Signal structure: from manifolds to molecules and structured sparsity

THÈSE N^{O} 6832 (2015)

PRÉSENTÉE LE 3 DÉCEMBRE 2015

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE TRAITEMENT DES SIGNAUX 4

PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sofia KARYGIANNI

acceptée sur proposition du jury:

Dr J.-M. Vesin, président du jury
Prof. P. Frossard, directeur de thèse
Prof. C. Guillemot, rapporteuse
Prof. C. De Vleeschouwer, rapporteur
Prof. J.-Ph. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

Με τη καρδιά, με τι πνοή,
τι πόθους και τι πάθος
πήραμε τη ζωή μας· λάθος!
κι αλλάξαμε ζωή.

Άρνηση, Γιώργος Σεφέρης

With what spirit, what heart,
what desire and passion
we lived our life· a mistake!
So we changed our life.

Denial, Giorgos Seferis

Acknowledgements

The idea of getting a PhD has been introduced to me early in my life and since then it has always stayed with me. Now, this path has come to an end and it has been quite a journey; a journey that would not have been the same without the contribution of some people.

First of all, I would like to thank my advisor, Prof. Pascal Frossard. During all the years that I worked with him, Pascal has always been very friendly, considerate and supportive. I will always feel grateful for the opportunity he gave me to study in such a nice environment, for the freedom he allowed me in picking the directions of research for my degree and for all the things I learned from him.

I would also like to thank the members of my thesis committee, Prof. Jean-Marc Vesin, Prof. Jean-Philippe Thiran, Prof. Christine Guillemot and Prof. Christophe De Vleeschouwer, for reading my thesis, evaluating my work and giving me meaningful feedback.

Nothing would have been the same in my PhD if I have not been lucky enough to have great friends and colleagues to accompany me on the way. Many thanks to my LTS mates Xiaowen, Ana, Laura, Thomas, David, Pinar, Hussein, Stefano, Eirini, Nikos, Elif, Tamara, Luigi, Zafer, Vijay, Eymen, David, Francesca, Renata, Mattia, Anna, Alia, Alessandra, Eleni, Meri and Elda for all the great activities we organized together and for all the uplifting chit-chat sessions. Special thanks to Dorina for always being there to listen me and share with me all the bad and the good moments along the way. I would also like to thank my friends Iris, Christina, Vicky, Giorgos and Lorenzo for all the 'extra-EPFL' fun activities as well as their valuable advice and support in times of need. Special thanks to Emre for his unconditional support, tolerance and patience especially during the last and toughest part of the completion of this thesis. I would also like to express my gratitude to my childhood friend Ifigeneia who is my friend as long as I remember myself and whom I always feel close even when separated by thousands of kilometers.

Last but not least, I would like to thank my family. Many thanks to my uncle, John, for being an inspirational figure in my life. I would also like to thank my dad, Aristeidis, my mom, Voula, and my sister, Ioanna, for all the support and love they have given me, each in his unique personal way.

Sofia

Abstract

Effective representation methods and proper signal priors are crucial in most signal processing applications. In this thesis we focus on different structured models and we design appropriate schemes that allow the discovery of low dimensional latent structures that characterize and identify the signals.

Motivated by the highly non-linear structure of most datasets, we firstly investigate the properties and the geometry of manifolds. Manifolds are low dimensional, non-linear structures embedded in a higher dimensional space. They are naturally employed to describe sets of strongly related signals such as the images of an 3-D object captured from different viewpoints or the images of objects belonging to the same category but having different appearances. However, despite the direct link between signals and manifolds, the use of manifolds in applications is not straightforward due to their usually complex, non-analytic and non-linear form. We propose a way to 'disassemble' a manifold into simpler, more flexible components by approximating it with affine subspaces. Our objective is to discover a set of low dimensional affine subspaces that can represent manifold data accurately while preserving the manifold's structure. To this end, we employ a greedy technique that iteratively merges manifold samples into groups based on the difference of local tangents. We use our algorithm to approximate synthetic and real manifolds and to demonstrate that it is competitive to state-of-the-art techniques.

Then, we consider different signal models that are represented by structured sparse representations. While sparsity has been one of the major drives in signal processing in the last decade, structured sparsity, where the support defined by the signal components is considered in addition to the number of elements, has also lately emerged as a way to enrich signal priors towards more meaningful and accurate representations. In this thesis we propose a new sparsity model, where signals are essentially composed of a small number of structured *molecules* . We define the molecules to be linear combinations of a small number of elementary functions in a redundant dictionary. Our new multi-level model takes into account the energy distribution of the significant signal components in addition to their support. It permits to define typical visual patterns and recognize them in prototypical or deformed form, a quality that is particularly useful in the reconstruction of noisy or incomplete images. We define a new structural difference measure between molecules and their deformed versions, which is based on their sparse codes. We create an algorithm for decomposing signals into molecules that can

account for different deviations in the internal molecule structures, from small errors in the coefficients to deviations on both the coefficients and the support of the molecule prototypes. Our experiments verify the benefits of the new image model in various image restoration tasks. They confirm that the development of proper models that extend the mere notion of sparsity can be very useful for various inverse problems in imaging, especially if the original data is of low quality. In addition, our model provides evidence of the extra power of richer signal priors when equipped with similarity measures and flexible sparse coding.

Finally, we investigate the problem of learning molecule representations directly in the sparse code domain. We constrain sparse codes to be linear combinations of a few, possibly deformed, molecules and we design an algorithm that can learn the structure from the codes without transforming them back into the signal domain. To this end, we take advantage of our structural difference which is based on the sparse codes and we devise a scheme for representing the codes with molecules and learn the molecules at the same time. To illustrate the effectiveness of our proposed algorithm we apply it to various synthetic and real datasets and we compare the results with traditional sparse coding and dictionary learning techniques. From the experiments, we verify the superior performance of our scheme in interpreting and recognizing correctly the underlying structure.

In short, in this thesis we are interested in low-dimensional, structured models. Among the various choices, we focus on manifolds and sparse representations and we propose schemes that enhance their structural properties and highlight their effectiveness in signal representations.

Keywords: manifolds, approximation, flats, low-dimensional, structure, sparsity, linear combinations, two-level, dictionaries, molecules, deformations, structure learning

Résumé

La représentation des signaux représente un choix crucial pour de nombreuses applications en traitement du signal. Dans cette thèse, nous nous concentrons sur différents modèles structurés et nous proposons des algorithmes appropriés qui permettent la découverte de structures latentes de basse dimension qui caractérisent et identifient les signaux.

Inspiré par la structure fortement non-linéaire de la plupart des signaux que l'on rencontre en pratique, nous examinons tout d'abord les propriétés et la géométrie des variétés. Les variétés sont des structures non-linéaires de basse dimension plongés dans un espace de dimension supérieure. Ils sont naturellement utilisées pour décrire des ensembles de signaux fortement liés comme les images d'un objet 3-D capturé à partir de différents points de vue ou les images d'objets appartenant à la même catégorie, mais ayant des apparences différentes. Cependant, malgré le lien direct entre les signaux et les variétés, l'utilisation de variétés dans les applications n'est pas simple en raison de leurs formes généralement complexe, non-analytique et non-linéaire. Nous proposons un moyen à 'démonter' les variétés en composantes simples en utilisant une approximation des variétés en sous-espaces affines. Notre objectif est de découvrir un ensemble de sous-espaces affines de basse dimension qui peut représenter les signaux dans la variété précisément tout en préservant la structure de la variété. Pour ce faire, nous employons un algorithme glouton qui divise de manière itérative les multiples échantillons de la variété en groupes sur la base de la différence des tangentes locales. Nous utilisons notre algorithme pour approximer des variétés synthétiques et réelles et démontrons experimentalement que notre méthode est compétitive avec l'état de l'art.

Ensuite, nous considérons différents modèles de signaux parcimonieux structurés. Alors que la parcimonie a été l'une des grandes tendances en traitement des signaux dans la dernière décennie, la parcimonie structurée, où le support défini par les composantes du signal est considérée en plus du nombre d'éléments, est apparu récement, permettant ainsi d'avoir des représentations plus significatives, interprétables et précises. Dans cette thèse, nous proposons un nouveau modèle de parcimonie, où les signaux sont essentiellement composés d'un petit nombre de *molécules* structurées. Nous définissons les molécules comme des combinaisons linéaires d'un petit nombre de fonctions élémentaires dans un dictionnaire redondant. Notre nouveau modèle à niveaux multiples prend en compte la distribution d'énergie des composantes importantes du signal, en plus de leurs supports. Il permet de définir des modèles visuels typiques assez flexible pour inclure une forme prototypique ainsi que des déformations.

Notre modèle est particulièrement utile dans la reconstruction des images bruitées ou incomplètes. Nous définissons une nouvelle mesure de différence structurelle entre les molécules et leurs versions déformées, qui est basée sur leurs représentations parcimonieuses. Nous proposons un algorithme pour décomposer les signaux en molécules flexibles qui permettent de tenir compte des différentes déviations dans les structures internes de la molécule, de petites erreurs dans les coefficients à des déviations plus importantes au niveau des coefficients et du support des prototypes de molécules. Nos expériences permettent de vérifier les avantages du nouveau modèle dans différentes tâches de restauration d'images. Cela confirme que l'extension de la simple notion de parcimonie peut être très utile pour divers problèmes inverses en imagerie, en particulier si les données d'origine sont de faible qualité. En outre, notre modèle fournit une preuve de l'importance de la méthode de représentation quand elle est équipée de mesures de similarités et représentation parcimonieuses flexibles.

