldsymbol{x}}_n)\} = \underset{\{\Phi(\boldsymbol{z}_1), \ldots, \Phi(\boldsymbol{z}_n)\} \subset \Phi(X)}{\arg\min} \sum_{i=1}^{n} \tilde{d}^2(\boldsymbol{\varphi}, \Phi(\boldsymbol{z}_i)). \tag{3.13}$$

For an isotropic kernel, the relation $d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = g(\tilde{d}^2(\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)))$ between the distances in the input and the feature spaces can be established. For instance, with Gaussien kernel, given a parameter σ, we have

$$g(z) = -2\sigma^2 ln(1 - \frac{1}{2}z) \quad \text{for} \quad z \in \mathbb{R}. \tag{3.14}$$

A solution is then deployed to determine the pre-image $\boldsymbol{x}$ such that

$$[d^2(\boldsymbol{x}, \dot{\boldsymbol{x}}_1), \ldots, d^2(\boldsymbol{x}, \dot{\boldsymbol{x}}_n)] = \big[g(\widetilde{d^2}(\Phi(\boldsymbol{x}), \Phi(\dot{\boldsymbol{x}}_1))), ..., g(\widetilde{d^2}(\Phi(\boldsymbol{x}), \Phi(\dot{\boldsymbol{x}}_n)))\big],$$
$$[\| \boldsymbol{x} - \dot{\boldsymbol{x}}_1 \|^2, \ldots, \| \boldsymbol{x} - \dot{\boldsymbol{x}}_n \|^2] = [d^2(\boldsymbol{x}, \dot{\boldsymbol{x}}_1), \ldots, d^2(\boldsymbol{x}, \dot{\boldsymbol{x}}_n)].$$

For that, an SVD is deployed on the centered version of the submatrix $X_n = [\dot{\boldsymbol{x}}_1, ..., \dot{\boldsymbol{x}}_n]$, namely

$$X_n \, (I_n - \mathbb{1}_n) = U \, \Lambda \, V^T = U \, Z, \tag{3.15}$$

where $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_q]$ is the $d \times q$ matrix of the left-singular vectors. Let $Z = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n] = \Lambda V^T$ be the $q \times n$ matrix giving the projections of $\dot{\boldsymbol{x}}_i$ on the $\boldsymbol{u}_j$'s orthonormal vectors. We see the distance between $\boldsymbol{x}_i$ to the origin equal to $d_0^2 = \| \boldsymbol{z}_i \|^2$. Let $\boldsymbol{z}$ be the presentation of $\boldsymbol{x}$ in new system basic U, we have

$$d^2(\boldsymbol{z}, \boldsymbol{z}_i) = d^2(\boldsymbol{x}, \dot{\boldsymbol{x}}_i). \tag{3.16}$$

Following [24], Eq. (3.16) can be shown to satisfy:

$$-2Z^T \, \boldsymbol{z} = (\boldsymbol{d}^2 - \boldsymbol{d}_0^2) - \frac{1}{n} \mathbf{1}_n \, \mathbf{1}_n^T \, (\boldsymbol{d}^2 - \boldsymbol{d}_0^2), \tag{3.17}$$

with $\boldsymbol{d}_0^2 = [\|\boldsymbol{z}_1\|^2, ..., \|\boldsymbol{z}_n\|^2]^T$, $\boldsymbol{d}^2 = [d_1^2, \ldots, d_n^2]^T$ and $\boldsymbol{d}_i^2 = g(\widetilde{d^2}(\Phi(\boldsymbol{x}), \Phi(\dot{\boldsymbol{x}}_i)))$. We multiple Z to Eq. (3.17):

$$-2ZZ^T \, \boldsymbol{z} = Z(\boldsymbol{d}^2 - \boldsymbol{d}_0^2) - \frac{1}{n} \, Z \, \mathbf{1}_n \, \mathbf{1}_n^T \, (\boldsymbol{d}^2 - \boldsymbol{d}_0^2). \tag{3.18}$$

As Z is centered e.i. $Z \, \mathbf{1}_n = 0$, Eq. (3.18) is thus

$$\begin{aligned} -2ZZ^T \, \boldsymbol{z} &= Z(\boldsymbol{d}^2 - \boldsymbol{d}_0^2), \\ \boldsymbol{z}^* &= -\frac{1}{2} \, (Z \, Z^T)^{-1} \, Z \, (\boldsymbol{d}^2 - \boldsymbol{d}_0^2). \end{aligned} \tag{3.19}$$

The pre-image $\boldsymbol{x}^*$ estimation is then obtained as:

$$\boldsymbol{x}^* = U \, \boldsymbol{z}^* + \frac{1}{n} X_n \mathbf{1}_n. \tag{3.20}$$

The proposed method that directly finds the location of the pre-image based on distance constraints. Applying a multidimensional scaling technique (MDS) leads

to an inverse map estimate and thus to the pre-image. This method opens the door to a range of other techniques taking prior knowledge from input data in both space, such as manifold learning ([29]) and out-of-sample methods ([45],[49]). This method is non-iterative, involves only linear algebra and does not suffer from numerical instability or the local minimum problem. Moreover, it can be applied equally well to both isotropic kernel and dot product kernels. As the method uses distance constraints involved on the neighbourhood of $\boldsymbol{\varphi}$, there is only the local information that may affect the pre-image estimation.

3.4 Pre-image estimation by isometry preserving

Solving the pre-image problem is pioneered by Mika's fixed point iterative optimisation technique. Recent approaches take advantage of prior knowledge provided by the input data , whose coordinates are known in the input space and implicitly in the feature space, a first step in this direction made by Kwork's algorithm based on multidimensional scaling. Using such prior knowledge, Paul Honeine [48] proposes a new approach to learn the pre-image, with the elegance that only linear algebre is involved. This method focus on learning a new coordinate system in RKHS, that preserves an isometry with the input space. That means the inner products between the input data are preserved in both representations. The representation of a given feature vector $\boldsymbol{\varphi}$ in the new coordinate system can give us some information to estimate its pre-image $\boldsymbol{x}$. This proposed approach (illustrated in Figure 3.3) proceeds in two steps. First, a coordinate system, spanned by the feature vectors $\{\Phi(\boldsymbol{x}_i)\}_{i=1}^N$ is learned to ensure an isometry with the input space; subsequently, the coordinate system is used to estimate the pre-image $\boldsymbol{x}$ of $\boldsymbol{\varphi}$. These two main steps are summarised in the followings:

First, let $\Psi = \{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_p\}$ $(p \leq N)$ be a coordinate system in the feature space with

$$\boldsymbol{\psi}_k = \sum_{i=1}^{N} \alpha_{ik} \Phi(\boldsymbol{x}_i) = \Phi(X)\boldsymbol{\alpha_k}, \quad \forall k = 1, \ldots, p,$$

e.i., each $\boldsymbol{\psi}_k$ is defined as a linear combination of the mapped input samples $\Phi(\boldsymbol{x}_i)$. This coordinate system can be written in a matrix form: $\Psi = \Phi(X)A$, where $A = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p] \in \mathbb{R}^{N \times p}$ and $\Phi(X) = [\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N)]$. The projection of $\Phi(\boldsymbol{x})$ onto the coordinate system Ψ is defined by

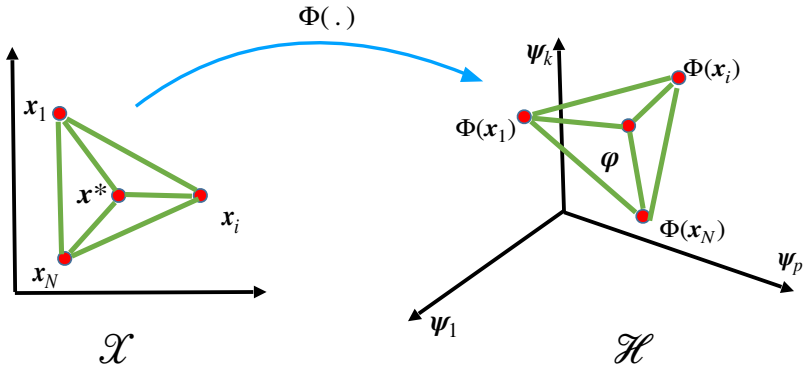


Figure 3.3: The pre-image estimation by isometry preserving

$$\begin{aligned} P(\Phi(\boldsymbol{x})) &= [P_1(\Phi(\boldsymbol{x})), \ldots, P_p(\Phi(\boldsymbol{x}))]^T \\ &= [\langle \boldsymbol{\psi}_1, \Phi(\boldsymbol{x}) \rangle, \ldots, \langle \boldsymbol{\psi}_p, \Phi(\boldsymbol{x}) \rangle]^T \\ &= [\langle \Phi(X)\boldsymbol{\alpha_1}, \Phi(\boldsymbol{x}) \rangle, \ldots, \langle \Phi(X)\boldsymbol{\alpha_p}, \Phi(\boldsymbol{x}) \rangle]^T \\ &= [\boldsymbol{k_x}\boldsymbol{\alpha}_1, \ldots, \boldsymbol{k_x}\boldsymbol{\alpha}_p]^T \\ &= (\boldsymbol{k_x}[\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p])^T \\ &= (\boldsymbol{k_x}A)^T. \end{aligned}$$

with $\boldsymbol{k_x} = \Big(\kappa(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}, \boldsymbol{x}_N)\Big)$. Similarly, the projection of the mapped input set $\Phi(X)$ can be determined by

$$\begin{aligned} P(\Phi(X)) &= [P(\Phi(\boldsymbol{x}_1)), \ldots, P(\Phi(\boldsymbol{x}_N))] \\ &= [(\boldsymbol{k}_{\boldsymbol{x}_1}A)^T, \ldots, (\boldsymbol{k}_{\boldsymbol{x}_N}A)^T] \\ &= (KA)^T. \end{aligned}$$

Hence, to estimate the coordinate system Ψ that is isometric with the input space, the problem is determined by

$$\begin{aligned}
\langle P(\Phi(\boldsymbol{x}_i)), P(\Phi(\boldsymbol{x}_j))\rangle &= \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle \quad \forall i,j \in \{1,\ldots,N\},\\
\langle P(\Phi(X)), P(\Phi(X))\rangle &= \langle X, X\rangle,\\
\langle (KA)^T, (KA)^T\rangle &= \langle X, X\rangle,\\
KAA^TK &= X^TX. \qquad (3.21)
\end{aligned}$$

This leads to the following optimisation problem:

$$\arg\min_A \|X^TX - KAA^TK\|_{\mathcal{F}} + \lambda \sum_{k=1}^{p} \|\boldsymbol{\psi_i}\|^2, \qquad (3.22)$$

where λ is a the regularisation parameter of the term $\| \Psi \|^2_{\mathcal{F}}$ developed as follows:

$$\begin{aligned}
\sum_{k=1}^{p} \|\boldsymbol{\psi}_i\|^2 &= \sum_{k=1}^{p} \langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_i\rangle,\\
&= \sum_{k=1}^{p} \langle \Phi(X)\boldsymbol{\alpha}_i, \Phi(X)\boldsymbol{\alpha}_i\rangle,\\
&= \sum_{k=1}^{p} \Phi(X)\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T\Phi(X)^T,\\
&= \Phi(X)AA^T\Phi(X)^T = tr(KAA^T).
\end{aligned}$$

Hence, Eq. (3.22) is rewritten by using matrix formulation as

$$\arg\min_A \frac{1}{2} \| X^T X - K A A^T K \|^2_{\mathcal{F}} + \lambda \operatorname{tr}(KA A^T) \qquad (3.23)$$

We write briefly $A A^T$ by Z, and call $f(Z) = \frac{1}{2} \| X^T X - K Z K \|^2_{\mathcal{F}} + \lambda \operatorname{tr}(KZ)$, the derivation of f respected to Z computed by

$$\frac{\partial f}{\partial Z} = \frac{\partial[\frac{1}{2} tr(X^T X - K Z K)(X^T X - K Z K)^T + \lambda tr(KZ)]}{\partial Z} = 0, \qquad (3.24)$$

$$\frac{\partial f}{\partial Z} = \frac{\partial[\frac{1}{2}\,(tr((X^T\,X)^2) - tr((X^T\,X)\,K\,Z^T\,K) - tr(K\,Z\,K\,(X^T\,X))+}{\partial Z}$$
$$\frac{+tr(K\,Z\,K\,K\,Z^T\,K)) + \lambda\,tr(KZ)]}{\partial Z} = 0,$$
$$\frac{\partial f}{\partial Z} = \frac{1}{2}(0 - K\,X^T\,X\,K - K\,X^T\,X\,K + K\,K\,Z\,K\,K + K\,K\,Z\,K\,K) + \lambda\,K = 0,$$
$$K\,K\,Z\,K\,K = K\,P\,K - \lambda\,K = K\,(P - \lambda\,K^{-1})\,K,$$
$$Z = K^{-1}\,(X^T\,X - \lambda\,K^{-1})\,K^{-1},$$
$$AA^T = K^{-1}\,(X^T\,X - \lambda\,K^{-1})\,K^{-1}. \tag{3.25}$$

We use $A\,A^T$ to solve the pre-image problem rather that using A. Based on the isometric property, the pre-image $\boldsymbol{x}^*$ is estimated as

$$\begin{aligned}
\langle P(\Phi(\boldsymbol{x}_i)), P(\boldsymbol{\varphi})\rangle &= \langle \boldsymbol{x}_i, \boldsymbol{x}^*\rangle, \quad \forall i = 1, \ldots, N,\\
\langle P(\Phi(X)), P(\boldsymbol{\varphi})\rangle &= \langle X, \boldsymbol{x}^*\rangle,\\
\langle (KA)^T, (K\boldsymbol{\gamma})^T\rangle &= \langle X, \boldsymbol{x}^*\rangle,\\
KAA^T\boldsymbol{\gamma} &= X^T\boldsymbol{x}^*.
\end{aligned} \tag{3.26}$$

Substituting Eq. (3.25) into Eq. (3.26), we have:

$$\begin{aligned}
X^T\,\boldsymbol{x}^* &= K\,K^{-1}\,(X^T\,X - \lambda\,K^{-1})\,K^{-1}\,K\boldsymbol{\gamma},\\
X^T\,\boldsymbol{x}^* &= (X^T\,X - \lambda\,K^{-1})\boldsymbol{\gamma},\\
\boldsymbol{x}^* &= \arg\min_{\boldsymbol{z}} \parallel X^T\,\boldsymbol{z} - (X^T\,X - \lambda\,K^{-1})\boldsymbol{\gamma} \parallel^2 .
\end{aligned} \tag{3.27}$$

The problem (3.27) defines a standard overdetermined equation system ($N \gg d$) that can be resolved as a least-square minimisation problem (i.e., any technique such as the pseudo-inverse or the eigendecomposition). The pre-image estimation is then:

$$\boldsymbol{x}^* = (X\,X^T)^{-1}\,X\,(X^T\,X - \lambda\,K^{-1})\,\boldsymbol{\gamma}. \tag{3.28}$$

As opposed to previous method, the proposed method neither suffers from the numerical instability, not requires computing the distances in the input space and the feature space. Using the inner product information in both spaces, this method provides a coordinate system in $\mathcal{H}$ space to learn the inverse mapping. The major

advantage of this method resides on its simplicity in dealing with the optimisation issus, thanks to conventional linear algebra. Moreover, it is universal in the sense that it is independent of the type of kernels and the feature under investigation. Note that, for lower values of $\lambda \approx 0$, the obtained solution is no longer dependent of the kernel κ:

$$\boldsymbol{x}^* = (X\,X^T)^{-1}\,X\,X^T\,X\,\boldsymbol{\gamma},$$
$$\boldsymbol{x}^* = X\boldsymbol{\gamma}$$

3.5 Pre-image estimation by kernel regression

Bakir [51] the pre-image estimation consists in learning a kernel regression function that maps all the $\Phi(\boldsymbol{x}_i)$ in the feature space $\mathcal{H}$ related to the kernel K to $\boldsymbol{x}_i$ in the input space $\mathbb{R}^d$. For that, first kernel PCA is deployed to embed $\Phi(X)$ into the subspace spanned by the eigenvectors $\boldsymbol{u}_1, ..., \boldsymbol{u}_p$ defined in Eq. (2.6), with $\boldsymbol{u}_j = \Phi(X)\boldsymbol{\alpha}_j$. This embedding can be defined by:

$$P : \mathcal{H} \to \mathbb{R}^p$$
$$\Phi(\boldsymbol{x}) \mapsto P(\Phi(\boldsymbol{x})) = (\boldsymbol{k}_{\boldsymbol{x}}\boldsymbol{\alpha})^T,$$

where $P(\Phi(\boldsymbol{x}))$ is the coefficient of $\Phi(\boldsymbol{x})$ with respect to PCA system $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p\}$ in Eq. (2.12). Then, a kernel regression is learned between the set of the projections in the kernel PCA subspace and X by the pre-image mapping Γ as

$$\Gamma : \mathbb{R}^p \to \mathbb{R}^d$$
$$P(\Phi(\boldsymbol{x})) \mapsto \Gamma(P(\Phi(\boldsymbol{x}))) = (\Gamma_1(P(\Phi(\boldsymbol{x}))), \ldots, \Gamma_d(P(\Phi(\boldsymbol{x}))))^T,$$

with

$$\Gamma_i = \arg\min_{\Gamma_i} \sum_{j=1}^{N} ||\boldsymbol{x}_j(i) - \Gamma_i(P(\Phi(\boldsymbol{x}_j))||^2 + \lambda\Omega(\Gamma_i) \quad \forall i = 1, \ldots, d, \tag{3.29}$$

with $\Gamma_i(P(\Phi(\boldsymbol{x})) = \sum_{j=1}^{N} \delta_i^j \widehat{\kappa}(P(\Phi(\boldsymbol{x})), P(\Phi(\boldsymbol{x}_i)))$. Here, $\widehat{\kappa}(.,.)$ is a new kernel on $\mathbb{R}^p$ and δ_i^j are unknown parameters to estimate .

Denote that $B \in \mathbb{R}^{d \times N}$ is the regression coefficient matrix and $\widehat{K}$ is the Gram matrix with entries $\widehat{K}_{ii'} = \widehat{\kappa}(P(\Phi(\boldsymbol{x}_i)), P(\Phi(\boldsymbol{x}_{i'})))$. The problem is typically rewritten as

$$\underset{B \in \mathbb{R}^{d \times N}}{\arg\min} \|X - B\widehat{K}\|_{\mathcal{F}}^2 + \lambda \|B\|_{\mathcal{F}}^2. \tag{3.30}$$

Note $f(B) = \| X - B\widehat{K} \|_{\mathcal{F}}^2 + \lambda \| B \|_{\mathcal{F}}^2$. To minimise the function f, we put its derivation with respect to B equal to zero:

$$\begin{aligned}
\frac{\partial f}{\partial B} = \frac{\partial \| X - B\widehat{K} \|_F^2 + \lambda \| B \|_F^2}{\partial B} &= 0, \\
\frac{\partial \Big(tr((X - B\widehat{K})(X - B\widehat{K})^T) + \lambda\, tr(B\, B^T) \Big)}{\partial B} &= 0, \\
\frac{\partial \Big(tr(X\, X^T) - tr(X\, \widehat{K}^T B^T) - tr(B\widehat{K}\, X^T) + tr(B\widehat{K}\, \widehat{K}^T B^T) + \lambda\, tr(B\, B^T) \Big)}{\partial B} &= 0.
\end{aligned}$$

As K is symmetric, then we have:

$$\begin{aligned}
0 - X\,\widehat{K} - X\,\widehat{K} + 2\,B\widehat{K}^2 + \lambda\, 2\; BI_N &= 0, \\
B(\widehat{K}^2 + \lambda\, I_N) &= X\,\widehat{K}, \\
B &= X\,\widehat{K}(\widehat{K}^2 + \lambda\, I_N)^{-1}.
\end{aligned} \tag{3.31}$$

For a feature vector $\boldsymbol{\varphi} = \Phi(X)\boldsymbol{\gamma} \in \mathcal{H}$, its pre-image $\boldsymbol{x}^*$ is then estimated as:

$$\boldsymbol{x}^* = B(\widehat{\boldsymbol{k}}_{P(\boldsymbol{\varphi})})^T, \tag{3.32}$$

with

$$\widehat{\boldsymbol{k}}_{P(\boldsymbol{\varphi})} = [\widehat{\kappa}(P(\boldsymbol{\varphi}), P(\Phi(\boldsymbol{x}_1))), ..., \widehat{\kappa}(P(\boldsymbol{\varphi}), P(\Phi(\boldsymbol{x}_N)))], \tag{3.33}$$

where

$$\begin{aligned} P(\boldsymbol{\varphi}) &= (\boldsymbol{k}_{\boldsymbol{\varphi}}\boldsymbol{\alpha})^T, \\ &= \big((\langle\boldsymbol{\varphi}, \Phi(\boldsymbol{x}_1)\rangle, \ldots, \langle\boldsymbol{\varphi}, \Phi(\boldsymbol{x}_N)\rangle)\boldsymbol{\alpha}\big)^T, \\ &= \big((\langle\Phi(X)\boldsymbol{\gamma}, \Phi(\boldsymbol{x}_1)\rangle, \ldots, \langle\Phi(X)\boldsymbol{\gamma}, \Phi(\boldsymbol{x}_N)\rangle)\boldsymbol{\alpha}\big)^T, \\ &= \big((\boldsymbol{k}_{\boldsymbol{x}_1}\boldsymbol{\gamma}, \ldots, \boldsymbol{k}_{\boldsymbol{x}_N}\boldsymbol{\gamma})\boldsymbol{\alpha}\big)^T, \\ &= \big(\boldsymbol{\gamma}^T K\boldsymbol{\alpha}\big)^T, \\ &= \boldsymbol{\alpha}^T K\boldsymbol{\gamma}, \end{aligned}$$

and

$$P(\Phi(\boldsymbol{x}_i)) = (\boldsymbol{k}_{\boldsymbol{x}_i}\boldsymbol{\alpha})^T = \boldsymbol{\alpha}^T\boldsymbol{k}_{\boldsymbol{x}_i}^T,$$

with $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p]$.

The method introduces a technique based on kernel principal component analysis and regression to reconstruct corresponding pre-image in the input space. This method avoids difficult and unstable numerical optimisation, is easy to implement, and permits the computation of pre-images in discrete input space. By using the pre-image mapping, each pre-image can be computed very efficiently, and there are no longer issues with complex optimisation code. Moreover, the method proposed a nonlinear model to adapt to the flexible data as well as keep the important role of the kernel. However, as substituting feature vector to its projection by kernel PCA, it requires that the used input samples be representative. Besides, the choice of kernel $\widehat{\kappa}$ for kernel regression is trick to select.

3.6 Overview

To sum up, the three major methods (Section 3.3, 3.4, 3.5) presented above define three different approaches for pre-image estimation problem. First of all, all the methods involve only linear algebra and propose solutions that don't

suffer from numerical instabilities. In Kwok et al. [55], the solution is mainly requiring the definition of a relation between the distances into the input and the kernel feature spaces. That requirement limite the Kwok et al. [55] approach to linear or isotropic kernels. Honeine et al. [48] alleviate that point by proposing a closed-form solution that is applicable to any type of kernels. Furthermore, while in Honeine et al. [48] the pre-image estimation is obtained by learning a linear transformation into the feature space that preserves the isometry between the input and the feature space, in Bakir et al. [51], the pre-image estimation is obtained by using a nonlinear kernel regression that predicts the input samples from their images into the feature space. Finally, while both [48] and [51] proposals involve the whole training samples for pre-image estimation, Kwok et al. [55] uses only the samples on the neighborhood of $\boldsymbol{\varphi}$, which offers a significant speed-up; highly valuable in the case of large scale data.

4

Pre-image estimation for time series kernel analytics

While kernel machinery has been increasingly investigated with success for time series analytics [60, 17, 39, 11], the pre-image problem for temporal data remains in its infancy. In addition, time series data, that may involve varying delays and be of different lengths, are naturally lying in a non-Euclidean input space, that makes the pre-image methods presented in Chapter 3 inapplicable. This Chapter proposes a pre-image estimation approach for time series kernel analytics, that consists of two steps. In the first step, a time warp function, driven by distance constraints in the feature space, is defined to embed time series in a metric space. Subsequently, the time series pre-image estimation is cast as learning a linear or

a nonlinear transformation that ensures a local isometry between the time series embedding space and the feature space.

4.1 Pre-image estimation by isometry preserving between X and Y

Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ be a $d \times N$ matrix giving the description of N samples $\boldsymbol{x}_i \in \mathbb{R}^d$, and $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]$ be a $q \times N$ matrix giving the description of the same N samples. We formalise the pre-image problem as the estimation of a linear transformation R that ensures an isometry between X and Y:

$$R^* = \arg\min_R \|X^T X - Y^T R Y\|_{\mathcal{F}}^2 \quad \text{with } R \in \mathbb{R}^{q \times q}. \tag{4.1}$$

Assume that YY^T is invertible ($q << N$). The closed-form solution can be obtained as:

$$R^* = (Y Y^T)^{-1} Y X^T X Y^T (Y Y^T)^{-1}. \tag{4.2}$$

Based on the inner product preservation, the pre-image estimation $\boldsymbol{x}^*$ of a given $\boldsymbol{y} \in [Y]$ is then:

$$\boldsymbol{x}^* = (X X^T)^{-1} X Y^T R^* \boldsymbol{y}. \tag{4.3}$$

If XX^T is not invertible, we can add the regularity term as

$$\boldsymbol{x}^* = (X X^T + \lambda I_d)^{-1} X Y^T R^* \boldsymbol{y}.$$

Substituting Eq. (4.2) into Eq. (4.3), we have

$$\boldsymbol{x}^* = (XX^T)^{-1} XY^T (YY^T)^{-1} YX^T XY^T (YY^T)^{-1} \boldsymbol{y}. \tag{4.4}$$

If Y are invertible, then $(YY^T)^{-1} = (Y^T)^{-1} Y^{-1}$ and the Eq. (4.4) is then simplified as:

$$\boldsymbol{x}^* = XY^{-1} \boldsymbol{y}. \tag{4.5}$$

4.2 Pre-image estimation by isometry preserving between X and $\Phi(X)$

Let $X = [\boldsymbol{x}_1 \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$ be a matrix giving the description of N samples $\boldsymbol{x}_i$. In the context of kernel machinery, $\Phi(X)$ is the embedding of X into the RKHS under the kernel κ. The proposed pre-image method relies on learning a linear transformation R in the feature space that ensures an isometry between X and $\Phi(X)$. This result, is then extended to learn a nonlinear transformation R.

4.2.1 Learning linear transformation for pre-image estimation

The main idea to solve the pre-image problem is the isometry preserving in the same spirit as the method described in Section 4.1. For this purpose, we formalise the pre-image problem as the estimation of the square matrix R that establishes an isometry between X and $\Phi(X)$, by solving the optimization problem

$$R^* = \arg\min_R \|X^T X - \Phi(X)^T R \, \Phi(X)\|_{\mathcal{F}}^2. \tag{4.6}$$

By using a kernel PCA where a relevant subspace is considered, an explicit description $P(\Phi(X)) \in \mathbb{R}^{p \times N}$ of $\Phi(X)$ is given and Eq. (4.6) can thus be rewritten as:

$$R^* = \arg\min_{R \in \mathbb{R}^{p \times p}} \|X^T X - P(\Phi(X))^T R \, P(\Phi(X))\|_{\mathcal{F}}^2. \tag{4.7}$$

As $P(\Phi(X))\, P(\Phi(X))^T$ is invertible, similarly to Eq. (4.2), a closed-form solution is given by:

$$\begin{aligned} R^* = \left(P(\Phi(X))P(\Phi(X))^T\right)^{-1} P(\Phi(X))X^T X \\ P(\Phi(X))^T \left(P(\Phi(X))P(\Phi(X))^T\right)^{-1}. \end{aligned} \tag{4.8}$$

The pre-image estimation $\boldsymbol{x}^*$ of $\boldsymbol{\varphi} = \sum_{i=1}^N \gamma_i \Phi(\boldsymbol{x}_i)$, is then given by:

$$\boldsymbol{x}^* = (XX^T)^{-1} X \, P(\Phi(X))^T R^* P(\boldsymbol{\varphi}). \tag{4.9}$$

From Eq. (2.12) in kernel PCA 2.1, the projection $P(\Phi(X))$ is defined by

$$\begin{aligned} P(\Phi(X)) &= (P(\Phi(\boldsymbol{x}_1)), \ldots, P(\Phi(\boldsymbol{x}_N))) \\ &= ((k_{\boldsymbol{x}_1}\boldsymbol{\alpha})^T, \ldots, k_{\boldsymbol{x}_N}\boldsymbol{\alpha})^T) \\ &= \boldsymbol{\alpha}^T K. \end{aligned} \tag{4.10}$$

and $P(\boldsymbol{\varphi})$ is determined by

$$\begin{aligned} P(\boldsymbol{\varphi}) &= (\langle \boldsymbol{\varphi}, \boldsymbol{u}_1 \rangle, \ldots, \langle \boldsymbol{\varphi}, \boldsymbol{u}_p \rangle)^T \\ &= (\langle \boldsymbol{\varphi}, \Phi(X)\boldsymbol{\alpha}_1 \rangle, \ldots, \langle \boldsymbol{\varphi}, \Phi(X)\boldsymbol{\alpha}_p \rangle)^T \\ &= (\langle \Phi(X)\boldsymbol{\gamma}, \Phi(X)\boldsymbol{\alpha}_1 \rangle, \ldots, \langle \Phi(X)\boldsymbol{\gamma}, \Phi(X)\boldsymbol{\alpha}_p \rangle)^T \\ &= (\boldsymbol{\gamma} K \boldsymbol{\alpha})^T \\ &= \boldsymbol{\alpha}^T K \boldsymbol{\gamma}. \end{aligned} \tag{4.11}$$

with $\boldsymbol{\alpha}$ defined in Eq. (2.12). Substituting Eq. (4.11) into Eq. (4.9), we have:

$$\boldsymbol{x}^* = (XX^T)^{-1} X \, P(\Phi(X))^T R^* \boldsymbol{\alpha}^T K \boldsymbol{\gamma}. \tag{4.12}$$

One can easily include some regularisation terms in the optimisation problems (4.7) and (4.8), which can be easily propagated to the pre-image expression. For example, in the case of non-invertible XX^T, a regularisation term is introduced in Eq. (4.12) as:

$$\boldsymbol{x}^* \quad = \quad (XX^T + \lambda I_d)^{-1} X \, P(\Phi(X))^T R^* \boldsymbol{\alpha}^T K \boldsymbol{\gamma}, \tag{4.13}$$

for some positive regularisation parameter λ.