Enfin, nous étudions le problème de l'apprentissage des représentations de molécules directement dans le domaine de représentations parcimonieuses. Nous limitons les représentations parcimonieuses à des combinaisons linéaires de quelques molécules, possiblement déformées, et nous proposons un algorithme qui peut apprendre la structure à partir des codes parcimonieux sans avoir à les transformer de nouveau dans le domaine des signaux originaux. Pour ce faire, nous profitons de notre mesure de différence structurelle entre les molécules et leurs versions déformées qui est définie à l'aide des codes parcimonieux et nous proposons un algorithme pour représenter les codes avec des molécules et apprendre les molécules en même temps. Pour illustrer le bon fonctionnement de notre algorithme, nous l'appliquons à divers ensembles de données synthétiques et réelles et nous comparons les résultats avec des techniques d'apprentissage de dictionnaire et de représentation parcimonieuse. D'après les expériences, nous vérifions la performance supérieure de notre algorithme dans l'interprétation et la reconnaissance de la structure sous-jacente.

En bref, dans cette thèse, nous nous sommes intéressés à des modèles à faibles dimensions et structurés. Parmi les différents choix, nous nous concentrons sur les variétés et les représentations parcimonieuses et nous proposons des algorithmes qui améliorent leurs propriétés structurelles et mettent en valeur leurs efficacités dans les représentations des signaux.

Mots clefs : variétés, approximation, faible dimension, structure, parcimonie, combinaisons linéaires, deux niveaux, dictionnaires, molécules, déformations, apprentissage de la structure.

Contents

List of Figures

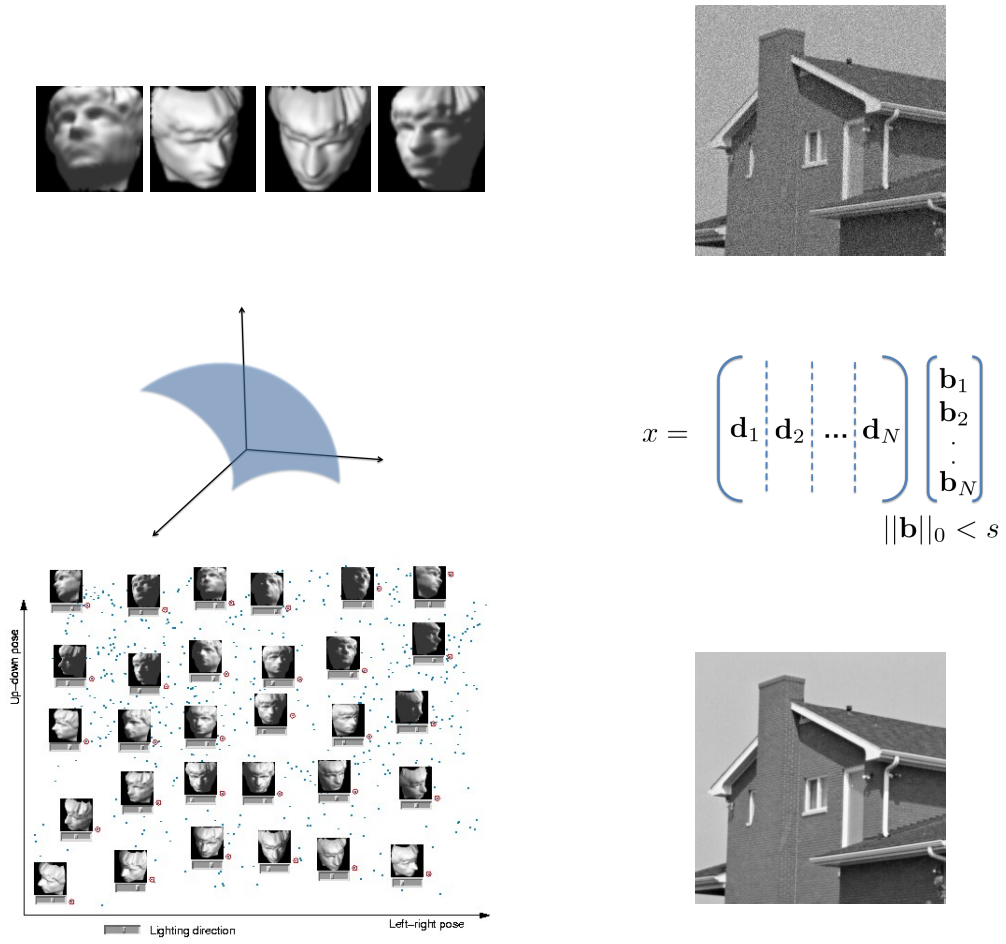
List of Tables

1 Introduction

1.1 Motivation

We live in a data-driven era: the abundance of sensing devices in combination with the plethora of storage capabilities have resulted in a vast collection of high-dimensional data being available in domains as diverse as engineering, astronomy, biology and economics. The analysis of such datasets is often challenging due to the curse of dimensionality: as the signal dimension increases the volume of the space increases fast, resulting in many traditional methods of signal analysis like statistics or distance-based algorithms to fail. However, not all is lost as the data usually exhibits some underlying structure of lower dimensionality. In such cases, not all observed variables are important for understanding the underlying phenomena of interest and the analysis could be significantly facilitated when the signals are transformed into the right, low-dimensional representation.

However, discovering the right signal representation is not an easy task. The requirements are usually many and not always very well defined: we need data representations that disentangle the underlying explanatory factors while being concise and efficient, meaningful and easy to compute. As a result there is a variety of models and methods that can be used each focusing essentially on the application at hand. Possible data models include but are not limited to linear ones, manifolds, graphical models, overcomplete dictionaries, bag-of-words, multi-layer architectures, graphs. Moreover, there is a vast collection of features that can be used to represent and analyze signals like SIFT [79], SURF [6], Haar-like features [118], edge detectors [77] and the list goes on. Many of them, have been used with great success in applications like signal restoration, data visualization, recognition, natural language processing as well as multivariate analysis in social sciences and psychology. Some examples of such applications are shown in Figure 1.1. In particular, in (a) we see the effect of manifold learning when embedding high-dimensional head images to the 2D plane. As we can observe form the resulting mapping, despite the dimensionality reduction, important aspects of the structure of the original image set are preserved and highlighted, as the signals are placed according to the up-down and left-right head pose. Additionally, in (b) we present an example of signal



(a) (b)

Figure 1.1 – Examples of applications of different models to images. In (a) we have the set of views of a 3-D head are embedded into a 2-D plane with the use of manifold learning. The figures are taken from [108]. In (b) we present the denoising of a house image through sparse coding.

denoising achieved by the sparse decomposition of the signal to an appropriate dictionary.

In this thesis, we concentrate on low dimensional structured models and their applications for images. Among the various choices, we focus on two popular and promising models: the manifolds and the sparse signal representations. These two model categories are of very different nature. The manifolds are low dimensional, structured models that can easily express complex variations in signals through their highly non-linear, and usually non-analytic form. On the other hand, sparse representations have a simple, comprehensible linear form as combinations of a few basic features in an dictionary. Although dictionaries can be learned from signals, the resulting features are generally quite simple. As a result, the representations are less structured and far away from modeling complex dependencies among different components of the signals. In this thesis, in an attempt to discover the factors of variation

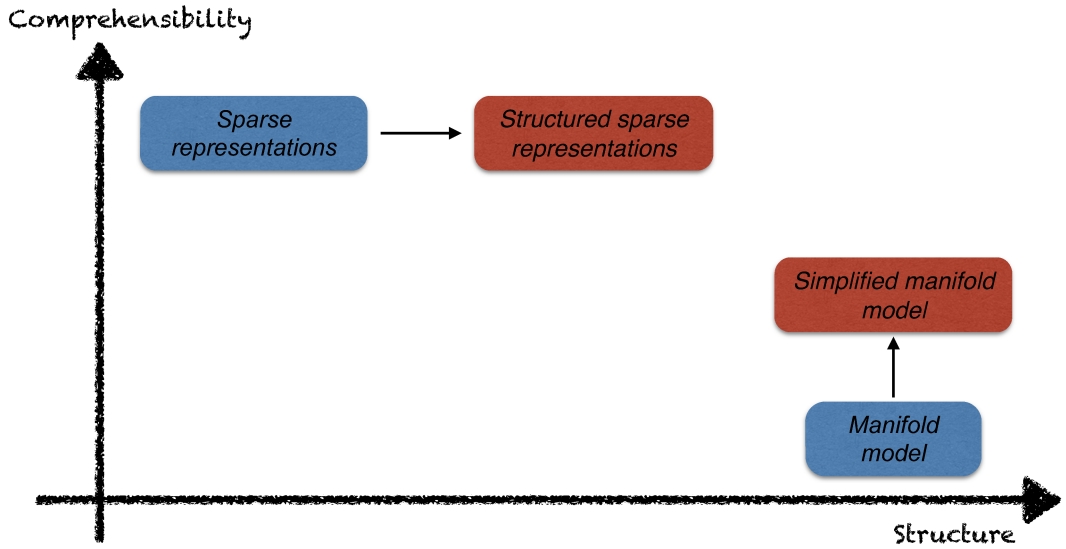


Figure 1.2 – Position of the sparse representation and manifold models on the plane defined by the comprehensibility and structure properties. The horizontal axis expresses how structured a model is and the vertical how comprehensible and simple it is. The sparse representations are positioned in the upper right corner while the manifolds are placed in the lower left corner. The red squares stand for the proposed models in our work, namely a sparsity model with more structure and a simplified manifold model.

and the correlations among them, we target the weaknesses of each model separately namely the non-analytic, incomprehensible manifold form and the unstructured form of sparse representations. In particular, with manifolds we try to uncover novel and simple ways to approximate their form while preserving the model's structure. On the other side, with sparse representations, we follow the opposite direction and we try to introduce more structure into the model in an attempt to better model high level dependencies among the different signal components. At the same time, we also try learn this structure directly from the sparse codes. A diagram of the relative position of the models in terms of comprehensibility and structure is shown in Figure 1.2. From the diagram we see that in our work we try to bridge the gap between the two properties for both manifolds and sparse representations by working on the property that is missing each time, namely structure for sparse codes and comprehensibility for manifolds. A detailed description of the contents of this thesis follows in Section 1.2. Finally, our contributions are highlighted in Section 1.3.

1.2 Thesis outline

The outline of the thesis is as follows. In Chapter 2 we review some popular low-dimensional structure models for signals. We start with the simple case of linear models and its straightforward generalization to the union of subspaces model. Then, we discuss the more generic case of manifolds that can be used to model signals with low-dimensional but non-linear structure.

Next, we review the major advances in sparse and structured sparse signal representations as well as dictionary learning and we conclude the chapter by a brief description of the recently regenerated field of deep architectures.

In Chapter 3, we focus on manifolds. Motivated by their locally linear nature, we employ a set of low dimensional affine models, the flats, to approximate the manifold data while preserving the manifold structure. We model the underlying structure through the neighborhood graph and we use the variance of the local tangents as a measure of linearity. Then, based on elements from the constrained clustering theory, we propose a greedy scheme for partitioning the data into groups that comply with the low dimensionality of the flats. We provide results on both synthetic and real data that show the superior performance of our scheme compared to other state-of-the-art manifold approximation techniques.

In Chapter 4, we combine elements of structured sparse coding and multilevel architectures to propose a new, two-level, structure signal model. In an attempt to better model higher level patterns, we define our structural elements, the molecules, to be linear combinations of atoms from a redundant dictionary. To account for some of the variability of the patterns in real scenarios, we introduce the concept of molecule realizations that permits the prototypes to get signal dependent in slightly deformed versions. We investigate different options for the molecule realizations and we design a novel structured sparse coding scheme adapted to our molecule signal model. Finally, we illustrate the use of our framework with experiments in various applications such as compressed sensing, inpainting and denoising where we observe that our molecule-based representation allows for better reconstruction performance than classical sparsity priors.

Moving to Chapter 5, we address the problem of learning the signal structure. We focus on the signals' sparse codes and we aim at uncovering structured representations for the codes without transforming them back to the signal domain. We use the concepts introduced in the previous chapter for the molecule prototypes and realizations to formulate the structure learning problem directly in the sparse code domain. To solve the learning problem, we propose an alternating optimization algorithm that iterates between steps of code representation and structure update. We test the performance of our scheme on learning the structure of various datasets like synthetic, digit and object images. From our experiments we verify the superior performance of our scheme compared to other existing learning techniques that are however not designed explicitly for the sparse domain.

To conclude the thesis, in Chapter 6 we summarize our findings and we analyze possible future directions and open questions. In particular, we highlight some connections between the structure imposed by the flats in our manifold approximation scheme and the molecule structure we propose for sparse codes. Moreover, we discuss alternative ways to compare the molecule prototypes and the molecule realizations as well as the possibility to learn the dictionary and the molecule structure simultaneously from the data. Finally, we also comment on the possibility to extend the molecule structure model to more than two layers.

1.3 Summary of contributions

In summary, the main contributions of this thesis are the following:

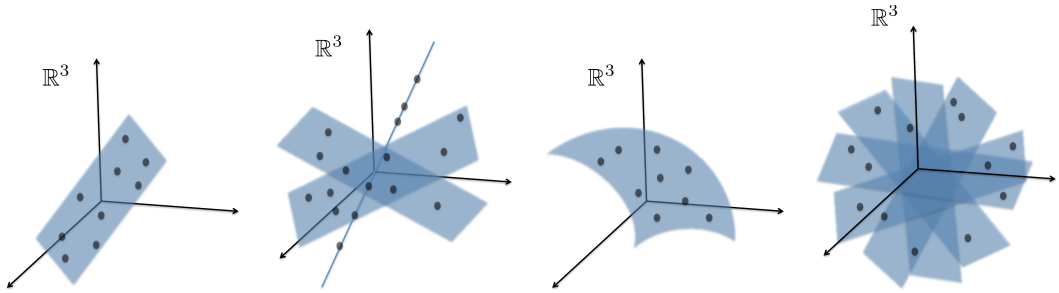
- We design a new manifold approximation scheme that guarantees the preservation of the local linear structure of the manifold data. We highlight the importance of tangent spaces in correctly identifying linear regions on manifolds and we provide theoretical justification for our new clustering algorithm that gathers linear regions into a flat-band approximation.
- We propose a new structured sparsity prior based on molecules that is more informative than traditional approaches. Molecules take into account both the coefficients and the support of the sparse codes and as a result enable the differentiation of structures that share the same support but have distinct energy distributions in their components. Additionally, we propose the new concept of molecule realizations that permits more adaptation to signals, thus retaining some precious flexibility in the data representation.
- We provide a new structural difference measure for molecule prototypes and realizations and we devise a sparse coding scheme that allows the decomposition of signals into molecule realizations according to our novel structure model.
- Based on our new structure model, we formulate the problem of molecule learning directly in the sparse code domain with minimal involvement of the underlying dictionary. We design an efficient algorithm to learn the molecules from the sparse codes by dividing the corresponding complex optimization problem into simpler sub-problems that we carefully analyze and simplify to get an approximate, yet effective solution.

2 State of the art

2.1 Overview

In signal processing and machine learning, the performance of most algorithms is heavily dependent on the representation of the data. As a result, it is quite challenging to uncover the representation that suits best an application. The quest for the best signal representation is of course very old and the related literature is very broad and rich [87, 11, 91]. However, in this thesis we are mostly interested in low-dimensional structured models for data. In other words, we focus on models that assume the existence of a latent structure of lower dimensionality than the original signal space. Very often, such a structure can be represented geometrically. For example, in Figure 2.1 3-D signals exhibit different latent structures: in Figure 2.1a signals fall on a 2-dimensional plane, in Figure 2.1b they come from a union of subspaces composed from 2 planes and a line, in Figure 2.1c, they live on a 2-D manifold embedded in $\mathbb{R}^3$ and finally in 2.1d signals can belong to any plane in the space.

In this chapter, we review the categories of models that are most closely related to the contents of this thesis. In Section 2.2 we review the most popular linear models for data that are still quite successful and insightful in some applications despite their simplicity. Then, in Section 2.3 we consider the more generic case of manifold models and we review the most popular methods for handling data with such a latent structure. In Section 2.4 we present the case of sparsity-based data models. We describe the major directions that have been studied in the field of sparse and structured sparse signal priors as well as the algorithms for learning the corresponding models from the data. Finally, in Section 2.5 we conclude the chapter with a reference to deep architectures, i.e, the family of models composed of multiple similar layers stacked one on top of the other. These representations have become a trend recently a trend in representation learning and some of the proposed models, even though they are not well understood, yet are quite successful in applications such as image recognition and classification.



(a) Linear model. (b) Union of subspaces. (c) Manifold. (d) 2-sparse.

Figure 2.1 – Geometric illustrations of various low-dimensional models.

2.2 Linear models

To start with, we examine the oldest and simplest model for the latent data structure namely a low dimensional, linear or affine subspace. To be more specific, assuming $x \in \mathbb{R}^N$ to be the data vector, this model is described by $x = A * b + \mu$, where $A \in \mathbb{R}^{N \times K}$ is a basis of the underlying subspace and $b, \mu \in \mathbb{R}^K$ are the coefficient vector of x and the offset of the subspace respectively (0 for the linear case). Such an example in $\mathbb{R}^2$ is shown in Figure 2.1a. This is a very standard model and there exist various techniques to uncover the model parameters A, μ with the most popular of them being the Principal Component Analysis (PCA) [121]. When treated as a generative model where b stands for the values of the latent, uncorrelated factors, the problem is called factor analysis (FA) [49, 110] and it has been studied extensively as well. A famous variant of FA is the independent component analysis (ICA) [57] where the factors are further constrained to be independent. These methods have been quite successful in various signal processing tasks like dimensionality reduction [21], blind source separation [9] and multivariate analysis in social sciences [107] and in neuroscience [32].

However, limiting the latent structure to a single subspace is not always the appropriate choice as it can happen that different parts of the data are correlated with different subsets of the underlying factors e.g., video sequences where multiple objects are moving or face images of different people under varying conditions. In such cases, the resulting data clusters around more than one low dimensional subspace $S_i = \{x \in \mathbb{R}^N : x = A_i * b + \mu_i\}, i = 1 \dots p$. Such an example is shown in Figure 2.1b where the data comes from the union of two planes and a line instead of a single plane as in the simple model above. The union of subspaces model is a direct expansion of the linear model and it preserves its simple geometric interpretation while extending its expressive power.

The problem of uncovering a set of appropriate subspaces for a data set X is called subspace clustering or hybrid linear modeling. The objective is usually to cluster the data into groups so that each group can be well represented by a low-dimensional affine space. A common approach is to use an iterative scheme that alternates between a data segmentation step and a subspace estimation step aiming at either minimizing the sum of reconstruction errors [25], [129] or maximizing the likelihood of the data under a probabilistic model, like probabilistic

PCA [15]. Alternatively, different kinds of algebro-geometric approaches have also been proposed. An interesting formulation has been presented in [116], where the problem of subspace clustering is transformed into a problem of fitting and manipulating polynomials. Finally, another alternative is the use of spectral clustering, a technique that requires an affinity matrix which summarizes the similarities between data. Then, the clustering of data into subspaces is usually performed on the eigenvectors of the Laplacian of the affinity matrix. As a result, the choice of the affinity is crucial for the success of the algorithm and is is the most common point of differentiation among the various clustering schemes: in [126] the authors employ the similarity of local tangent spaces, in [29] the polar curvature of subsets of data, in [52] the conic affinity, and in [37, 103] the sparse representation of each data with respect to the rest.