4.2.2 Learning nonlinear transformation for pre-image estimation

In the following, we propose to extend the above result to learn nonlinear transformations for pre-image estimation. Let $\widehat{\kappa}$ be a kernel defined on the feature space $\mathcal{H}$, and $\widehat{\Phi}$ the corresponding embedding function that maps any element of $\mathcal{H}$ into

the Hilbert space defined by $\widehat{\kappa}$. With some abuse of notation, we denote $\widehat{\Phi}(\Phi(X))$ the matrix of all mapped elements $\widehat{\Phi}(\Phi(\boldsymbol{x}_i))$, for $i = 1, ..., N$. Let $\widehat{K}$ be the Gram matrix of general term $\widehat{\kappa}(\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j))$.

The pre-image estimation problem can be then defined as learning a nonlinear transformation that defines an isometry between X and $\widehat{\Phi}(\Phi(X))$ as:

$$R^* = \arg\min_R \|X^T X - \widehat{\Phi}(\Phi(X))^T \, R \, \widehat{\Phi}(\Phi(X))\|_{\mathcal{F}}^2. \tag{4.14}$$

By using kernel PCA, Eq. (4.15) can be rewritten by:

$$R^* = \arg\min_R \|X^T X - P(\widehat{\Phi}(\Phi(X)))^T \, R \, P(\widehat{\Phi}(\Phi(X)))\|_{\mathcal{F}}^2. \tag{4.15}$$

Similarly, a closed-form solution for R^* can be obtained as:

$$\begin{aligned} R^* &= (P(\widehat{\Phi}(\Phi(X))) \, P(\widehat{\Phi}(\Phi(X)))^T)^{-1} P(\widehat{\Phi}(\Phi(X))) \\ &\quad X^T X P(\widehat{\Phi}(\Phi(X)))^T (P(\widehat{\Phi}(\Phi(X))) P(\widehat{\Phi}(\Phi(X)))^T)^{-1}, \end{aligned} \tag{4.16}$$

and

$$P(\widehat{\Phi}(\Phi(X))) = \widehat{\boldsymbol{\alpha}}^T \, \widehat{K}. \tag{4.17}$$

where $\widehat{\boldsymbol{\alpha}}$ is a matrix of the eigenvectors of $\widehat{K}$. To estimate $\widehat{K}$, an indirect manner is to use a kernel PCA, with $\widehat{\kappa}(\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)) \approx \widehat{\kappa}(P(\Phi(\boldsymbol{x}_i)), P(\Phi(\boldsymbol{x}_j)))$. A simpler way is possible when dealing with kernels that are radial basis functions. For example, for the well-known Gaussian kernel $\widehat{\kappa}$, $\widehat{K}$ is estimated directly from K as:

$$\begin{aligned} \widehat{\kappa}(\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)) &= \exp\left(-\frac{\| \Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j) \|^2}{2\sigma^2}\right), \\ &= \exp\left(-\frac{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_i) \rangle - 2\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle + \langle \Phi(\boldsymbol{x}_j), \Phi(\boldsymbol{x}_j) \rangle}{2\sigma^2}\right), \\ &= \exp\left(-\frac{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j)}{2\sigma^2}\right). \end{aligned} \tag{4.18}$$

The estimation of the pre-image of $\boldsymbol{\varphi} = \sum_{i=1}^{N} \gamma_i \, \Phi(\boldsymbol{x}_i)$ is then given by the time series $\boldsymbol{x}^*$:

$$\boldsymbol{x}^* = (X X^T)^{-1} \, X \, P(\widehat{\Phi}(\Phi(X)))^T R^* P(\widehat{\Phi}(\boldsymbol{\varphi})), \tag{4.19}$$

with $P(\widehat{\Phi}(\boldsymbol{\varphi})) = (\widehat{\boldsymbol{k}}_{\boldsymbol{\varphi}}\widehat{\boldsymbol{\alpha}})^T$, where $\widehat{\boldsymbol{k}}_{\boldsymbol{\varphi}}$ is the vector whose i-th entry is

$$\begin{aligned}
\widehat{\kappa}(\boldsymbol{\varphi}, \Phi(\boldsymbol{x}_i)) &= \exp\left(-\frac{\| \boldsymbol{\varphi} - \Phi(\boldsymbol{x}_i) \|^2}{2\sigma^2}\right), \\
&= \exp\left(-\frac{\langle \boldsymbol{\varphi}, \boldsymbol{\varphi} \rangle - 2\langle \varphi, \Phi(\boldsymbol{x}_i) \rangle + \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_i) \rangle}{2\sigma^2}\right), \\
&= \exp\left(-\frac{\langle \Phi(X)\boldsymbol{\gamma}, \Phi(X)\boldsymbol{\gamma} \rangle - 2\langle \Phi(X)\boldsymbol{\gamma}, \Phi(\boldsymbol{x}_i) \rangle + \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_i) \rangle}{2\sigma^2}\right), \\
&= \exp\left(-\frac{\boldsymbol{\gamma}^T K \boldsymbol{\gamma} - 2\,\boldsymbol{\gamma}^T \boldsymbol{k}_{\boldsymbol{x}_i}^T + K_{ii}}{2\sigma^2}\right). \qquad (4.20)
\end{aligned}$$

The above proposed formulations and results for pre-image estimation (Section 4.2.1) present some similarities and differences with the method proposed in [48] and presented in Section 3.4. First of all, both approaches propose formulations and solutions that only require linear algebra and are independent of the type of kernel. To establish the isometry, in [48] a linear transformation restricted to the form $R = \Phi(X)AA^T\Phi(X)^T$ is estimated, whereas in our proposal the estimated R may be linear Eq.(4.6) or non linear Eq.(4.15) and is importantly unconstrained, namely of general form which enlarges its potential to deal with complex structures. Finally, while in [48] the solution Eq.(3.28) involves the kernel information through the regularisation term, which may be canceled for lower values of λ, in the proposed solutions Eq.(4.13) and Eq.(4.19) the kernel information is entirely considered regardless of the regularisation specifications.

4.3 Pre-image estimation for time series kernel analytics

In the previous Section 4.2.1, data are assumed static or be a set of time series of the same length and thus lying in a metric space. In this Section we consider $X = \{\boldsymbol{x}_i\}_{i=1}^N$ as instead composed of time series $\boldsymbol{x}_i$ of different lengths t_i that are located in a non-metric space, rendering the previous results as well as the

pre-image estimation related works not applicable.

To address the pre-image estimation for such challenging time series, we define an embedding function that allows to represent the time series in a metric space, where the previous linear and nonlinear transformations method for pre-image estimation can be performed conveniently. Before that, we introduce the concept of time series alignment and give the definition of some temporal kernels.

4.3.1 Time series alignment

Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be two time series. To resorb the delays arising in time series, a temporal alignment between each $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is performed by dynamic programming. An alignment $\boldsymbol{\pi}$ of length $|\boldsymbol{\pi}| = m$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as the set of m increasing couples

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

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

$$1 = \pi_1(1) \leq \pi_1(2) \leq ... \leq \pi_1(m) = t_i,$$
$$1 = \pi_2(1) \leq \pi_2(2) \leq ... \leq \pi_2(m) = t_j,$$

and $\forall\, l \in \{1, ..., m\}$, $\pi_1(l+1) \leq \pi_1(l)+1$ 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 $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ describes a way to associate each element of $\boldsymbol{x}_i$ to one or more elements of $\boldsymbol{x}_j$ and vice-versa. Such an alignment can be conveniently represented by a path in the $t_i \times t_j$ grid, as shown in Figure 4.1 (left), where the above monotonicity conditions ensure that the path is neither going back nor jumping. The optimal alignment $\boldsymbol{\pi}^*$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is then

obtained as:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \|\boldsymbol{x}_i^{\boldsymbol{\pi}_1} - \boldsymbol{x}_r^{\boldsymbol{\pi}_2}\|^2. \tag{4.21}$$

where $\boldsymbol{x}_i^{\boldsymbol{\pi}_1} = (\boldsymbol{x}_{i\,\pi_1(1)}, ..., \boldsymbol{x}_{i\,\pi_1(m)})$ and $\boldsymbol{x}_j^{\boldsymbol{\pi}_2} = (\boldsymbol{x}_{j\,\pi_2(1)}, ..., \boldsymbol{x}_{j\,\pi_2(m)})$ are $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ aligned through $\boldsymbol{\pi}$.

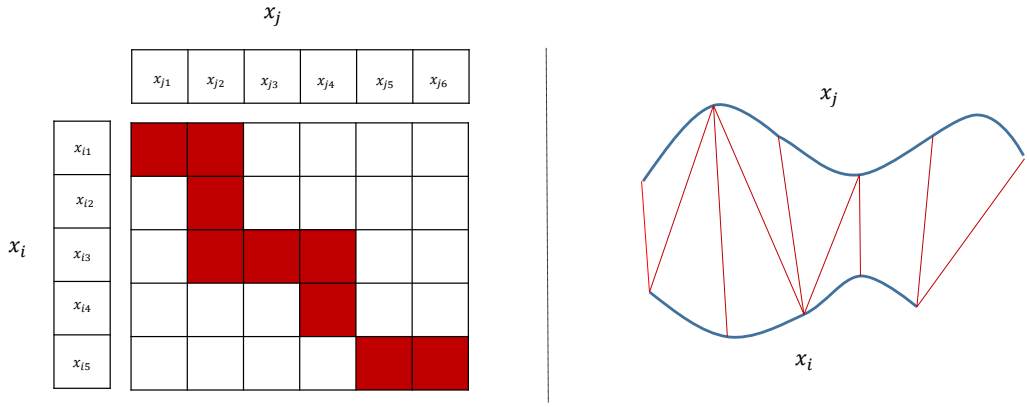


Figure 4.1: In the left, the temporal alignment between $\boldsymbol{x}_i$ ($t_i = 5$) and $\boldsymbol{x}_j$ ($t_j = 6$), the optimal alignment $\boldsymbol{\pi}^*$ is indicated in red. In the right, the optimal alignment is illustrated as connections between two time series.

4.3.2 Proximity measure between time series

Dynamic time warping [7] is a well-known dissimilarity measure on time series that capture temporal distortions. Based on the optimal alignment $\boldsymbol{\pi}^*$, the Dynamic time warping (DTW) between two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined by

$$\text{DTW}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i^{\boldsymbol{\pi}_1^*} - \boldsymbol{x}_j^{\boldsymbol{\pi}_2^*}\|^2,$$

where $\boldsymbol{\pi}^*$ is determined in Eq. (4.21).

However, DTW is not a metric as not satisfy the triangle inequality:

$$\text{DTW}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \text{DTW}(\boldsymbol{x}_i, \boldsymbol{x}_k) \ngeq \text{DTW}(\boldsymbol{x}_k, \boldsymbol{x}_j).$$

The dynamic programming approach [7] is used to find the optimal alignment as well as the minimum distance DTW between two time series. The complexity of

the dynamic time warping is $O(|\boldsymbol{x}_i| \times |\boldsymbol{x}_j|)$, where $|\boldsymbol{x}_i|, |\boldsymbol{x}_i|$ are respectively lengths of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Constraints are widely used to speed up dynamic time warping programming. Two well-known global constraint region are the Sakoe-Chiba band [4] and Itakura parallelogram [6], shown in Figure 4.2. The region of optimal alignment is selected only from respective shaded region.

The Sakoe-Chiba band runs symmetrically along the diagonal and has a width

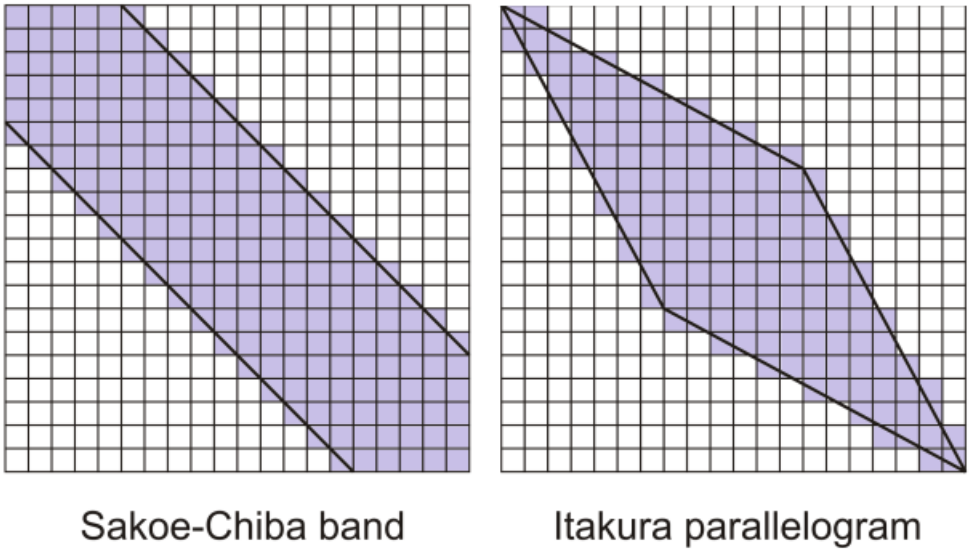


Figure 4.2: Illustration of the DTW constraints

of $T \in \mathbb{N}$. This constraint implies that an element at time t of $|\boldsymbol{x}_i|$ can be aligned only to one element at time t' of $\boldsymbol{x}_j$ such that

$$t' \in \left[\frac{|\boldsymbol{x}_i| - T}{|\boldsymbol{x}_j| - T}(t - T), \frac{|\boldsymbol{x}_i| - T}{|\boldsymbol{x}_j| - T}(t + T)\right] \cap \left[1, \ldots, |\boldsymbol{x}_i|\right].$$

The Itakura parallelogram describes a region that constrains the slope of a warping path. Given any slope $S \in \{\mathbb{R} > 1\}$, the domain of Itakura parallelogram lies between two warping paths with the slopes of the values $1/S$ and S. These constraints significantly speed up the computation of DTW as well as any measure based on the optimal time warping alignment. For instance, in case of a Sakoe-Chiba of a band T, the complexity of computation is only $O(T \times \max(|\boldsymbol{x}_i|), |\boldsymbol{x}_j|)$ instead of $O(|\boldsymbol{x}_i| \times |\boldsymbol{x}_j|)$ required in standard DTW.

For temporal data, several kernels under time warping that are proposed in the last years allow to apply kernel methods for time series. First of all is the Gaussian dynamic time warping kernel (KDTW) [3] defined by

$$\text{KDTW}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\Big(-\frac{1}{\sigma}\text{DTW}(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big),$$

where σ is a normalisation parameter. In general, KDTW is not positive definite kernel that allows to embed data into Hilbert feature space. However, this kernel can procedure good results in some cases [30], [52].