The most algorithms in the field assume that the number of subspaces and their dimensions are known. While some algorithms can provide estimates for these quantities, like a multiscale analysis of the growth rate of the local neighborhoods' eigenvalues [28], these estimates usually come with no theoretical guarantees. Moreover, the model cannot handle cases where the data comes from low-dimensional non-linear subspaces. Nevertheless, the union of subspaces model is a quite popular choice for signals with various applications like motion segmentation [117], face clustering [52] and gene expression analysis [63]. A comprehensive survey on the most popular methods, their applications as well as their limitations can be found in [115].

2.3 Manifold models

While the above models are quite successful when the underlying structure is linear, they fail to properly model data that concentrates near low-dimensional non-linear surfaces instead of linear subspaces, like in Figure 2.1c. In such cases, the data essentially belongs to a manifold model of lower dimensionality embedded in the signal space. It is often the case that the underlying structure of signals of a given family can be described adequately by such a manifold model. For example, prominent examples of image manifolds are the images generated by different views of the same three dimensional object [102, 44] or images of objects belonging to the same category but having different appearances [72]. Some examples of such datasets are shown in Figure 2.2. However, handling manifold models is very challenging as in most of the interesting cases, the manifold's analytic form is not known. In the rest of this section we review the main tools for handling such manifolds.

2.3.1 Manifold approximation

To deal with cases of non-linear structures as opposed to linear ones the idea of principal directions in PCA has been extended to that of principal curves and surfaces for manifold models [50]. Conceptually, principal surfaces are surfaces that pass through the middle of the data distribution. The functions employed to represent the surfaces are of great importance and can take various forms. A common representation consists in expressing them as a

(a) (b)

Figure 2.2 – Examples of images belonging to manifolds. In (a) we show the images of a 3D object from the ALOI dataset [44] taken from different viewpoints. In (b) we have the images of a face from VidTIMIT database [102] as the head performs a rotation to one side.

weighted sum of kernel functions at normally distributed locations in the latent space [16, 86, 43]. Alternatively, such low dimensional encodings of the data can be uncovered with àutoencoders' [51] which are neural net architectures with a small hidden representation layer trained to minimize the reconstruction error. When regularized properly the autoencoders encourage the hidden representations to lie on a low dimensional manifold. An example of such a representation is the contractive autoencoder [95, 96] that penalizes the sensitivity of the architecture to the input so that it ends up modeling an approximation function that locally varies in only a few significant directions in the space. These directions could be considered to be the ones tangent to the underlying manifold at this location.

From another perspective, manifolds are topological spaces that locally resemble an Euclidean space. Therefore, although they might be extremely complicated structures, they have locally, i.e., in the neighborhood of a point, the same characteristics as the usual Euclidean space. Thus, taking into account the locally linear nature of manifolds, the approximating functions can be affine models, each approximating a specific region of the manifold. The affine sub-models can be used further to either devise a global parametrization in a lower dimensional space by proper alignment like in [99, 19] or to approximate the manifold structure in the original space like in [90, 120, 40]. The procedure for learning such models usually resides on alternation between two steps: assigning the data to affine models under proper constraints that express the manifold structure and then updating the affine spaces accordingly.

2.3.2 Manifold embedding

Instead of learning an explicit mapping or an approximating function, a direct embedding of the data could be devised while preserving some important properties of the manifold structure. Tools to achieve this goal are offered by the so called fields of manifold learning and dimensionality reduction. Two pioneer works in the field are the Isomap [108] and the LLE

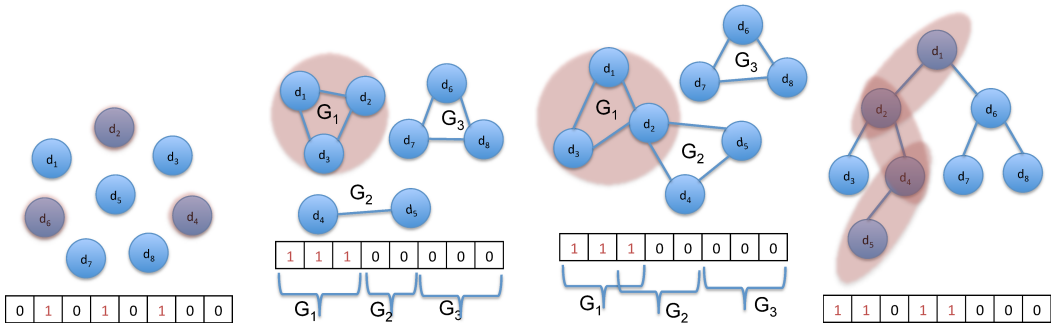
algorithms [100]. In Isomap, a parametrization is found that preserves the geodesic distances between the points while in LLE the focus is on preserving the local linear properties of neighborhoods. Recently, in [37] the neighborhoods and weights have been automatically extracted based on the sparse representations of the points instead of their euclidean geometry. Other well known approaches that aim at preserving local properties of the points' neighborhoods are provided by the Laplacian Eigenmaps (LE) [8] and the Hessian Eigenmaps (HLLE) [34]. These methods have been extended to points lying on Riemmanian manifolds as well [45]. Overall, all these methods fall under the category of spectral dimensionality reduction [12] as they are based on the eigenvalue decomposition of a matrix that encodes the local properties to be preserved in the manifold model.

The manifold learning techniques have found various application like in visualization of image sets [108, 100], tracking [76] and medical image analysis [104]. However, they suffer from some important shortcomings. Since their target is to learn a data embedding and not a mapping function, most of the techniques cannot easily handle out-of-sample data. Moreover, they usually cannot handle properly closed or highly curved manifolds. A detailed list of the most popular algorithms in the field can be found in [114] and [91], along with interesting comments on their strengths and weaknesses.

2.4 Sparse models

Another popular model is the K-sparse signal representation. This model generalizes the linear subspace model by considering again signals with only $K < N$ non-zero coefficients that are however not constrained to live on a specific $K-$dimensional subspace. In this way, the model becomes much more expressive and yet it preserves the low dimensional nature of signals. As in the linear case, sparse signals can be expressed as linear combinations of a set of basis functions, i.e., $x = A * b$ where only K entries in b are non-zero. However, A is no longer of dimension $N \times K$ but of a dimension $N \times L$ with $L >= N$, i.e., it is at least a basis of $\mathbb{R}^N$ but it can also be an overcomplete dictionary. Geometrically, that means that the set of K-sparse signals in A consists of the union of all K-dimensional subspaces in $\mathbb{R}^N$ spanned by vectors in A. A subset of these subspaces with $K = 2$ in $\mathbb{R}^3$ is shown in Figure 2.1d.

Sparsity is a pretty intuitive prior that is also biologically plausible, as shown in the pioneer work of Olshausen and Field [88] where it is suggested that sparsity could be a property employed by the mammalian visual system for achieving efficient representations of natural images. Vast research efforts have been deployed in the last decades in order to design algorithms that solve the hard problem of sparse decomposition of signals by effective approximation [84, 112] or convex relaxation [111, 30]. In such sparse models however, the choice of the underlying dictionary is also important in the success of the model. There exist various predefined dictionaries like waveletes, curvelets and bandelets that are proven to be quite successful in various applications [83]. However, the most recent trend in the field is to learn an adaptive dictionary from the data. The corresponding formulation can be either proba-

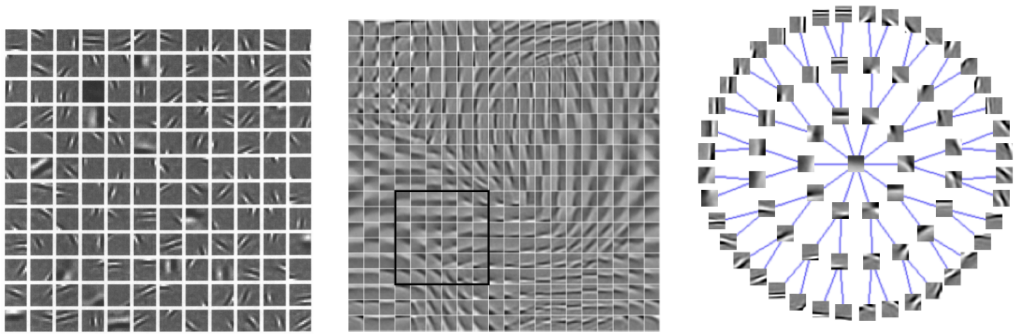


(a) Sparsity (b) Group sparsity (c) Overlapping groups (d) Hierarchical

Figure 2.3 – Examples of priors on sparse models where the atoms are represented as graph nodes and their dependencies with edges. In (a) we have the case of simple sparsity and all atoms are represented as isolated nodes. A sparse code can be any vector with non-zeros in a few of the atoms, like the one shown in (a). Then, in (b) we have the case of non-overlapping groups: the atoms are separated into 3 cliques and the allowed sparse codes have either non-zero entries for all atoms in each group or the whole group is zero. In (c), two of the groups are overlapping: atom d_2 belongs on both G_1 and G_2. Finally, in (d) we have a hierarchical structure in which an atom is allowed to be non-zero in a sparse code iff all its anscestors are non-zero as well.

bilistic [88, 75] or closer to a matrix factorization problem that is usually solved by alternation between steps of sparse coding and dictionary update [38, 1, 73]. Additional constraints can also be imposed like positivity of the sparse codes and the dictionary elements [53]. The sparse model has found quite a few applications notably in compressed sensing [23] and signal and video restoration tasks like denoising and inpainting [82, 92]. When the dictionary learning is enriched with additional discriminative terms, the learned dictionaries can also be used for classification and recognition [80, 56].

While sparsity is a simple and generic model, it is not always a sufficient prior to obtain good signal reconstruction, especially if the original data measurements are compressed or inaccurate. More effective signal models can therefore be built by considering the dependencies between the dictionary elements that appear in the signal representation instead of their mere number. In that spirit, group sparsity has been introduced as a way to enforce a pre-specified structure in the sparse signal decomposition. In particular, the components of the dictionary are partitioned into groups and the elements of each group are encouraged to appear simultaneously in the signal decomposition by an $l_1/\ l_2$ regularization term [125]. Alternatively, the atoms can also obey a predefined hierarchical structure [130]. Other approaches have considered additional flexibility by constraining the signal decomposition to include elements from overlapping groups of atoms [60, 55, 58]. The priors for structured sparsity can also be considered in dictionary learning with interesting results like topographic or hierarchical dictionaries [66, 61]. Such structured sparse models have been shown to improve the prediction performance and the interpretability of the learned models when the imposed structure is relevant [54]. They have been used successful in various applications like image restoration



(a) Sparse (b) Topographic (c) Hierarchical

Figure 2.4 – Examples of dictionaries learned with different constraints for natural image patches. In (a) we have the dictionary learned with traditional l_1 sparse coding from [88]. In (b) we have the topographic dictionary from [66]. During learning atoms are placed in a 2D grid and separated into overlapping neighborhoods which serve as groups for the $l_1 - l_2$ penalty. As a result similar atoms in the dictionary are encouraged to be spatially close in the 2D map. Finally, in (c) we have the hierarchical dictionary from [61].

and topic modelling [60] as well as multi-task learning where they account for shared features among tasks [78]. Some simple examples of the underlying relations between the atoms for different cases of structure are shown in Figure 2.3. Moreover, in Figure 2.4, we show some examples of dictionaries learned for natural image patches with different structure priors.

Although they have been quite successful, these structured priors cannot cover all forms of structure in a dictionary as the form of the dependencies between the atoms is decided a priori. To account for more generic cases of dependencies, some works describe the statistical dependencies between the atoms in probabilistic form with graphical models. For example, Markov Random Fiels (MRFs) are employed for modeling the underlying dependencies in [89, 42, 26] . The resulting structure model is essentially a probability distribution function that compares the different possible supports of atoms in the signal representation. Unfortunately, these models tend to be highly parametric and hard to learn and; as a result they are less popular in data analysis applications.

2.5 Multilevel architectures

To conclude our review on signal structures, we briefly discuss multilevel or, as they are more often called, deep architectures [71]. In such cases, the structure of the model relies in its depth: the model consists of a hierarchy of layers where each layer feeds its output to the next one. The benefits of such architectures over the flat ones has been a subject of research for a long time in the feature extraction and machine learning community. In some cases these benefits have been validated experimentally [59]. The basic motivation is the possible re-usability of the features in the hierarchy along with the potential progressive appearance of more abstract features and disentangled factors at the higher levels.

Among the most popular architectures are the deep autoencoders [51] and the convolutional nets [67, 70, 74, 128]. The deep autoencoders on the one hand are neural nets with many hidden, fully connected layers whose output target is the data input itself. The different hidden layers successively transform the input into a hidden encoding which is then transformed back to its original form at the output. The convolutional nets on the other hand consist of a hierarchy of features that are convoluted with the input at each layer to produce the corresponding feature maps. The features maps can be further processed with pooling and contrast normalization operators before they serve as input to the next layer.

The training of these models consists in finding the weights or equivalently the features in each layer and it can be a very complicated procedure. More importantly, it is not well understood what the deep nets actually learn. Recently, there have been efforts to shed light on this matter by trying to visualize the inputs that maximize the responses in each layer [127, 131] but still the true behavior of deep nets remains a mystery. Nevertheless, deep nets have been applied successfully in image recognition and classification [132, 67] achieving in many cases state-of-the-art results. Finally, an alternative deep architecture, called invariant scattering convolutional network [20], has recently emerged. It uses predefined wavelets as filters, and therefore does not require any learning. Its performance on applications is still being evaluated and the question 'to learn or not to learn' the filters is still unanswered.

To sum up, in this chapter we have described some of the most popular models in representation learning that address the problem of uncovering the low-dimensional, latent structure of signals. While very successful at times, most of them fail to provide representations that are effective, meaningful and efficient at the same time. In the rest of this thesis, we will focus on some specific models and we will present our ideas on how to remedy the aforementioned issues.

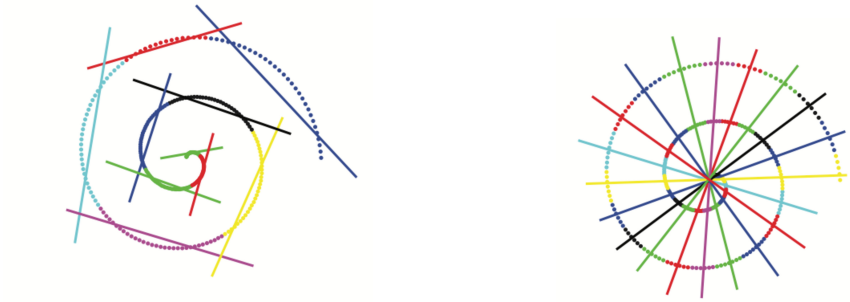
3 Manifold approximation

3.1 Introduction

In this chapter, we focus on manifold models and we try to address their lack of analytic form by devising a manifold approximation scheme based on affine subspaces. Our objective is to uncover a set of low dimensional affine subspaces that represent manifold data accurately while preserving the manifold's structure. We consider d-dimensional, differentiable manifolds that are embedded into a higher dimensional Euclidean space, $\mathbb{R}^N, N >> d$. Intuitively, one can think of a d-dimensional manifold embedded into $\mathbb{R}^N$ as the generalization of a surface in N dimensions: it is a set of points that locally seem to live in $\mathbb{R}^d$ but that macroscopically synthesize a structure living into $\mathbb{R}^N$.

Although manifolds are appealing for effective data representation, their unknown and usually strongly non-linear structure makes their manipulation quite challenging. State-of-the-art techniques in the field of manifold learning try to overcome this issue by inferring a global, data-driven embedding scheme to map the manifold data from the original space to a low-dimensional space. However, it is in general hard to compute a universal manifold representation that is accurate across all manifold areas. Therefore, it is often preferable to employ a set of multiple, simpler structures to approximate locally and in the original space the manifold's geometry. An example of such an approximation for an 1D manifold is shown in Figure 3.1a, where a set of lines approximates the spiral shape.

In this chapter, we employ affine subspaces (flats) to approximate generic manifolds. Such a choice is motivated by the locally linear character of manifolds as well as the simplicity and efficiency of flats for performing local computations, e.g., projections. Data representation with affine models has received quite some attention lately as the popularity of state-of-the-art techniques in subspace clustering has increased. However, these methods apply mainly to cases where data is generated from different low dimensional subspaces that do not necessarily form a manifold. Hence, they uncover a set of linear spaces that do not necessarily comply with the manifold structure, such as the set of lines shown in Figure 3.1b.



(a) (b)

Figure 3.1 – Manifold approximation illustration. On the left, we have an example of a valid approximation by lines of a 1D manifold embedded into $\mathbb{R}^2$. The different colors represent the different groups of samples, each approximated by a line. On the right, we have an example where the approximation does not align well with the manifold structure, as a result of the median k-flats algorithm [25].

Our objective is to compute a set of low dimensional flats that represent the data accurately while preserving the geometry of the underlying manifold. To that end, we formulate the manifold approximation problem as a constrained clustering problem for manifold samples where the constraints are related to the underlying geometry. To represent the manifold structure we use the neighborhood graph of the data samples and we relate the capability of a set of points to be represented by a flat, with the variance of the tangents at these points. As it will be shown in the following sections, this measure emerges naturally from the definition of the local properties of a manifold. Other proposed measures in the literature are more ad-hoc and rely more on the geodesic distances on the manifold as these are computed from the neighborhood graph [120, 40].

Finally, to motivate the use of a greedy scheme for manifold approximation we borrow elements of the constrained clustering theory. The partitioning in our scheme is done in a bottom-up manner where each manifold sample is considered as a different group at the beginning. Groups are then iteratively merged until their number reduces to the desired value. We have tested our algorithm on both synthetic and real data where it gives a superior performance compared to state-of-the-art manifold approximation techniques.

The rest of the chapter is organized as follows. In Section 3.2 we give some mathematical definitions related to manifolds and tangent spaces, which are essential for the work presented in the rest. In Section 3.3, we motivate the use of a greedy strategy with concepts from constrained clustering theory and we present our novel problem formulation for the manifold approximation. We present our approximation algorithm in detail in Section 3.4. In Section 3.5, we describe the experimental setup and the results of our experiments. Finally, in Section 3.6, we provide concluding remarks.