The Dynamic time alignment kernel (DTAK) proposed in [28] adjust another similarity or kernel between two time series by finding the optimal alignment to maximise the accumulated similarity between two time series:

$$\text{DTAK}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max_{\boldsymbol{\pi}} \frac{1}{|\boldsymbol{\pi}|} \sum_{(t,t')\in\boldsymbol{\pi}} s(\boldsymbol{x}_{it}, \boldsymbol{x}_{jt'})$$

where in particular $s(\boldsymbol{x}_{it}, \boldsymbol{x}_{jt'}) = \exp\Big(-\frac{1}{\sigma^2}\|\boldsymbol{x}_{it} - \boldsymbol{x}_{jt'}\|^2\Big)$, and in general $s(.,.)$ is any similarity measure on $\mathbb{R}^d$. DTAK is a symmetric kernel function, however, it may be not a positive definite kernel. Note that, in practice, several ad-hoc methods that perturb the whole diagonal by the absolute of the smallest eigenvalue are used to ensure the positive definiteness of Gram matrix of DTAK .

Global alignment kernel (KGA) [36] is not based on the optimal alignment, but takes advantage of all accumulated similarity by all possible alignment, defined by:

$$\text{KGA}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{\pi}} \prod_{(t,t')\in\boldsymbol{\pi}} \kappa(\boldsymbol{x}_{it}, \boldsymbol{x}_{jt'})$$

where

$$\kappa(\boldsymbol{x}_{it}, \boldsymbol{x}_{jt'}) = \exp\Big(-\lambda(\frac{1}{2\sigma^2}\|\boldsymbol{x}_{it} - \boldsymbol{x}_{jt'}\|^2 + \log(2 - e^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}_{it} - \boldsymbol{x}_{jt'}\|^2}))\Big).$$

KGA that is positive definite kernel under mild condition, do a better job of quantifying all similarities coherently, because it consider all possible alignments. Global

alignment kernel have been obtain success in different application fields [47], [67] and shown to be competitive to other kernels. However, similar to KDTW and DTAK, KGA has quadratic complexity of computation $O(|\boldsymbol{x}_i| \times |\boldsymbol{x}_j|)$.

4.3.3 Pre-image estimation for time series analytics

Let us consider now that $X = \{\boldsymbol{x}_i\}_{i=1}^N$ is a set of N time series, where each $\boldsymbol{x}_i \in \mathbb{R}^{d \times t_i}$ is a multivariate time series that may have different length t_i and involve varying delays. Let $\Phi(\boldsymbol{x}_i)$ be the Φ-mapping of the time series $\boldsymbol{x}_i$ into the Hilbert space $\mathcal{H}$ related to a temporal kernel $\kappa(.,.)$ that involves dynamic time alignments such as DTAK [28], KDTW [2], KGA [36]. Given $\boldsymbol{\varphi} = \sum_{i=1}^N \gamma_i \ \Phi(\boldsymbol{x}_i)$ a result generated in $\mathcal{H}$, the objective is to estimate the time series $\boldsymbol{x}^* \in \mathbb{R}^{d \times t^*}$ that is the pre-image of $\boldsymbol{\varphi}$. This problem is particularly challenging since, under varying delays, the time series are not longer lying into a metric space, which makes inapplicable the related work pre-image estimation approaches. Note that, if the time series are assumed of the same length and lying in a metric space, then the proposed method in Section 4.2 can be applied.

To address the pre-image estimation for such challenging time series, we propose an embedding function that allows to represent time series into a metric space, where the previous linear and nonlinear pre-image estimation can be performed conveniently.

For this purpose, first we define $\mathcal{N}_{\boldsymbol{\varphi}}$ in $\mathcal{H}$ and $\mathcal{N}_{\boldsymbol{\varphi}}^{-1}$ as the set of the n-closest neighbours of $\boldsymbol{\varphi}$ and its pre-image, given as:

$$\mathcal{N}_{\boldsymbol{\varphi}} = \left\{ \Phi(\boldsymbol{x}_i) \middle| \langle \Phi(\boldsymbol{x}_i), \boldsymbol{\varphi} \rangle = \sum_{j=1}^{N} \gamma_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ is among the } n \text{ highest values} \right\} \tag{4.22}$$

$$\mathcal{N}_{\boldsymbol{\varphi}}^{-1} = \left\{ \boldsymbol{x}_i \mid \Phi(\boldsymbol{x}_i) \in \mathcal{N}_{\boldsymbol{\varphi}} \right\}. \tag{4.23}$$

Let $\Phi(\boldsymbol{x}_r)$ be the representative of $\mathcal{N}_{\boldsymbol{\varphi}}$ with $\boldsymbol{x}_r \in \mathbb{R}^{d \times t^*}$ defined as:

$$\begin{aligned} \Phi(\boldsymbol{x}_r) &= \underset{\Phi(\boldsymbol{x}_i) \in \mathcal{N}_{\boldsymbol{\varphi}}}{\arg\max} \sum_{\Phi(\boldsymbol{x}_j) \in \mathcal{N}_{\boldsymbol{\varphi}}} \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle \\ &= \underset{\Phi(\boldsymbol{x}_i) \in \mathcal{N}_{\boldsymbol{\varphi}}}{\arg\max} \sum_{\Phi(\boldsymbol{x}_j) \in \mathcal{N}_{\boldsymbol{\varphi}}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{4.24}$$

We define f_r, the temporal embedding function, that allows to embed time series

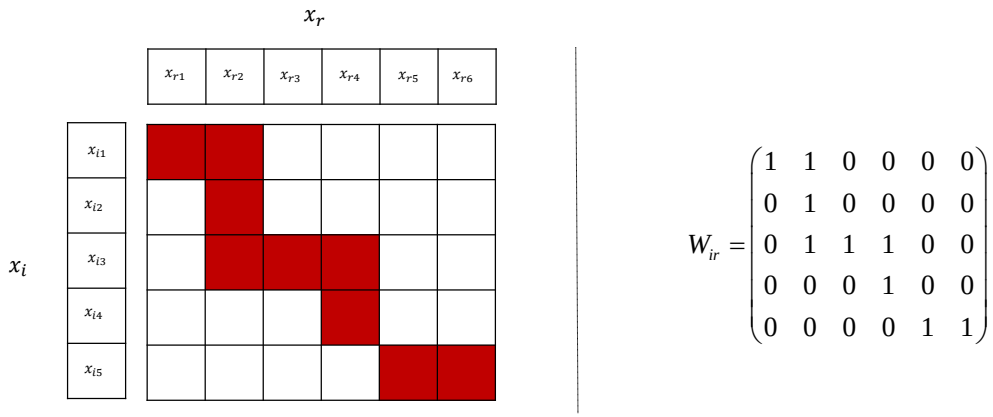


Figure 4.3: In the left, the temporal alignment between $\boldsymbol{x}_i$ ($t_i = 5$) and $\boldsymbol{x}_r$ ($t^* = 6$), the optimal alignment $\boldsymbol{\pi}^*$ is indicated in red. In the right, the adjacency binary matrix related to the optimal temporal alignment.

$\boldsymbol{x}_i \in \mathbb{R}^{d \times t_i}$ into a new temporal metric space as:

$$\begin{aligned} f_r : \quad X &\longrightarrow \widetilde{X} \subset \widetilde{\mathcal{I}} = \mathbb{R}^{d \times t^*} \\ \boldsymbol{x}_i &\longrightarrow f_r(\boldsymbol{x}_i) = \boldsymbol{x}_i \, W_{ir} \, N_{ir} \end{aligned} \tag{4.25}$$

where $W_{ir} \in \{0,1\}^{t_i \times t^*}$ is the binary matrix related to the optimal temporal alignement between $\boldsymbol{x}_i$ and $\boldsymbol{x}_r$, as shown in Figure 4.3 (right). The matrix $N_{ir} = diag(W_{ir}^T \, \mathbf{1}_{t_i})^{-1}$ is the weight diagonal matrix of order t^*, of general term $\frac{1}{|N_t|}$, that gives the weight of the element t of $\boldsymbol{x}_r$, where $|N_t|$ is the number of time stamps of $\boldsymbol{x}_i$ aligned to t. In particular, note that $\mathbf{x}_r$ remains unchanged by f_r, as $W_{rr} = N_{rr} = diag([1, 1, \ldots, 1])$. The set of embedded time series $\widetilde{X} = \{f_r(\boldsymbol{x}_1), ..., f_r(\boldsymbol{x}_N)\}$ is for now lying in a metric space $\widetilde{\mathcal{I}}$, where the delays are resorbed w.r.t. the representative time series $\boldsymbol{x}_r$. The pre-image solution

provided in the method described in Section 4.1 can be developed to establish a linear or nonlinear isometry between $\widetilde{X}$ and $\Phi(X)$.

5

Experimental results

In this section, we evaluate the efficiency of the proposed pre-image estimation method under three major time series analysis tasks: 1) time series averaging, 2) time series reconstruction and denoising and 3) time series representation learning. The proposed pre-image estimation method TsPrima is compared to three major alternative approaches introduced in Chapter 3 as Honeine in Section 3.4, [48], Kwok in Section 3.3, [55], and Bakir in Section 3.5, [51] methods. The experiments are conducted on 33 public datasets (Table 5.1) including univariate and multivariate time series data, that may involve varying delays and be of the same or different lengths. The 25 first datasets in Table 5.1 are selected from the archive given in [12, 20] by using three selection criteria: a) have a reasonable number of classes (Nb. of Classes < 50), b) have a sufficient size for train and test samples (Train size $<= 500$ and Test size < 3000), c) avoid time series of

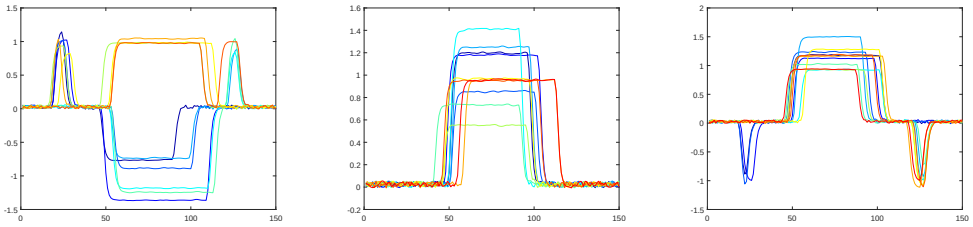
extra large lengths (Time series length < 700). To obtain a manageable number of datasets, the 3 above selection criteria are applied on the top 40 datasets, in the order set out in [65]. The 25 obtained datasets are composed of univariate time series and half of the datasets include significant delays. We consider a dataset as including significant delays if the difference between the 1-NN Euclidean distance error and the 1-NN Dynamic time warping [19] error is greater than 5%. The 5 next datasets include univariate and multivari- ate time series covering local and noisy salient events as described in [11, 17, 15] and the three last datasets are related to handwritten digits and characters, they are described as multivariate time series of variable lengths [25]. In the following, Section 5.1 gives the data description and Section 5.2 details the validation and evaluation process. Finally, we discuss the obtained results in Section 5.7.

5.1 Data description

The experiments are conducted on three groups of datasets. The first group is composed of the 25 datasets from UCR [12] that is the most commonly used data for time series analytics. Each dataset is a collection of univariate time series of the same length and pre-divided into training and test sets. In the second group, we consider BME, UMD, POWERCONS and SPIRAL datasets, where time series share local temporal features within the classes while being of distinctive global behaviour, and include huge noise (Figure 5.4). For example, BME (Figure 5.2) includes two challenging classes BEGIN and END characterised by a small bell arising at the initial and final periods respectively. The overall behaviour may be different depending on whether the large bell is pointing upward or downward. UMD (Figure 5.1) introduces more complexity with the classes UP and DOWN characterised by a small bell that may occur at different time stamps. SPIRAL1 (Figure 5.3) and SPIRAL2 (Figure 5.4) share a latent 3-D time series that may appear randomly at different time stamps, in particular of SPIRAL2 time series involve high level of noise at the initial and final periods.

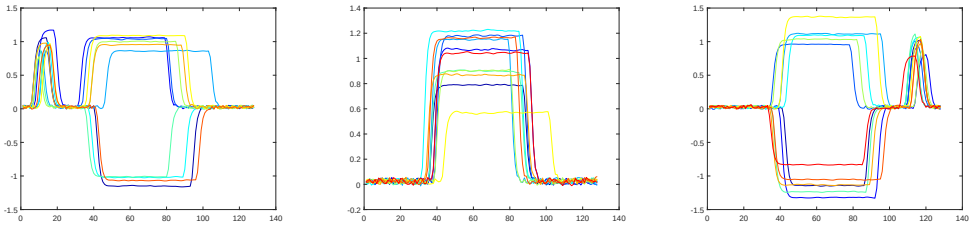
Table 5.1: Data Description

Dataset	Nb. class	Train size	Test size	Time series length	Univariate
CC	6	300	300	60	✓
GUNPOINT	2	50	150	150	✓
CBF	3	30	900	128	✓
OSULEAF	6	200	242	427	✓
SWEDISHLEAF	15	500	625	128	✓
TRACE	4	100	100	275	✓
FACEFOUR	4	24	88	350	✓
LIGHTING2	2	60	61	637	✓
LIGHTING7	7	70	73	319	✓
ECG200	2	100	100	96	✓
ADIAC	37	390	391	176	✓
FISH	7	175	175	463	✓
BEEF	5	30	30	470	✓
COFFEE	2	28	28	286	✓
OLIVEOIL	4	30	30	570	✓
DIATOMSIZER	4	16	306	345	✓
ECG5DAYS	2	23	861	136	✓
FACESUCR	14	200	2050	131	✓
ITALYPOWERD	2	67	1029	24	✓
MEDICALIMAGES	10	381	760	99	✓
MOTESTRAIN	2	20	1252	84	✓
SONYAIBOII	2	27	953	65	✓
SONYAIBO	2	20	601	70	✓
SYMBOLS	6	25	995	398	✓
TWOLEADECG	2	23	1139	82	✓
SPIRAL1	1	50	50	101	✗
SPIRAL2	1	50	50	300	✗
POWERCONS	2	73	292	144	✓
BME	3	30	150	128	✓
UMD	3	36	144	150	✓
DIGITS	10	100	100	29∼218	✗
LOWER	26	130	260	27∼163	✗
UPPER	26	130	260	27∼412	✗



UP MIDDLE DOWN

Figure 5.1: Time series of UMD dataset with classes: UP, MIDDLE, DOWN



BEGIN MIDDLE END

Figure 5.2: Time series of BME dataset with classes: BEGIN, MIDDLE, END.

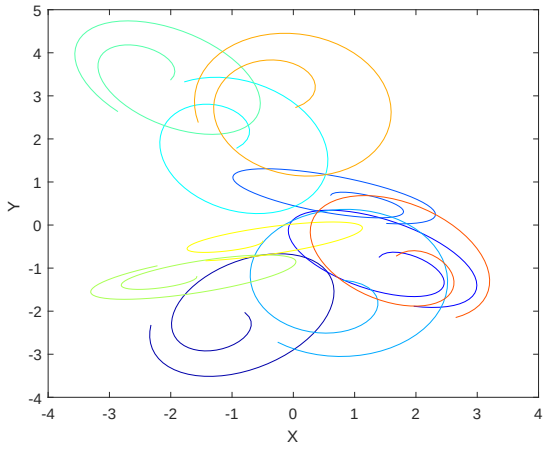


Figure 5.3: Time series of SPIRAL1 dataset.

The third group composed of datasets DIGITS, UPPER and LOWER consists of 6-D motion gesture database (6DMG) where time series are multivariate and involved varying delays. These datasets (Figure 5.5) give the description of the 2-D Air-handwriting motion gesture of digits, upper and lower case letter performed on a Nintendo device by different writers [25].