3.2 Preliminaries

In our manifold approximation method we use d-dimensional linear subspaces to approximate the distribution of the data. The linear subspaces as models form themselves a Riemannian manifold as well, which is called the Grassmann manifold. The Grassmann manifold is often referred to when signals are modeled with linear low-dimensional models [48, 120]. Among the various metrics for computing distances between linear subspaces [46], the most natural one is the geodesic distance on the Grassmann manifold. This metric is computed based on the angles between the subspaces. In the rest of this section, we will review some basic definitions necessary for our method, along with the description of proper metrics.

First of all, a set $\mathscr{M} \subseteq \mathbb{R}^N$ is a *d-dimensional differentiable manifold* [105] iff $\forall x \in \mathscr{M}$ there exist open sets $V \in \mathbb{R}^N$ with $x \in V$ and $W \in \mathbb{R}^d$ as well as a one-to-one, differentiable function $f : W \to \mathbb{R}^N$ with continuous inverse such that

$$\begin{aligned} &f(W) = \mathscr{M} \cap V \\ &f^{'}(y) = Df(y), \text{ the Jacobian matrix of } f, \text{ has rank } d, \forall y \in W \end{aligned} \tag{3.1}$$

The function f is called a coordinate system at x. Assuming that $f(a) = x$, the d-rank Jacobian matrix $Df(x)$ and the corresponding linear transformation $f_* : \mathbb{R}^d_a \to \mathbb{R}^N_x$ define a d-dimensional subspace of $\mathbb{R}^N_x$, which is the *tangent space of $\mathscr{M}$ at x* denoted M_x. Instead of working with a set of d-dimensional subspaces that are positioned at point x, it is more convenient to translate all of them to the origin of $\mathbb{R}^N$. For simplicity in the rest of the paper, M_x refers to the tangent space of x translated to the origin of $\mathbb{R}^N$.

After the shifting, the tangent spaces of $\mathscr{M}$ belong to the space of all d-dimensional linear subspaces of $\mathbb{R}^N$; this space is called the Grassmann manifold and it is denoted as $G_{N,d}$ [36]. In $G_{N,d}$, the geodesic distance (arc length) between two subspaces is computed based on their principal angles [122]. In particular, *the principal angles* $0 \le \theta_1 \le \cdots \le \theta_d \le \frac{\pi}{2}$ between two tangents M_x and M_y are defined recursively by:

$$\cos\theta_k = \max_{u_k \in M_x} \max_{v_k \in M_y} u_k^T v_k$$

where $u_k^T u_k = 1$, $v_k^T v_k = 1$ and $u_k^T u_i = 0$, $v_k^T v_i = 0$, $\forall i = 1, \ldots, k-1$. Then, the *geodesic distance* between M_x and M_y is defined as:

$$D_T(M_x, M_y) = \sqrt{\sum_{i=1}^{d} \theta_i^2} = ||\theta||_2 \tag{3.2}$$

where $\theta = \{\theta_1, \ldots, \theta_d\}$ is the vector of the principal angles of M_x and M_y.

Finally, we describe the notion of the mean tangent of a set of samples C_i. To define such a quantity, we can use the generalization of the arithmetic mean to manifolds. To be more

specific, the mean or center of a set C of points in the metric space S (with respect to a distance $\mathscr{D}$) has been given by Karcher in [64] as the element $m_C \in S$ that minimizes the sum of square distances $\mathscr{D}$'s to the points x in the set, i.e.,

$$m_C = \underset{s \in S}{\arg\min} \sum_{x \in C} \mathscr{D}^2(x, s) \tag{3.3}$$

For a set C_i, where each sample $x \in C_i$ has a tangent space M_x. The mean tangent M_{C_i} can be computed using the geodesic distance introduced in Eq. (3.2). Hence, Eq. (3.3) translates into:

$$M_{C_i} = \underset{M \in G_{N,d}}{\arg\min} \sum_{x \in C_i} D_T^2(M_x, M_{C_i}) \tag{3.4}$$

There are several methods that can be used to solve for M_{C_i} in Eq. (3.4). In this work, we have used the algorithm based on singular value decomposition [27].

3.3 Manifold approximation problem

3.3.1 General framework

Equipped with the above definitions, we can now present our problem formulation. We consider the problem of approximating a d-dimensional manifold $\mathscr{M}$, embedded into $\mathbb{R}^N$, with a set of d-dimensional affine subspaces, which we call flats. The dimension d is an external parameter in our problem; in practice it is either specific to the application at hand or estimated a priori from the data. The manifold is represented by the set of samples $\mathscr{X} = \{x_k \in \mathbb{R}^N, k \in [1, n]\}$ and the undirected and symmetric neighborhood graph $G_{\mathscr{X}} = G(\mathscr{X}, E)$, which represents the manifold's geometry by connecting neighbor samples on the manifold. Our objective is to uncover a partition of $\mathscr{X}$ into $\mathscr{L}$ clusters, $\mathbf{C}_{\mathscr{L}}(\mathscr{X}) = \{C_i, i \in [1, \mathscr{L}]\}$, so that each cluster can be well represented by a d-dimensional flat that respects the underlying geometry of the manifold. The number of clusters $\mathscr{L}$ is also specific to the target application; it could also be inferred from the data through an iterative procedure that stops when the approximation error reaches a pre-defined threshold. In this paper, we simply consider that the number of clusters is given as an external parameter to the algorithm.

3.3.2 Feasible partitions

In order for $\mathbf{C}_{\mathscr{L}}(\mathscr{X})$ to be a partition of $\mathscr{X}$, the involved clusters should not overlap and they should cover the whole set $\mathscr{X}$, i.e., $C_j \cap C_i = \emptyset$, $\forall\, i \neq j$ and $\cup_{i=1}^{\mathscr{L}} C_i = \mathscr{X}$. There are many different ways to partition a set into L clusters. However, not all possible partitions of $\mathscr{X}$ are valid in our case since we are interested only in partitions that respect the underlying geometry of the manifold. In particular, we would like to rule out the partitions whose clusters spread over different regions of the manifold even if these clusters can be approximated well with

flats as the flats do not comply with the local manifold structure. An example of such a bad partitioning is illustrated in Figure 3.1b.

In order to check the compliance of a partition $\mathbf{C}_{\mathscr{L}}(\mathscr{X})$ with the manifold's shape we can use the graph $G_{\mathscr{X}}$. Based on the above description, a sufficient condition for a partition to be valid is to be composed of clusters with connected subgraphs. To be more specific, each cluster's subgraph is defined as $G_{C_i} = G_{\mathscr{X}}(C_i, E_i)$ where $E_i = \{a_{ij} \in E : x_i, x_j \in C_i\}$ is the set of edges in E with both endpoints in C_i. Then, the subgraph G_{C_i} is connected if and only if there is a connecting path in E_i for every pair of nodes in C_i.

The set of all partitions that fulfill this condition, i.e., the 'good' partitions, is called the *feasible set of order* $\mathscr{L}$ and denoted by $\Phi_{\mathscr{L}}(\mathscr{X})$. The corresponding feasibility predicate[1], $\Phi_{\mathscr{X}}(\mathbf{C}_{\mathscr{L}}) \equiv \mathbf{C}_{\mathscr{L}} \in \Phi_{\mathscr{L}}(\mathscr{X})$, is then defined as:

$$\Phi_{\mathscr{X}}(\mathbf{C}_{\mathscr{L}}) = \bigwedge_{C_i \in \mathbf{C}_{\mathscr{L}}} \phi(C_i), \text{ where } \phi(C_i) = \begin{cases} \text{true}, & \text{if } G_{C_i} \text{ is connected} \\ \text{false}, & \text{if } G_{C_i} \text{ is not connected,} \end{cases} \tag{3.5}$$

where the symbol $\wedge$ stands for logical addition.

In what follows, we are proposing a bottom-up approach to solve for the best partition $\mathbf{C}_{\mathscr{L}}$. Therefore, we need a rule that permits to merge clusters while preserving the feasibility of the resulting partition. To this end, we define the fusibility predicate $\psi(C_i, C_j)$ that expresses whether two clusters C_i and C_j are 'related', i.e., they could be merged. It is closely related with the feasibility predicate ϕ of Eq. (3.5) by the following property of binary heredity:

$$\text{if } C_i, C_j \neq \emptyset,\ C_i \cap C_j = \emptyset,\ \phi(C_i) \wedge \phi(C_j) \text{ and } \psi(C_i, C_j), \text{ then } \phi(C_i \cup C_j) \tag{3.6}$$

This property means that the fusion of two 'good' and 'related' clusters should give a 'good' cluster. In our case, the 'goodness' of a cluster is defined in (3.5) and is related to the connectivity of the clusters' graph G_C. Therefore, an appropriate choice for the 'related' predicate is to make sure that the graph corresponding to the union of the two clusters is connected. A sufficient condition consists in the presence of an edge between any sample in C_i and any sample in C_j. Therefore, the fusibility predicate becomes

$$\psi(C_i, C_j) = \begin{cases} \text{true}, & \text{if } C_i, C_j \text{ have an edge connecting them} \\ \text{false}, & \text{otherwise.} \end{cases} \tag{3.7}$$

3.3.3 Evaluation of feasible partitions

Equipped with the definition of feasible partitions and with a method to create new feasible partitions from existing ones through merging fusible clusters, we now define a way to evaluate the effectiveness of a feasible partition in capturing the manifold's local geometry. We first

[1] The term 'predicate' is used to refer to boolean valued functions.

need a criterion function P that is non-negative, distributive over the clusters in $\mathbf{C}$ and zero for the case of single-element clusters, i.e.,

$$P(\mathbf{C}) = \sum_{C_i \in \mathbf{C}} p(C_i) \text{ with } p(C_i) \geq 0 \text{ and } p(\{x\}) = 0, \forall x \in \mathscr{X}. \tag{3.8}$$

The function $p(C_i)$, which represents the distribution of P over the clusters in a partition, has to be non-negative for all clusters and zero for single-element clusters. In our case, the goal is to uncover clusters that can be well-represented by d-dimensional flats; therefore the function p should be measuring how well the points in the corresponding cluster can be represented by a linear d-dimensional space.

From the definition of manifolds in Section 3.2, we can observe that the regions of the manifold that can be well represented by linear d-dimensional spaces are the ones for which the function f is linear. In such a case, we have the Jacobian matrices $Df(a) = Df(b)$, $\forall a, b \in W$, which means that the tangent spaces of all points $x \in \mathscr{M} \cap V$ coincide when they are seen as subspaces in $\mathbb{R}^N$. Therefore, an appropriate measure of the linearity of a manifold region is the variability of the tangent spaces in it. Hence, we introduce a variance-based criterion function $p(C_i)$ that measures the variance of the tangents of the samples in a cluster C_i, i.e.,

$$p(C_i) = \sum_{x \in C_i} D_T^2(M_{C_i}, M_x) \tag{3.9}$$

where M_{C_i} is the mean tangent over the tangents of the samples in C_i and D_T is the geodesic distance on the Grassman manifold given in Eq.(3.2) .

3.3.4 Problem formulation

We now formalize our manifold approximation objective as the problem of finding the feasible partition $\mathbf{C}^*_{\mathscr{L}}(\mathscr{X})$ that minimizes P, i.e.,

$$\mathbf{C}^*_{\mathscr{L}}(\mathscr{X}) = \underset{\mathbf{C} \in \Phi_{\mathscr{L}}(\mathscr{X})}{\arg\min}\, P(C) = \underset{\mathbf{C} \in \Phi_{\mathscr{L}}(\mathscr{X})}{\arg\min} \sum_{C_i \in \mathbf{C}} p(C_i) \tag{3.10}$$

where the criterion function $p(C_i)$ is given in (3.10) and $\Phi_{\mathscr{L}}$ is defined in (3.5). The problem of Eq. (3.10) can be solved with dynamic programming, i.e. by incrementally creating the optimal partitions of different sizes starting with size 1 and exploring all possible ways to scale up. To be more specific, from [4], the constrained clustering problem of Eq. (3.10) can be expressed with the generalized Jensen equality [62]:

$$\mathbf{C}^*_{\mathscr{L}}(\mathscr{X}) = \begin{cases} \{\mathscr{X}\}, & \mathscr{L} = 1 \\ \mathbf{C}^*_{\mathscr{L}-1}(\mathscr{X} \setminus C^*) \cup \{C^*\}, & \mathscr{L} > 1 \end{cases} \quad \text{where} \tag{3.11}$$

$$C^* = \underset{\substack{\emptyset \subset C \subset \mathscr{X} \\ \exists \mathbf{C} \in \Phi_{\mathscr{L}-1}(\mathscr{X} \setminus C): \mathbf{C} \cup \{C\} \in \Phi_{\mathscr{L}}(\mathscr{X})}}{\arg\min} (P(\mathscr{X} \setminus C) + p(C)) \tag{3.12}$$

The symbol \ stands for set subtraction and ∪ for set addition. This is a dynamic programming equation that may lead to polynomial time solutions under certain constraints, depending on the characteristics of the clustering problem [5]. However, in the general case, this approach gives rise to algorithms that have exponential time complexity. An alternative way of solving problems of the form of Eq. (3.10) is presented in [3]. It allows for more efficient, but less accurate algorithms as it proposes the use of a greedy framework instead of the dynamic programming one. We opt for such an alternative approach for solving the problem in Eq. (3.10).

In order to get to our greedy framework, we need a measure for comparing clusters and deciding on proper merging choices. Thus, we define the dissimilarity measure $d : (C_i, C_j) \rightarrow \mathbb{R}_0^+$ as the difference in the criterion function before and after the merging of two clusters, i.e.,

$$d(C_i, C_j) = p(C_i \cup C_j) - p(C_i) - p(C_j) = \sum_{x \in C_i \cup C_j} D_T^2(M_x, M_{C_i \cup C_j}) - \sum_{x \in C_i} D_T^2(M_x, M_{C_i}) \quad (3.13)$$

where D_T is the geodesic distance on the Grassman manifold and M_{C_i} is the mean tangent of cluster C_i. We assume that the merging of any two fusible clusters always gives rise to a cluster with a higher score in terms of the criterion function. Under some mild assumptions on the relations between P, d and Φ [3], we can now rewrite Eq. (3.10) as

$$\mathbf{C}^*_{\mathscr{L}}(\mathscr{X}) = \left(\mathbf{C}^{'}{}_{\mathscr{L}+1}(\mathscr{X}) \setminus \{C_i^{'}, C_j^{'}\}\right) \cup \{C_i^{'} \cup C_j^{'}\} \qquad \text{where} \quad (3.14)$$

$$(\mathbf{C}^{'}{}_{\mathscr{L}+1}(\mathscr{X}), C_i^{'}, C_j^{'}) = \underset{\substack{C_i, C_j \in \mathbf{C} \\ \mathbf{C} \in \Phi_{\mathscr{L}+1} \\ \psi(C_i, C_j) \text{ is true}}}{\arg\min} \quad (P(\mathbf{C}) + d(C_i, C_j))$$

This equation still suggests a dynamic programming solution. The difference with Eq. (3.11) is that in Eq. (3.14) we move from higher values of L to lower ones, i.e., in order to find the best partition of size $\mathscr{L}$, we check all partitions of size $\mathscr{L}+1$ for the pair of fusible clusters that can be merged with the minimum cost.

From Eq. (3.14), it is now straightforward to derive a greedy approximation strategy for the clustering problem by eliminating the search over the set $\Phi_{\mathscr{L}+1}$, i.e.,

$$\hat{\mathbf{C}}_{\mathscr{L}}(\mathscr{X}) = \left(\hat{\mathbf{C}}_{\mathscr{L}+1}(\mathscr{X}) \setminus \{C_i^{'}, C_j^{'}\}\right) \cup \{C_i^{'} \cup C_j^{'}\} \qquad \text{where} \quad (3.15)$$

$$(C_i^{'}, C_j^{'}) = \underset{\substack{C_i, C_j \in \hat{\mathbf{C}}_{\mathscr{L}+1}(\mathscr{X}) \\ \psi(C_i, C_j) \text{ is true}}}{\arg\min} \quad d(C_i, C_j)$$

With this approach, we reduce significantly the computational complexity of the scheme, as we don't perform an exhaustive search over all possible partitions of size $\mathscr{L}+1$ anymore. Instead, we rely on one partition of size $\mathscr{L}+1$, the $\hat{\mathbf{C}}_{\mathscr{L}+1}(\mathscr{X})$, and perform a search over all fusible pairs of clusters in this one. However, as it is often the case with greedy strategies, we

cannot guarantee the optimality of the resulting partitions $\hat{\mathbf{C}}_{\mathscr{L}}(\mathscr{X})$ anymore.

3.4 Greedy cluster merging for locally linear approximation

Following the greedy strategy that is introduced in the previous section, our manifold approximation algorithm is based on grouping the manifold samples $\mathscr{X}$ according to local tangent spaces, in order to minimize the cost function in Eq. (3.10) and to preserve the manifold geometry. Our method is divided in two main steps. First, we perform the necessary preprocessing steps on the samples in order to compute the graph $G_{\mathscr{X}}$ and the tangent spaces M_x. Second, we use the graph $G_{\mathscr{X}}$ and the tangent spaces M_x's to greedily merge the samples into clusters according to Eq. (3.15) until we reach a feasible partition with $\mathscr{L}$ components. The block diagram of the method is presented in Figure 3.2 .

3.4.1 Tangent space

In the first step of the algorithm, our objective is to compute the neighborhood graph $G_{\mathscr{X}}$ and the local tangent space M_x for each sample $x \in \mathscr{X}$. There exist various ways to construct $G_{\mathscr{X}}$. We have chosen to use the simplest one, namely the k-nearest neighbor approach, i.e., we connect each sample in $\mathscr{X}$ with its k-nearest neighbors. The resulting graph $G_{\mathscr{X}}$ is assumed to be undirected and symmetric. For each sample x we can then define a neighborhood $N_x = \{y \in \mathscr{X} : (x, y) \in E\}$ as the set of samples that are connected to x by an edge in $G_{\mathscr{X}}$. Then, we can approximate the tangent space at x by the d-dimensional subspace of $\mathbb{R}^N$ that best approximates the data in N_x. Equivalently, we compute M_x as the d-dimensional subspace of $\mathbb{R}^N$ that best approximates the neighborhood N_x^0 i.e., N_x shifted to the origin[1]. In other words, we need to compute the best d-rank approximation of the data matrix corresponding to N_x^0, denoted as $[N_x^0]$. Based on Eckart-Young theorem [35], this approximation is equal to the d-rank SVD of $[N_x^0]$. Therefore, the tangent space M_x corresponds to the subspace spanned by the left eigenvectors of the d dominant singular values of $[N_x^0]$.

The first step of our scheme is not the main focus of our work. Its purpose is to infer the local geometry of the manifold and as such can be replaced by any other algorithm that achieves the same objective. We have made simple choices, namely a k-nearest neighbor algorithm for representing the local manifold geometry and SVD decomposition for tangent computation. Our goal is to show that tangent space information can be effective in manifold approximation even when tangents are computed with naive techniques. More sophisticated techniques can be used for tangent computation e.g. [37], [123], as any further improvement of this step can only benefit the overall algorithm. For example, one can account for the noise in the data using the method shown in 3.3. After computing the tangents with the process described above, an additional improvement step is performed by averaging neighboring tangents. In

[1] We apply a shift operator $T_{\vec{x}}$ to the whole neighborhood N_x, where $\vec{x}$ is the vector corresponding to the sample x in $\mathbb{R}^N$. This operator moves x to the origin and brings along all its neighborhood, while preserving all distances in it.