In summary, Table 5.1 indicates for each dataset: the number of classes (Nb. Class), the size of training set (Train size), the size of test set (Test set), the time

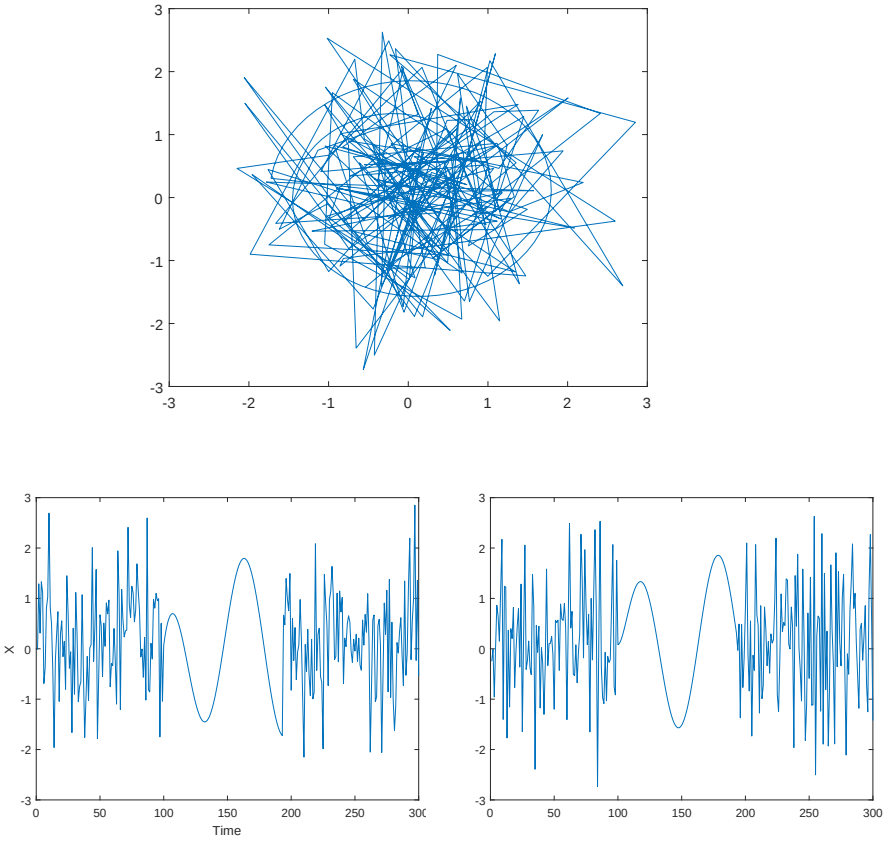


Figure 5.4: Time series of SPIRAL2 dataset.

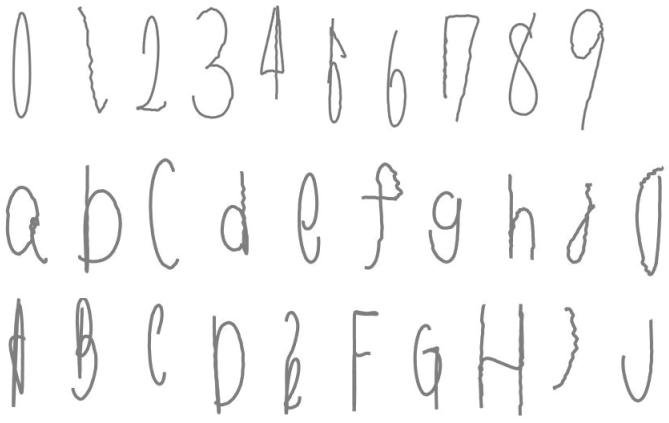


Figure 5.5: 6DMG Air-Handwriting dataset with classes: DIGITS, UPPER, LOWER

series length (Time series length), and the type of time series (Univariate).

5.2 Validation process

In this section, we evaluate the proposed method in the context of averaging, reconstruction and denoising, and representation learning. We conduct the four methods: Honeine, Kwok, Bakir and the proposed method TsPrima on the datasets (shown in Section 5.1). For the comparison, we rely on the standard DTAK which measures the similarity between the obtained time series and the truth ones in denoising and reconstruction, representation, or the inertia of the obtained centroid in averaging, to evaluate each method. The higher the index, the better the agreement is. For each method, the related parameters indicated in Table 5.2 are learned by a grid search on validation set, the best parameters are then used to perform these tasks on the evaluation set. The process is iterated over 10 runs and the averaged performances are reported in Table 5.5, 5.4, and 5.6.

Table 5.2: The descriptions of parameters

Methods	Parameters	Range of values	Description
All	σ^t	$\{0.2, 0.5, 1, 2, 5, 10\} * med(DTW(\boldsymbol{x}, \boldsymbol{y}))$	width of DTAK
All	T	$[0, 100]$ lag of 10	Sakoe-Chiba band
TsPrima, Bakir	σ	$\{0.2, 0.5, 1, 2, 5, 10\}$	width of Gaussian kernel
Honeine	λ	10^{-9}	regulary term
TsPrima, Kwok	n	$[2, round(\sqrt{N})]$ lag of 2	number of neighbors

5.3 Time series averaging by pre-image estimation

Estimating the centroid of a set of time series is a major topic for many time series analytics as summarisation, prototype extraction or clustering. Time series averaging has been an active area in the last decade, where the propositions mainly focus on tackling the tricky problem of multiple temporal alignments [15, 16, 17]. A suitable way to circumvent the problem of multiple temporal alignments is to

use a temporal kernel method to evaluate the time series centroid in the feature space. The pre-image of the centroid is then estimated to obtain the time series averaging in the input space.

In that context, let $X = \{\boldsymbol{x}_i\}_{i=1}^N$ and $\Phi(X) = \{\Phi(\boldsymbol{x}_i)\}_{i=1}^N$ be, respectively, a set of time series and their mapped images into the Hilbert space $\mathcal{H}$ related to the temporal kernel DTAK [22]. The centroid of X with respect to DTAK is defined by

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{y}\in\mathcal{X}} \sum_{i=1}^{N} \text{DTAK}(\boldsymbol{y}, \boldsymbol{x}_i).$$

As $\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle = \text{DTAK}(\boldsymbol{z}, \boldsymbol{z}) = 1$ for any time series $\boldsymbol{z}$, we have

$$\begin{aligned}
\boldsymbol{x}^* &= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{N} \text{DTAK}(\boldsymbol{y}, \boldsymbol{x}_i) \\
&= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{N} \langle \Phi(\boldsymbol{y}), \Phi(\boldsymbol{x}_i) \rangle \\
&= \arg\min_{\boldsymbol{y}} \sum_{i=1}^{N} 1 - 2\langle \Phi(\boldsymbol{y}), \Phi(\boldsymbol{x}_i) \rangle + 1 \\
&= \arg\min_{\boldsymbol{y}} \sum_{i=1}^{N} \langle \Phi(\boldsymbol{y}), \Phi(\boldsymbol{y}) \rangle - 2\langle \Phi(\boldsymbol{y}), \Phi(\boldsymbol{x}_i) \rangle + \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_i) \rangle \\
&= \arg\min_{\boldsymbol{y}} \sum_{i=1}^{N} \| \Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x}_i) \|^2 \\
&\Rightarrow \Phi(\boldsymbol{x}^*) = \frac{1}{N} \sum_{i=1}^{N} \Phi(\boldsymbol{x}_i)
\end{aligned}$$

Hence, let $\boldsymbol{\varphi} = \frac{1}{N}\sum_{i=1}^{N} \Phi(\boldsymbol{x}_i)$ be the centroid of the mapped time series in the feature space and $\boldsymbol{x}^*$ its pre-image in the input space. The quality of the obtained centroids is given by the within inertia $\sum_i \text{DTAK}(\boldsymbol{x}^*, \boldsymbol{x}_i)$; the higher the within inertia, the better is the estimated centroid.

To evaluate the efficiency of each pre-image estimation method, the time series centroid is estimated for each class of the studied datasets and the induced within-class inertia is evaluated. The average within-class inertia is then reported in Table 5.3 for each dataset and each pre-image estimation method; the best values are indicated in bold (t-test at 5% risk). In addition, a Nemenyi test [8] is performed to

compare the significance of the obtained results, with the related critical difference diagram given in Figure 5.6. The estimated time series centroids for some challenging classes are shown in Figure 5.7, where we retain particularly SPIRAL1 and the handwritten digits and characters datasets (DIGITS, LOWER and UPPER) as they are more intuitive to visually evaluate the quality of the estimated time series centroids.

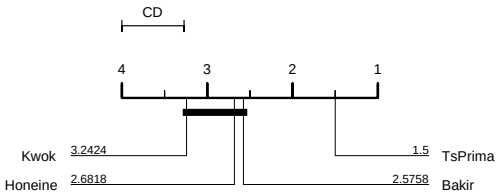


Figure 5.6: Nemenyi test: comparison of pre-image methods under centroid estimation task

5.4 Time series reconstruction and denoising by pre-image estimation

The reconstruction and denoising tasks represent a standard application context for pre-image estimation. For time series reconstruction task, a kernel PCA is performed on the training set, the reconstruction of a given test sample $\boldsymbol{x}$ is then defined as the pre-image $\boldsymbol{x}^*$ of its kernel PCA projection $P(\Phi(\boldsymbol{x}))$. The latter takes the form $\boldsymbol{\varphi} = \Phi(X)\boldsymbol{\gamma}$, with $\boldsymbol{\gamma}$ defined as:

$$\boldsymbol{\gamma} = (I_N - \mathbb{1}_N)\boldsymbol{\alpha}\,\boldsymbol{\alpha}^T\,\widetilde{\boldsymbol{k}}_{\boldsymbol{x}}^T + \frac{1}{N}\mathbf{1}_N \tag{5.1}$$

The quality of the reconstruction is then measured as the similarity $\text{DTAK}(\boldsymbol{x}^*, \boldsymbol{x})$ between each test sample $\boldsymbol{x}$ and its reconstruction $\boldsymbol{x}^*$; the higher the criterion, the better is the reconstruction. Table 5.4 gives the average quality of reconstruction obtained for each dataset and each method. Figure 5.8 gives the critical difference diagram related to the Nemenyi test for the average ranking comparison of the studied methods. Figure 5.9 shows the reconstructions obtained for some challenging time series of DIGITS, LOWER and UPPER datasets.

Table 5.3: Average within-class inertia of the estimated time series centroids

Dataset	TsPrima	Honeine	Kwok	Bakir
CC	**0.744**	0.709	0.721	0.709
GUNPOINT	0.902	**0.910**	0.882	0.886
CBF	**0.798**	0.737	0.755	0.737
OSULEAF	0.985	**0.987**	0.986	**0.987**
SWEDISHLEAF	0.910	**0.920**	**0.920**	**0.92**
TRACE	**0.998**	0.992	0.991	0.992
FACEFOUR	**0.981**	0.980	**0.981**	0.98
LIGHTING2	**0.918**	0.876	0.859	0.875
LIGHTING7	**0.964**	0.930	0.930	0.931
ECG200	**0.593**	0.565	0.567	0.566
ADIAC	**0.997**	**0.997**	0.996	**0.997**
FISH	**0.996**	0.995	0.994	0.995
BEEF	**0.900**	0.892	0.898	0.89
COFFEE	**0.998**	**0.998**	**0.998**	**0.998**
OLIVEOIL	**0.999**	**0.999**	0.998	**0.999**
DIATOMSIZER	**0.997**	**0.997**	**0.997**	**0.997**
ECG5DAYS	**0.777**	0.746	0.417	0.746
FACESUCR	**0.721**	0.699	0.648	0.700
ITALYPOWERD	**0.610**	0.552	0.420	0.542
MEDICALIMAGES	**0.671**	0.644	0.637	0.646
MOTESTRAIN	0.776	**0.777**	0.701	**0.777**
SONYAIBOII	**0.749**	0.740	0.716	0.740
SONYAIBO	0.960	**0.962**	0.955	**0.962**
SYMBOLS	**0.959**	0.949	0.904	0.951
TWOLEADECG	**0.980**	0.977	0.911	0.977
SPIRAL1	**0.831**	0.823	0.799	0.824
SPIRAL2	**0.947**	0.940	0.934	0.940
POWERCONS	**0.458**	0.328	0.436	0.33
BME	**0.701**	0.572	0.638	0.555
UMD	**0.800**	0.765	0.724	0.755
DIGITS	**0.746**	0.575	0.657	0.581
LOWER	**0.713**	0.544	0.645	0.545
UPPER	**0.764**	0.572	0.57	0.573
Nb. Best	**28**	9	4	8
Avg. Rank	**1.50**	2.68	3.24	2.58

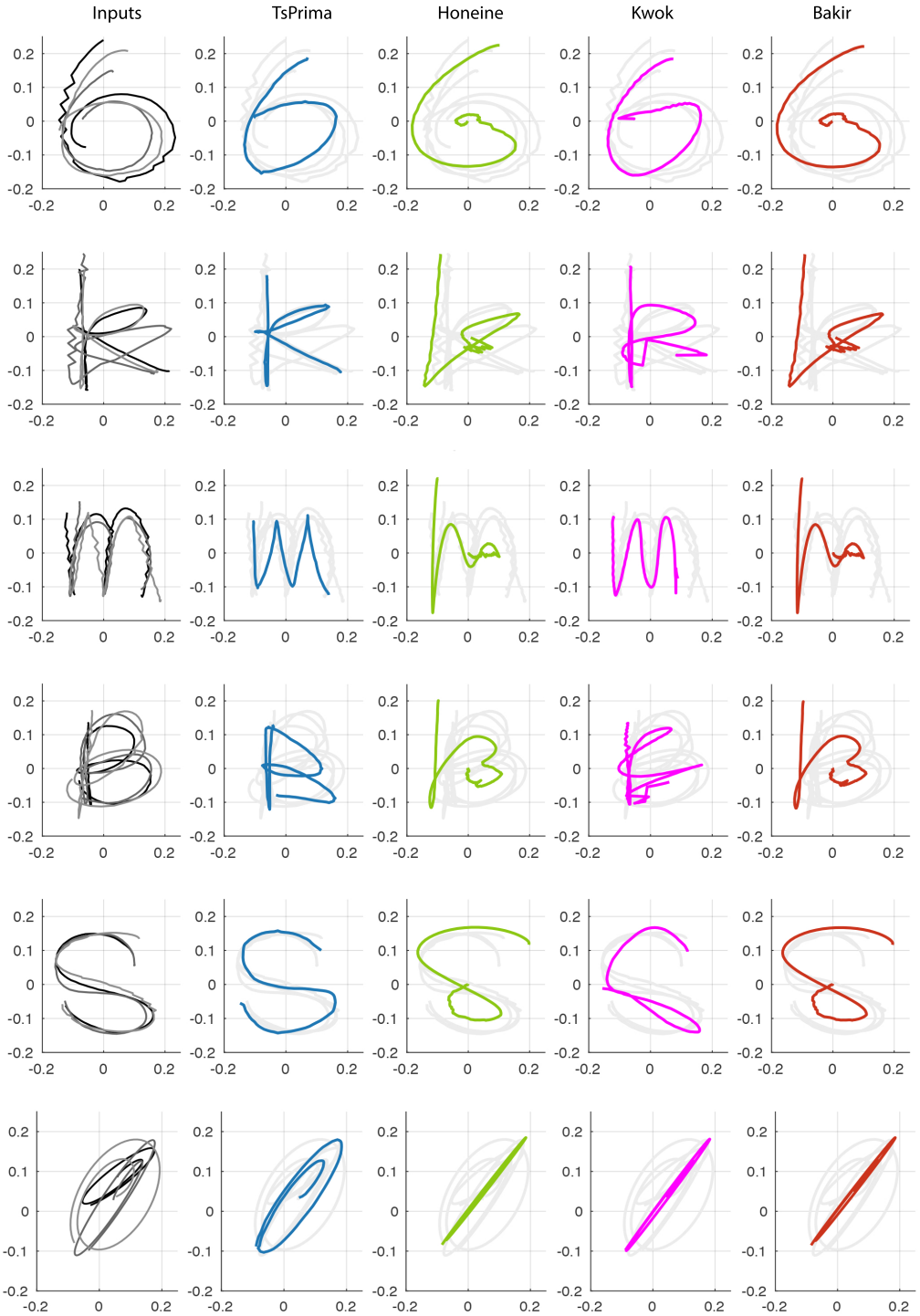


Figure 5.7: Time series centroids for some challenging classes of DIGITS, LOWER, UPPER and SPIRAL1 datasets.

Table 5.4: Quality of the time series reconstruction under kernel PCA

Dataset	TsPrima	Honeine	Kwok	Bakir
CC	**0.798**	0.747	0.758	0.747
GUNPOINT	0.994	**0.996**	0.992	0.99
CBF	**0.916**	0.854	0.896	0.875
OSULEAF	0.997	**0.998**	0.995	**0.998**
SWEDISHLEAF	**0.798**	0.701	0.69	0.65
TRACE	**0.689**	0.519	0.597	0.519
FACEFOUR	**0.981**	0.951	0.967	0.964
LIGHTING2	**0.993**	0.967	0.984	0.975
LIGHTING7	**0.954**	0.92	0.938	0.922
ECG200	0.965	**0.979**	0.959	0.962
ADIAC	**0.194**	0.127	0.139	0.125
FISH	**0.779**	0.58	0.586	0.579
BEEF	0.528	0.703	0.643	**0.704**
COFFEE	0.584	**0.595**	0.57	0.559
OLIVEOIL	**0.150**	0.125	0.141	0.121
DIATOMSIZER	**0.330**	0.174	0.186	0.173
ECG5DAYS	**0.996**	**0.996**	0.995	0.995
FACESUCR	**0.939**	0.825	0.878	0.847
ITALYPOWERD	0.831	**0.892**	0.023	0.851
MEDICALIMAGES	**0.946**	0.906	0.935	0.928
MOTESTRAIN	0.971	**0.987**	0.97	0.979
SONYAIBOII	0.978	**0.989**	0.969	0.985
SONYAIBO	0.939	**0.98**	0.924	0.967
SYMBOLS	**0.885**	0.822	0.724	0.761
TWOLEADECG	**0.825**	0.63	0.444	0.669
SPIRAL1	**0.961**	0.939	0.933	0.911
SPIRAL2	**0.966**	0.939	0.946	0.94
POWERCONS	0.971	0.966	0.955	**0.977**
BME	**0.896**	0.800	0.858	0.666
UMD	0.885	0.855	**0.904**	0.797
DIGITS	**0.84**	0.721	0.798	0.726
LOWER	**0.787**	0.696	0.747	0.685
UPPER	**0.856**	0.678	0.787	0.687
Nb. Best	**22**	9	1	3
Avg. Rank	**1.56**	2.67	2.71	3.06

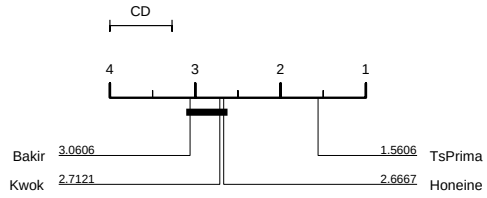


Figure 5.8: Nemenyi test: comparison of pre-image methods under kernel PCA reconstruction

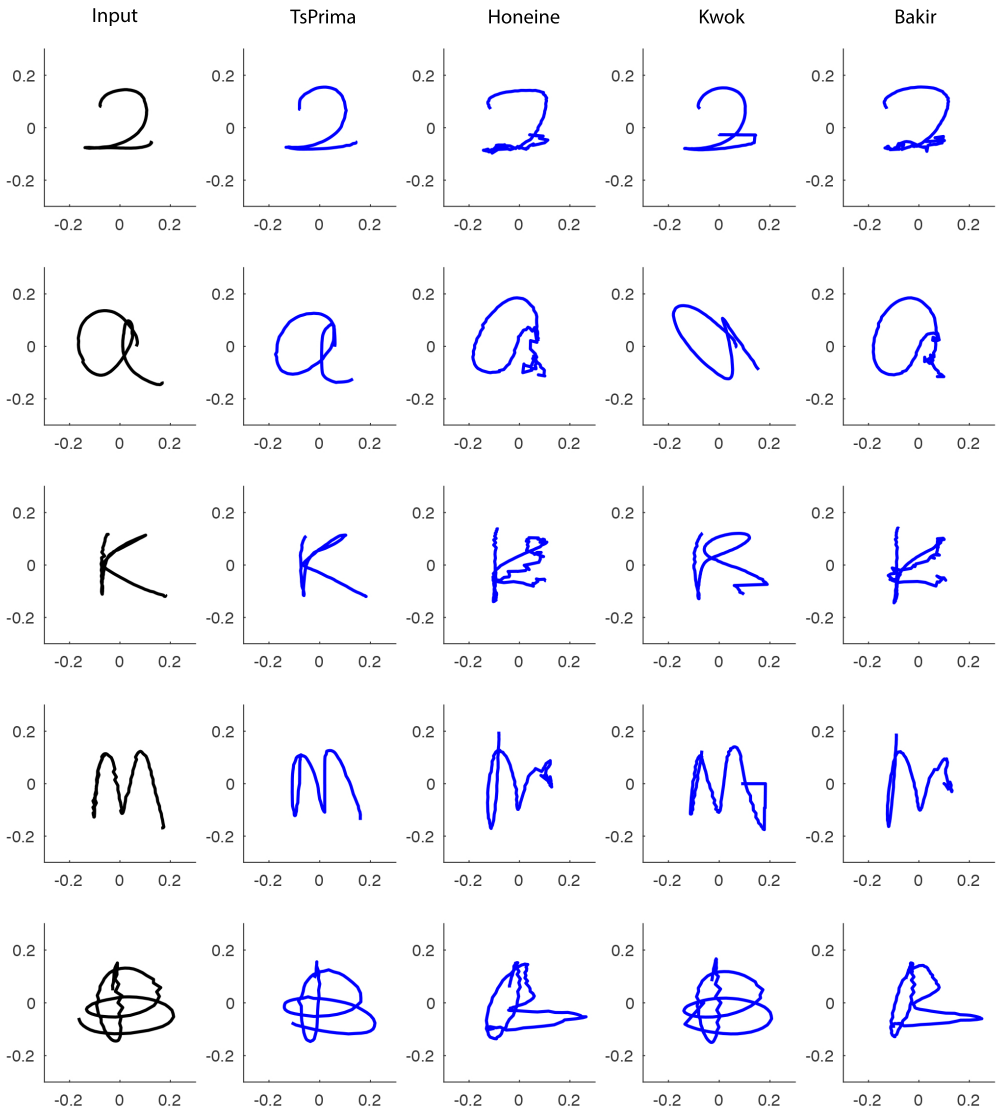


Figure 5.9: The time series reconstruction under kernel PCA of some samples of DIGITS, LOWER and UPPER datasets

For the time series denoising task, first a kernel PCA is performed on the training set, then a $(0, \sigma^2)$ Gaussian noise is added to the test samples $\boldsymbol{x}$ to generate noisy samples $\tilde{\boldsymbol{x}}$ with different variances σ^2. The denoised sample is obtained as the pre-image $\boldsymbol{x}^*$ of its kernel PCA projection $P(\Phi(\tilde{\boldsymbol{x}}))$, with $\boldsymbol{\gamma}$ defined as in Eq. (5.1). Similarly, the quality of the denoising is measured as the similarity $\text{DTAK}(\boldsymbol{x}^*, \boldsymbol{x})$ between $\boldsymbol{x}^*$ and the initial $\boldsymbol{x}$. Table 5.5 gives, for different values of σ^2, the average quality of the denoising for some datasets. Figure 5.10 illustrates the denoising results for some challenging times series of the noisy SPIRAL2 data and of the class "M" of UPPER dataset.

Table 5.5: Quality of the denoising for several noise levels

Dataset	σ^2	TSPRIMA	Honeine	Kwok	Bakir
DIGITS	0.01	**0.832**	0.669	0.782	0.666
	0.05	**0.808**	0.619	0.742	0.627
	0.1	**0.791**	0.605	0.723	0.612
	0.15	**0.783**	0.598	0.719	0.606
LOWER	0.01	**0.766**	0.651	0.721	0.637
	0.05	**0.746**	0.614	0.689	0.606
	0.1	**0.736**	0.601	0.675	0.596
	0.15	**0.729**	0.594	0.67	0.591
UPPER	0.01	**0.837**	0.627	0.765	0.638
	0.05	**0.806**	0.579	0.712	0.6
	0.1	**0.789**	0.561	0.688	0.59
	0.15	**0.782**	0.554	0.679	0.586
Nb. Best		**12**	0	0	0
Avg. Rank		**1.00**	3.58	2.00	3.42

5.5 Time series representation learning by pre-image estimation

For time series representation learning, the kernel k-SVD ($\tau = 5$) is used to learn, for each class of the considered datasets, the dictionary $\Phi(X)\mathcal{B}$ and the sparse representations $\mathcal{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_N]$ of its membership time series, as defined in Section 2.2. The pre-images D^* and X^* of the dictionary $\Phi(X)\mathcal{B}$ and of the sparse codes $\mathcal{A}$ are then obtained by considering $\boldsymbol{\gamma} = \mathcal{B}$ and $\boldsymbol{\gamma} = \mathcal{B}\mathcal{A}$, respectively. The quality of

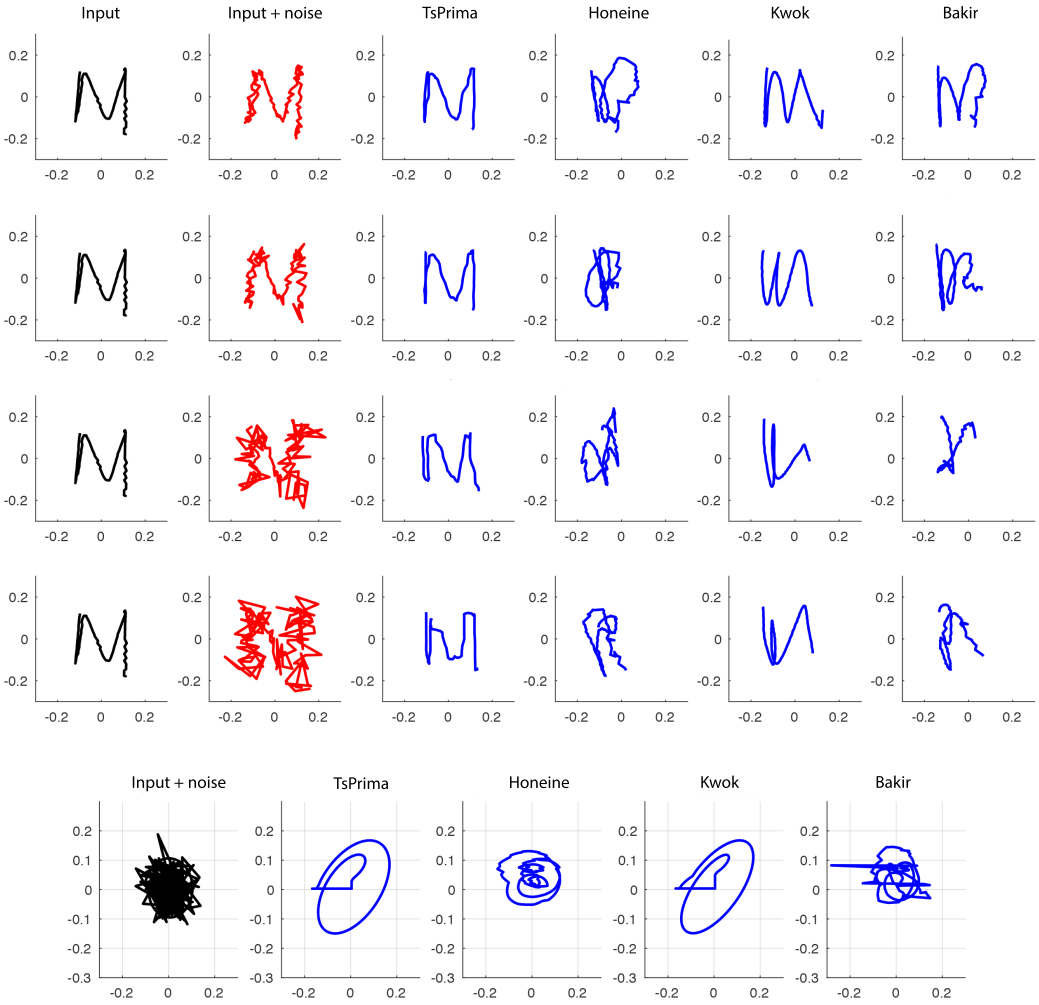


Figure 5.10: Time series denoising under kernel PCA of noisy samples of SPIRAL2 and of the class “M" of UPPER dataset.

the learned sparse representations is then measured as the similarity $\text{DTAK}(\boldsymbol{x}_i, \boldsymbol{x}_i^*)$ between each time series $\boldsymbol{x}_i$ and the pre-image $\boldsymbol{x}_i^*$ of the sparse representation $\Phi(X)\mathcal{B}\,\boldsymbol{a}_i$. Table 5.6 gives the average quality of the learned representations for each dataset and each pre-image estimation method. Figure 5.11 gives the critical difference diagram related to the Nemenyi test for the average ranking comparison of the studied methods. Figure 5.12 shows the learned representations for some time series of DIGITS, LOWER and UPPER datasets and Figure 5.13 illustrates, for a challenging sample of the class “k" of LOWER dataset, the learned representations as well as the top 3 atoms involved in its reconstruction.

Table 5.6: Quality of the time series representation learning under Kernel k-SVD

Dataset	TsPrima	Honeine	Kwok	Bakir
CC	**0.788**	0.73	0.751	0.732
GUNPOINT	0.993	**0.994**	0.992	0.985
CBF	**0.917**	0.862	0.9	0.872
OSULEAF	**0.996**	**0.996**	0.995	**0.996**
SWEDISHLEAF	**0.789**	0.659	0.691	0.623
TRACE	**0.687**	0.514	0.602	0.514
FACEFOUR	**0.971**	0.94	0.959	0.947
LIGHTING2	**0.991**	0.961	0.982	0.968
LIGHTING7	**0.961**	0.934	0.947	0.934
ECG200	0.953	**0.957**	0.95	0.941
ADIAC	**0.184**	0.122	0.131	0.117
FISH	**0.757**	0.553	0.579	0.56
BEEF	0.411	0.555	0.605	**0.621**
COFFEE	0.596	**0.607**	0.586	0.56
OLIVEOIL	0.145	0.133	**0.152**	0.12
DIATOMSIZER	**0.287**	0.177	0.198	0.178
ECG5DAYS	**0.996**	**0.996**	0.995	0.994
FACESURC	**0.917**	0.834	0.878	0.842
ITALYPOWERD	**0.8**	0.781	0.034	0.728
MEDICALIMAGES	**0.937**	0.86	0.93	0.878
MOTESTRAIN	0.969	0.97	**0.971**	0.97
SONYAIBOII	0.974	**0.975**	0.973	**0.975**
SONYAIBO	0.932	**0.938**	0.93	0.936
SYMBOLS	**0.811**	0.785	0.794	0.755
TWOLEADECG	**0.81**	0.617	0.411	0.629
SPIRAL1	**0.944**	0.913	0.92	0.914
SPIRAL2	**0.964**	0.936	0.949	0.937
POWERCONS	**0.968**	0.946	0.957	0.951
BME	**0.872**	0.734	0.843	0.622
UMD	0.888	0.842	**0.905**	0.788
DIGITS	**0.822**	0.699	0.793	0.706
LOWER	**0.773**	0.678	0.738	0.671
UPPER	**0.840**	0.664	0.797	0.675
Nb.Best	**24**	7	3	3
Avg.Rank	**1.50**	3.02	2.33	3.15

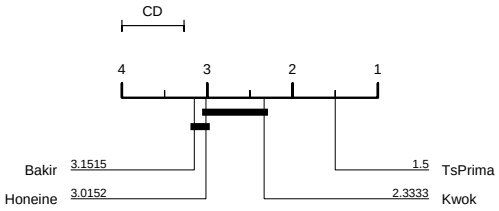


Figure 5.11: Nemenyi test: comparison of pre-image methods under kernel k-SVD representation learning

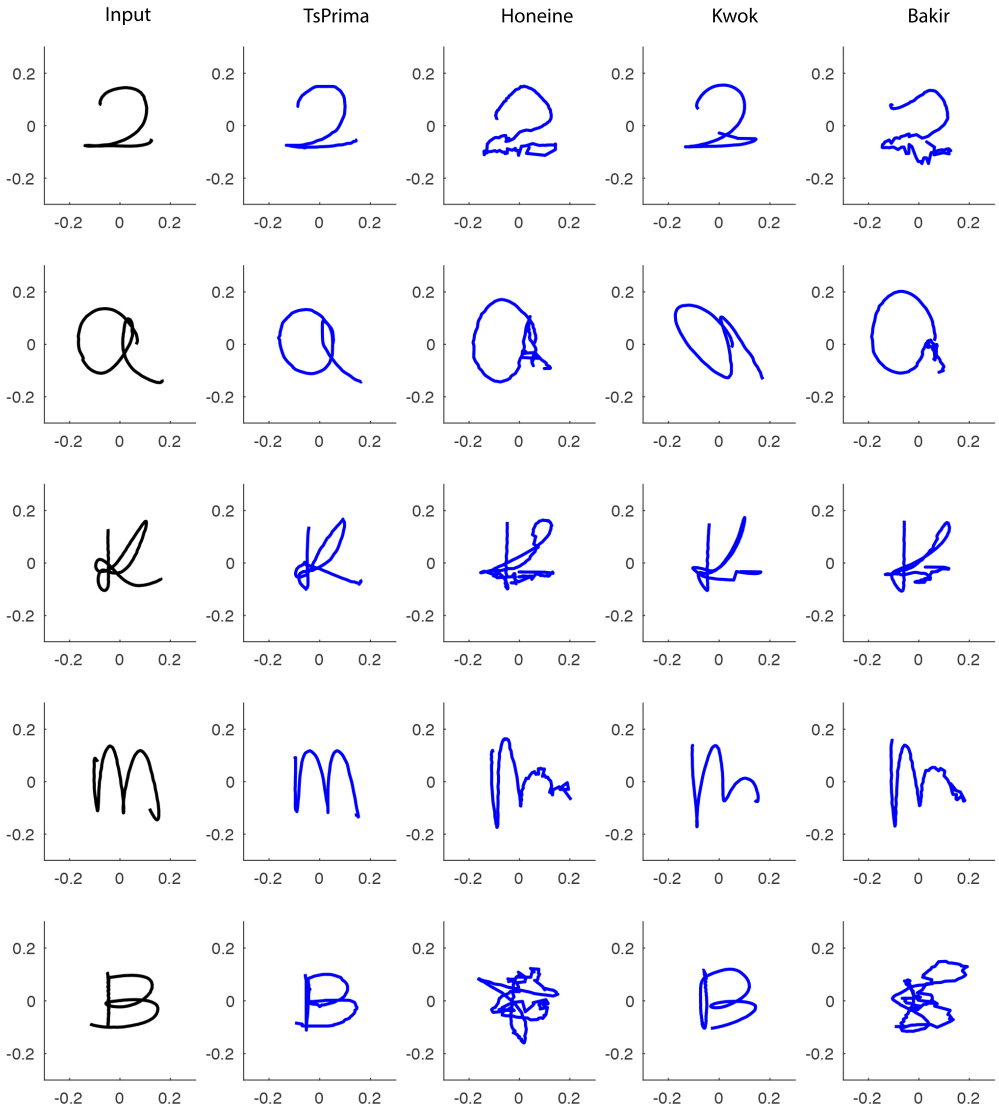


Figure 5.12: The learned time series representations under kernel k-SVD of some samples of DIGITS, LOWER, UPPER datasets

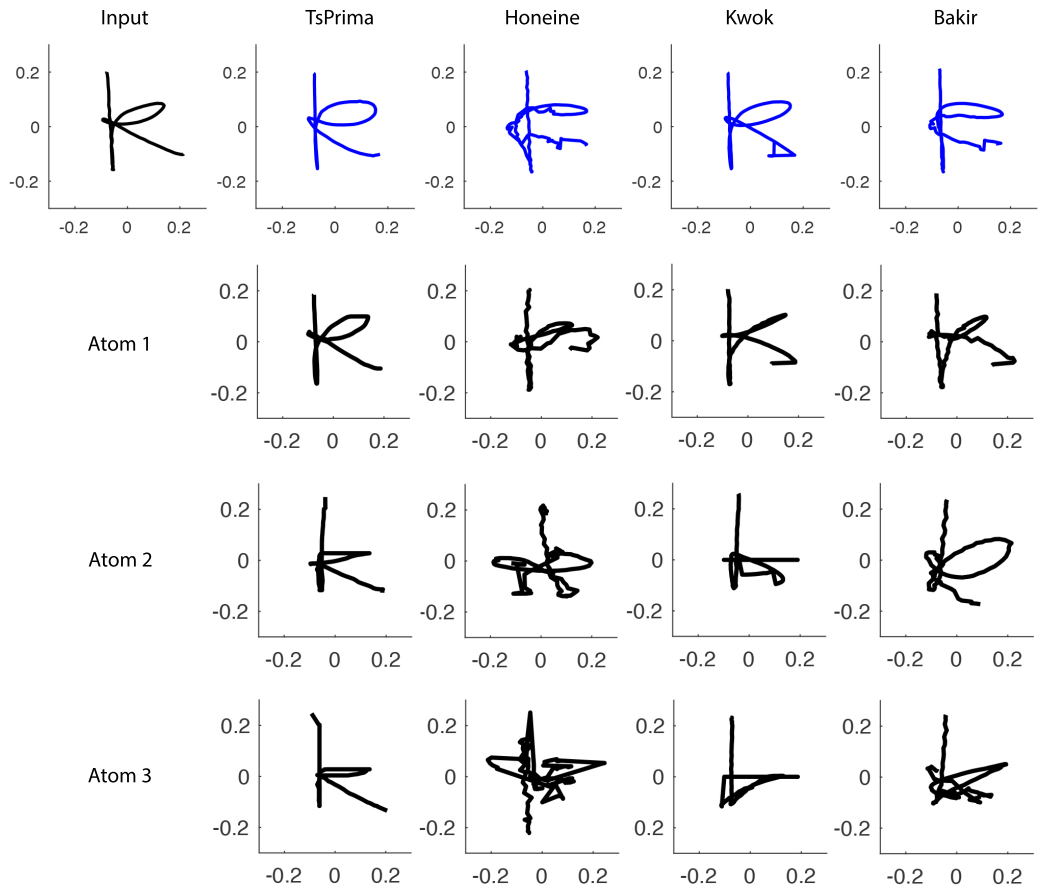


Figure 5.13: The sparse representation of a time series of the class "k" of LOWER dataset and the top 3 involved atoms for its reconstruction

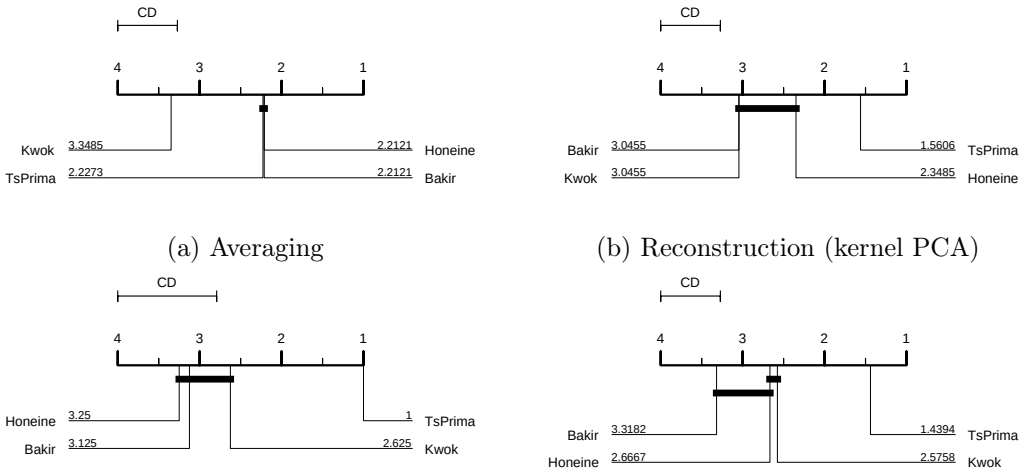
5.6 Further comparison

In the previous experiments (Sections 5.3 to 5.5), we have evaluated the performances of TSPRIMA that are mainly due to two major ingredients : 1) the defined temporal embedding function f_r (Section 4.3.3) and 2) the proposed transformation R to preserve an isometry between the time series embedding space and the feature space (Section 4.2). In this last part, the aim is to evaluate the efficiency of the proposed transformation R, regardless of the effect of f_r. For that, TSPRIMA is compared to the alternative methods Honeine, Kwok and Bakir once all the time series embedded into the same metric space; namely, all the pre-image estimation methods are performed between the time series embedding space and the feature space. Similar experiments are performed on the 33 public datasets (Table 5.1),

the results obtained for the three tasks are summarised into Table 5.7 and the related Nemenyi tests are given in Figure 5.14.

Table 5.7: Further comparisons for pre-image estimation

		TsPrima	Honeine	Kwok	Bakir
Averaging	Nb. Best	**19**	**20**	4	**19**
	Avg. Rank	**2.23**	**2.21**	3.35	**2.21**
Reconstruction	Nb. Best	**24**	10	0	1
(kernel PCA)	Avg. Rank	**1.56**	2.35	3.05	3.05
Denoising	Nb. Best	**12**	0	0	0
(kernel PCA)	Avg. Rank	**1.50**	3.25	2.62	3.12
Rep. Learning	Nb. Best	**25**	8	1	2
(kernel kSVD)	Avg. Rank	**1.44**	2.67	2.58	3.32



(a) Averaging

(b) Reconstruction (kernel PCA)

(c) Denoising (kernel PCA)

(d) Reconstruction (kernel k-SVD)

Figure 5.14: Nemenyi Tests.

5.7 Overall analysis

The experiments conducted show that the proposed method TsPrima leads on almost all the datasets and through the three studied tasks to the best results. On the other hand, the performances obtained by the alternative methods seem slightly equivalent and lower than those obtained by TsPrima.

In particular, for time series averaging task, we can see in Table 5.3 that the centroids estimated by TsPrima lead to the highest within-class similarity on almost all the datasets, namely, each centroid obtained by TsPrima is in general

the closest to the set of time series it represents. The analysis of the critical difference diagram given in Figure 5.6 indicates that the next best results are obtained respectively by Bakir, Honeine, and Kwok methods. In addition, as the state of the art methods are connected by a solid bold line, their performances remain equivalent. From Figure 5.7, we can see that while all the methods succeed to restitute the centroids of some input classes (shown on the left column) as the class "6" of DIGITS and "S" of UPPER datasets, only TSPRIMA succeeds to estimate the centroids of the most challenging classes, as the "k" class of LOWER dataset and SPIRAL1.

For time series reconstruction, Table 5.4, shows that TSPRIMA leads to the highest reconstruction accuracy through almost all the datasets, followed by Honeine, Bakir and Kwok methods. Figure 5.8 indicates that there is no significant difference between the performances of the three state of the art methods (connected by a solid bold line). These results are assessed in Figure 5.9 that shows, for some input time series, the quality of the reconstructions obtained by TSPRIMA and the state of the art methods.

For the time series denoising task, we observe from Table 5.5 and for all the methods that the quality of the denoising decreases when the intensity of noise increases. This result is illustrated in Figure 5.10, that shows the denoising results of the time series "M" of UPPER dataset and of the highly noisy time series of SPIRAL2 dataset. In particular, note that Kwok and TSPRIMA methods lead to the best results on SPIRAL2 dataset and seem less sensitive to noise than Honeine and Bakir.

Lastly, for time series representation learning task, Table 5.6 indicates that each studied method leads to the best sparse representation for at least some datasets and that TSPRIMA perform better on almost all the datasets. Figure 5.12 shows the goodness of the sparse representation obtained. While all the methods succeed to sparse represent some input time series, the time series "k" and "B" classes appear challenging for Honeine and Bakir methods. In Figure 5.13, we get a look on the quality of the learned atoms, that are involved into the reconstruction of the input samples. The first row gives for some input samples "k" (on the left),

the sparse representation learned by each method. The three next rows, provide the three first atoms involved into the reconstructions. We can see that while the first atom learned by TsPrima is nearly sufficient to sparse present the "k" input sample, the state of the art methods need obviously more that one atom to sparse represent the input sample. Finally, the analysis of Figure 5.13 indicates that Honeine method performs equivalently that Kwok and Bakir, whereas the Kwok performances are significantly better than those of Bakir method.

Further comparisons (Table 5.7) are conducted in Section 5.6 to evaluate the efficiency of TsPrima related to the learned transformation R, regardless of the temporal embedding f_r. For averaging task, TsPrima, Honeine and Bakir lead equivalently to the best performances, followed by Kwok method (Figure 5.14 (a)). From these results we can conjecture that, linear transformations seem sufficient to achieve good pre-image estimations for averaging task on these datasets, as both linear and nonlinear approaches (TsPrima, Honeine, Bakir) perform equivalently. Furthermore, while Honeine and Bakir involve the whole datasets for the centroid pre-image estimations, Kwok uses a subset of samples into the neighbourhood of $\boldsymbol{\varphi}$, which may explain the slightly lower performances of Kwok method. Note that, although TsPrima involves, similarly to Kwok method, fewer samples into the neighbourhood of $\boldsymbol{\varphi}$, it succeeds to reach the best performances thanks to the efficiency of the learned transformation R. For the remaining tasks reconstruction, denoising and representation learning, TsPrima achieves the highest performances, followed by far by Honeine, Kwok and Bakir (Figure 5.14 (b), (c) and (d)), which assesses the crucial contribution of the learned transformations R of TsPrima. Lastly, of particular note is that Honeine and Bakir that involve the whole training samples induce much computations, specifically for the time series embedding process, than Kwok and TsPrima that require fewer samples into the neighbourhood of $\boldsymbol{\varphi}$.

Finally, as all the studied methods propose closed-form solutions, they lead to comparable complexities. However, for large data, TsPrima and Kwok methods

are expected to perform faster as requiring fewer samples on the neighbourhood of $\boldsymbol{\varphi}$ than Honeine and Bakir that involve the whole samples for pre-image estimation. Note that the complexity of the proposed solutions is mainly related to the matrix inversion operator. In Kwok method, the inversion of ZZ^T required in Eq. 3.19, where Z is of dimension $(q \times n)$ and n is the neighbourhood size, induces a complexity of $O(q^2n) + O(q^3)$; as q is in general small and fixed beforehand, the overall complexity is about $O(n)$. For Honeine method, Eq. 3.28 requires two inversions of XX^T and K, which induces, respectively, a complexity of $O(d^2N) + O(d^3)$ and $O(N^3)$, that leads to an overall complexity of $O(N^3)$. For Bakir method, Eq. 3.31, requires the inversion of the Gram matrix, which leads to a complexity of $O(N^3)$. For TsPrima, Eq. 4.12 involves the inversion of XX^T, where X is of dimension $(d \times n)$, d is the time series length and n is the neighbourhood size. The induced complexity is of $O(d^2n) + O(d^3)$. For the time series embedding part, the complexity is mainly related to the time warping function which is of order $O(d^2n)$. As d is in general higher than the neighbourhood size n, the overall complexity for TsPrima is about $O(d^3)$. To sum up, as the neighbourhood size $n << N$ and $d << N$ (for not extra large time series), the complexity induced by both Kwok and TsPrima remains lower than the one of Honeine and Bakir. Note that, the Honeine method can be developed to consider only the neighbourhoods instead of all samples. Finally, as all the studied methods propose closed-form solutions, they lead to comparable complexities. However, for large data, TsPrima and Kwok methods are expected to perform faster as requiring fewer samples on the neighborhood of $\boldsymbol{\varphi}$ than Honeine and Bakir that involve the whole samples for pre-image estimation.

5.8 Conclusion

This work proposes TsPrima, a new closed-form pre-image estimation method for time series analytics under kernel machinery. The method consists of two stages. In the first step, we define a time warp embedding function, driven by distance

constraints in the feature space, that allows to embed the time series in a metric space. In the second step, the time series pre-image estimation is cast as learning a linear (or a nonlinear) transformation to ensure a local isometry between the time series embedding space and the feature space. Extensive experiments show the efficiency and the benefits of TsPrima through three major tasks that require pre-image estimation: 1) time series averaging, 2) time series reconstruction and denoising and 3) time series representation and dictionary learning.

6

Conclusion and future work

In Chapter 2, we introduce three well-known kernel methods that are kernel PCA, kernel SVD and kernel regression. These methods have been used commonly for the analysis of complex and unstructured data by embedding the data into a feature space via a kernel mapping. The main trick behind these methods is to learn nonlinear structures in the input space by learning linear models in the feature space. While such approaches are fruitful and have widely proven their efficiency, the results obtained are lying in the kernel feature space, limiting further interpretations and analysis. The pre-image problem in then crucial to complement the kernel approaches and allows for any result obtained in the feature space to be restored into the initial space.

In Chapter 3, the three major methods (Section 3.3, 3.4,3.5) presented above define three different approaches for pre-image estimation problem. First of all, all the methods involve only linear algebra and propose solutions that don't suffer from numerical instabilities. In Kwok et al. [55], the solution is mainly requiring the definition of a relation between the distances into the input and the kernel feature spaces. That requirement limits the Kwok et al. [55] approach to linear or isotropic kernels. Honeine et al. [48] alleviate that point by proposing a closed-form solution that is applicable to any type of kernels. Furthermore, while in Honeine et al. [48] the pre-image estimation is obtained by learning a linear transformation into the feature space that preserves the isometry between the input and the feature space, in Bakir et al. [51], the pre-image estimation is obtained by using a non linear kernel regression that predicts the input samples from their images into the feature space. Finally, while both [48] and [51] proposals involve the whole training samples for pre-image estimation, Kwok et al. [55] uses only the samples on the neighborhood of φ, which offers a significant speed-up; highly valuable in the case of large scale data.

Chapter 4 proposed formulations and results for pre-image estimation (Section 4.2.1) presented some similarities and differences with the method proposed in [48] and presented in Section 3.4. First of all, both approaches propose formulations and solutions that only require linear algebra and are independent of the type of kernel. To establish the isometry, in [48] a linear transformation restricted to the form $R = \Phi(X)AA^T\Phi(X)^T$ is estimated, whereas in our proposal the estimated R may be linear Eq. (4.6) or non linear Eq. (4.15) and is importantly unconstrained, namely of general form which enlarges its potential to deal with complex structures. Finally, while in [48] the solution Eq. (3.28) involves the kernel information through the regularisation term, which may be canceled for lower values of λ, in the proposed solutions Eq. (4.13) and Eq. (4.19) the kernel information is entirely considered regardless of the regularisation specifications. To address the pre-image estimation for such challenging time series, we proposed an embedding function that allows to represent the time series in a metric space,

where the previous linear and nonlinear transformations method for pre-image estimation can be performed conveniently.

Chapter 5 evaluated the efficiency of the proposed pre-image estimation method under three major time series analysis tasks: 1) time series averaging, 2) time series reconstruction and denoising and 3) time series representation learning. The proposed pre-image estimation method TsPrima is compared to three major alternative approaches introduced in Chapter 3 as Honeine in Section 3.4, [48], Kwok in Section 3.3, [55], and Bakir in Section 3.5, [51] methods.
To sum up, this thesis proposes TsPrima, a new closed-form pre-image estimation method for time series analytics under kernel machinery. The method consists of two stages. In the first step, we define a time warp embedding function, driven by distance constraints in the feature space, that allows to embed the time series in a metric space. In the second step, the time series pre-image estimation is cast as learning a linear (or a nonlinear) transformation to ensure a local isometry between the time series embedding space and the feature space. Extensive experiments show the efficiency and the benefits of TsPrima through three major tasks that require pre-image estimation: 1) time series averaging, 2) time series reconstruction and denoising and 3) time series representation and dictionary learning.

Future work will explore the benefits of pre-image estimation methods in several deep learning contexts. Indeed, although deep learning approaches remain among the powerful machine learning methods today, the results obtained and the performance achieved remain generally inexplicable and uninterpretable, which is a major drawback in the context of diagnostic analysis. Therefore, future studies will focus on exploring new approaches based on prior image estimation to make deep learning results interpretable and explainable.

Bibliography

[1] Michal Aharon, Michael Elad, Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, page 54(11):4311–4322, 2006.

[2] Claus Bahlmann, Bernard Haasdonk, Hans Burkhardt. Online handwriting recognition with support vector machines - a kernel approach. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, IWFHR '02, pages 49–, Washington, DC, USA, 2002. IEEE Computer Society.

[3] Claus Bahlmann, Bernard Haasdonk, Hans Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. *Frontiers in Handwriting Recognition*, 26(1):43–54, 2002.

[4] King-Sun Fu, Y. T. Chien, Gerald P. Cardillo. A dynamic programming approach to sequential pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 8(3):313–26, Mar 1986.

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

[6] Hiroaky Sakoe, Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[7] Donald J. Berndt, James Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 workshop on knowledge discovery in databases*, volume 2, 1994.

[8] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[9] David L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, page 1289–1306, 2006.

[10] Jean-Luc Starck, Emmanuel J. Candès, David L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 2002.

[11] Saeed Varasteh Yazdi, 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.

[12] Keogh, Eammon. The ucr time series data mining archive. *http://www. cs. ucr. edu/˜ eamonn/TSDMA/index. html*, 2002.

[13] Boaz Ophir, Michael Lustig, Michael Elad. Multi-scale dictionary learning using wavelets. *IEEE Journal of Selected Topics in Signal Processing*, 2011.

[14] Michal Aharon, Michael Elad. Sparse and redundant modeling of image content using an image signature dictionary. *SIAM Journal on Imaging Sciences*, 1(3):228–247, 2008.

[15] Cedric Frambourg, Ahlame Douzal-Chouakria, Eric Gaussier. Learning multiple temporal matching for time series classification. In *International Symposium on Intelligent Data Analysis*, pages 198–209. Springer, 2013.

[16] Saeid Soheily-Khah, Ahlame Douzal-Chouakria, Eric Gaussier. Progressive and iterative approaches for time series averaging. In *AALTD@ PKDD/ECML*, 2015.

[17] Saeid Soheily-Khah, Ahlame Douzal-Chouakria, Eric Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp. *Pattern Recogn. Lett.*, 75(C):63–69, May 2016.

[18] Stéphane G.Mallat. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, page 3397–3415, 1993.

[19] S. Chiba H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process*, pages 43–49.

[20] K. Kamgar C.C.M. Yeh Y. Zhu S. Gharghabi C.A. Ratanamahatana Hu B. Yanping N. Begum A. Bagnall A. Mueen G. Batista Hexagon-ML H.A. Dau, E. Keogh. The ucr time series classification archive. 2018.

[21] K. Engan, S.O. Aase, J. Hakon Husoy. Method of optimal directions for frame design. *Acoustics, Speech, and Signal Processing, 1999*, 1999.

[22] Hiroshi Shimodaira, Ken ichi Noma, Mitsuru Nakai, Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in neural information processing systems*, pages 921–928, 2002.

[23] Vladimir Vapnik Jason Weston, Olivier Chapelle, Andre Elisseeff, Bernhard Schölkopf. Kernel dependency estimation. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 897–904, Cambridge, MA, USA, 2002. MIT Press.

[24] J.C.Gower. Adding a point to vector diagrams in multivariate analysis. volume 55, pages 582–585. Biometrika, 1968.

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

[26] Mahdi Ataee, Hadi Zayyani, Massoud Babaie-Zadeh, Christian Jutten. Parametric dictionary learning using steepest descent. *IEEE International Conference*, 2010.

[27] Schölkopf, A.JSmola, K.-R.Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, pages 1299–1319, 1998.

[28] Hiroshi, Ken-ichi. Dynamic time-alignment kernel in support vector machine. *nips.cc*, 2007.

[29] Patrick Etyngier, Florent Ségonne, Renaud Keriven. Shape priors using manifold learning techniques. *In 11th IEEE international conference on computer vision. Rio de Janeiro, Brazil*, 2007.

[30] Bernard Haasdonk, Daniel Keysers. Tangent distance kernels for support vector machines. *In 16th ICPR*, page 24, 2002.

[31] Yagyensh Chandra Pati, Ramin Rezaiifar, Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44, 1993.

[32] Grigorios F. Tzortzis, Aristidis C. Likas. The global kernel k-means algorithm for clustering in feature space. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 20(7):1181–94, Jul 2009.

[33] John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 210–227, 2009.

[34] Stéphane Mallat. A wavelet tour of signal processing. *Academic press*, 1999.

[35] Q. Barthélemy, A. Larue, A. Mayoue, D. Mercier, J. Mars. Shift & 2d rotation invariant sparse coding for multivariate signals. *Signal Processing, IEEE Transactions*, page 1597–1611, 2011.

[36] Marco Cuturi, Jean-Philippe Vert, Birkenes, Oystein Birkenes, Tomoko Matsui. A kernel for time series based on global alignments. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pages II–413. IEEE, 2007.

[37] Lei Zhang Meng Yang. Gabor feature based sparse representation for face recognition with gabor occlusion dictionary. *European conference on computer vision, Springer*, 2010.

[38] Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. volume 2, chapter Constructive Approximation, pages 11–22. Springer-Verlag, Jul 1986.

[39] 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. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, 2018.

[40] Bernhard Schölkopf, Sebastian Mika, Alex Smola, Gunnar Rätsch, Klaus-Robert Mülle. Kernel pca pattern reconstruction via approximate pre-images. In *International Conference on Artificial Neural Networks*, pages 147–152. Springer, 1998.

[41] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller. Advances in kernel methods. chapter Kernel Principal Component Analysis, pages 327–352. MIT Press, Cambridge, MA, USA, 1999.

[42] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller. Kernel principal component analysis. *In Advances in Kernel Methods – Support Vector Learning, MIT Press*, page 327–352, 1999.

[43] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopft, Klaus-Robert Müller. Fisher discriminant analysis with kernels. *Neural networks for signal processing IX*, 1999.

[44] N.Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[45] Yoshua Bengio, Jean-Francois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, Marie Ouimet. Out-of-sample extensions forlle, isomap, mds, eigenmaps, and spectral clustering. *In neuralinformation processing systems*, 2004.

[46] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, Gunnar Rätsch. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11(1):536–542, 1999.

[47] Cyril Joder, Slim Essid, Gaël Richard. Temporal integration for audio classification with application to musical instrument classification. *IEEE Transactions on Audio, Speech and Language Processing*, page 174–186, 2009.

[48] Paul Honeine, Cédric Richard. A closed-form solution for the pre-image problem in kernel-based machines. *Journal of Signal Processing Systems*, 65(3):289–299, 2011.

[49] Pablo Arias, Gregory Randall, Guillermo Sapiro. Connecting the out-of-sample and pre-image problems in kernel methods. *In IEEE Computer Society conference on computer vision and pattern recognition*, 2007.

[50] Kevin Schlegel. When is there a representer theorem? *Journal of Global 540 Optimization 74*, page 401–415, 2009.

[51] Gökhan H. Bakır, Jason Weston, Bernhard Schölkop. Learning to find pre-images. In *Advances in Neural Information Processing Systems 16*, pages 449–456. MIT Press, 2004.

[52] Dennis Decoste, Bernhard Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190, 2002.

[53] Bernhard Schölkopf, Alexander Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

[54] Ryan J. Tibshirani. The lasso problem and uniqueness. *Electronic Journal of Statistics*, page 1456–1490, 2013.

[55] James T. Kwok, Ivor W. Tsang. The pre-image problem in kernel methods. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 15(6):1517–25, Nov 2004.

[56] 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*, page 1–5, 2008.

[57] Corinna Cortes, Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[58] H. Drucker, C. Burges, L. Kaufman, A. Smola, V. N. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, 1996.

[59] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[60] Do Cao-Tri, 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*, 418:272–285, 2017.

[61] Minh N. Do, Martin Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Process- ing*, 2005.

[62] C. Saunders, A. Gammerman, V. Vovk. Ridge regression learning algorithm in dual variables. *In Proceedings of the 15th International Conference on Machine Learning*, page 515–521, 1998.

[63] 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*, 2018.

[64] Tanaya Guha, Rabab K. Ward. Learning sparse representations for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1576–1588, 2012.

[65] B. Hu N. Begum A. Bagnall A. Mueen G. Batista Y. Chen, E. Keogh. The ucr time series classification archive. 2015.

[66] R. Rezaiifar, P.S. Krishnaprasad Y.C. Pati, Y.C. 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.

[67] Elisa Ricci, Gloria Zen. Learning pedestrian trajectories with kernels. *ICPR, IEEE Computer Society. IEEE Computer Society*, page 149–152, 2010.