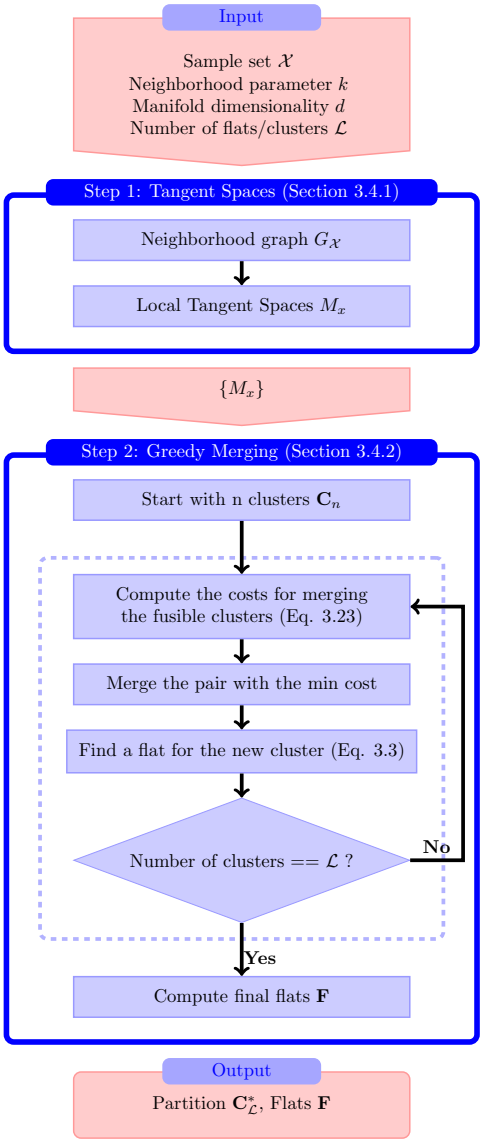
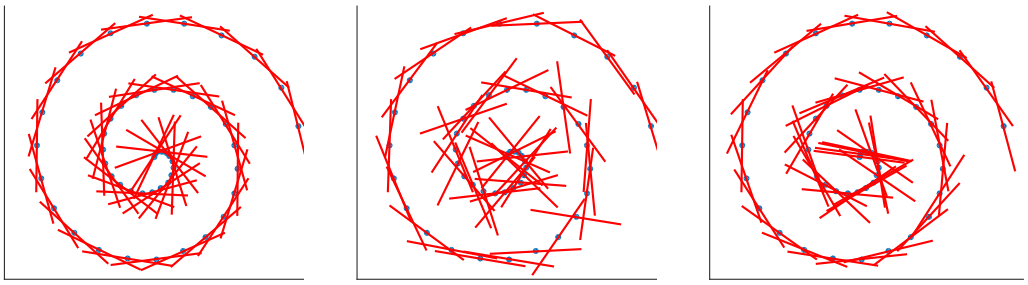


Figure 3.2 – The block diagram of the system.

this way, the final tangents are smoothed and the effect of noise is almost cancelled. However, such alternative solutions for tangent space computation go beyond the scope of this work.

3.4.2 Greedy merging

Once the graph and the tangent spaces have been computed, we proceed with solving the optimization problem presented in Eq. (3.10). In order to minimize the cost function, we follow the method presented in Eq. (3.15). We start with $n = |\mathcal{X}|$ separate clusters, one for each sample. This is the optimal partition for n clusters, i.e., $\mathbf{C}^*_n = \{\{x\}, x \in \mathcal{X}\}$. Then, we reduce the number of clusters iteratively, by merging the clusters C_i and C_j with the minimum



(a) (b) (c)

Figure 3.3 – An example of smoothing of the tangents in case of noisy data. In (a) we have the tangents computed based on the original data and in (b) the corresponding tangents in case of noisy data. Then, in (c) we see the result of the smoothing. As we can observe, the smoothing process (averaging in this case) improves significantly the appearance the computed tangents, resulting in almost removing the side effects of noise.

dissimilarity, until we reach the desired number of clusters $\mathscr{L}$.

At each iteration, there exists a set of possible mergings between the clusters in $\mathbf{C}$. The fusibility predicate, given in Eq. (3.6), defines the sufficient condition for a pair of clusters to be fusible: any cluster C_i can be merged with any of its neighbors, i.e., the set $NG_{C_i} = \{C_j : \exists x \in C_i, \exists y \in C_j \text{ s.t } (x,y) \in E\}$. The dissimilarity between C_i and $C_j \in NG_{C_i}$ is given by Eq. (3.13) as

$$\begin{aligned} d(C_i,C_j) &= \sum_{x\in C_i\cup C_j} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_i} D_T^2(M_x, M_{C_i}) - \sum_{x\in C_j} D_T^2(M_x, M_{C_j}) \\ &= \sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_i} D_T^2(M_x, M_{C_i}) \\ &+ \sum_{x\in C_j} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_j} D_T^2(M_x, M_{C_j}) \end{aligned} \tag{3.16}$$

Note that, since M_{C_i} and M_{C_j} are respectively the mean tangents of C_i and C_j, each of them is the subspace that minimizes the sum of the square distances from the tangents in each cluster (see Eq. (3.3)). As a result, $M_{C_i\cup C_j}$ can only produce the same or a higher value than M_{C_i} when measuring the sum of square distances from the tangents in C_i. In other words, $\sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j})$ is greater or equal to $\sum_{x\in C_i} D_T^2(M_x, M_{C_i})$. The same holds for the cluster C_j. Therefore, $d(C_i,C_j)$ is always non-negative.

Unfortunately, it is costly to compute Eq. (3.16) for all feasible mergings as it requires the computation of the mean tangent for all possible merged clusters. We would rather use a measure that depends only on the information that is already available to the algorithm, i.e., the means of the clusters that we have computed so far and their distances to the tangents in their clusters. Moreover, since we are using a greedy bottom-up approach with an initial cost equal to zero, we have to ensure that, at each iteration of the algorithm, the chosen merging does only marginally increase the overall cost. Therefore, an upper bound for $d(C_i,C_j)$ that

depends only on the means of the existing clusters would be a suitable approximate dissimilarity measure $\tilde{d}(C_i, C_j)$ for our algorithm. It would contribute in reducing the complexity of the algorithm by limiting the amount of necessary computations at each iteration.

In order to compute our approximate measure $\tilde{d}(C_i, C_j)$, we need to perform a series of steps. First, we observe that:

$$\sum_{x \in C_i} D_T^2(M_x, M_{C_i \cup C_j}) \leq \sum_{x \in C_i} D_T^2(M_x, M_{C_j}), \tag{3.17}$$

which means that the mean tangent of $C_i \cup C_j$ is closer to the mean tangent of C_i than the mean tangent of C_j. This statement, which also holds if we interchange the clusters C_i and C_j, is inevitably true. Indeed, by contradiction, if $\sum_{x \in C_i} D_T^2(M_x, M_{C_i \cup C_j})$ is larger than $\sum_{x \in C_i} D_T^2(M_x, M_{C_j})$, then $\sum_{x \in C_i \cup C_j} D_T^2(M_x, M_{C_i \cup C_j})$ is also strictly larger than $\sum_{x \in C_i \cup C_j} D_T^2(M_x, M_{C_j})$. But, this contradicts the optimal character of $M_{C_i \cup C_j}$ for representing $C_i \cup C_j$ in terms of the geodesic distance.

Then, by substituting Eq. (3.17), and its equivalent form for C_j in Eq. (3.16), we have:

$$\begin{aligned} d(C_i, C_j) \leq & \sum_{x \in C_i} \left[D_T^2(M_x, M_{C_j}) - D_T^2(M_x, M_{C_i}) \right] \\ & + \sum_{x \in C_j} \left[D_T^2(M_x, M_{C_i}) - D_T^2(M_x, M_{C_j}) \right] \end{aligned} \tag{3.18}$$

Moreover, by the triangle inequality:

$$D_T(M_x, M_{C_i}) \leq D_T(M_x, M_{C_j}) + D_T(M_{C_i}, M_{C_j}), \qquad \forall x \in \mathscr{X} \tag{3.19}$$

$$D_T(M_x, M_{C_j}) \leq D_T(M_x, M_{C_i}) + D_T(M_{C_i}, M_{C_j}), \qquad \forall x \in \mathscr{X} \tag{3.20}$$

Taking the square of these inequalities and summing over C_j and C_i respectively we get:

$$\begin{aligned} & \sum_{x \in C_j} \left[D_T^2(M_x, M_{C_i}) - D_T^2(M_x, M_{C_j}) \right] \leq \\ & \qquad 2 D_T(M_{C_i}, M_{C_j}) \sum_{x \in C_j} D_T(M_x, M_{C_j}) + |C_j| D_T^2(M_{C_i}, M_{C_j}) \\ & \sum_{x \in C_i} \left[D_T^2(M_x, M_{C_j}) - D_T^2(M_x, M_{C_i}) \right] \leq \\ & \qquad 2 D_T(M_{C_i}, M_{C_j}) \sum_{x \in C_i} D_T(M_x, M_{C_i}) + |C_i| D_T^2(M_{C_i}, M_{C_j}) \end{aligned} \tag{3.21}$$

Substituting Eq. (3.21) in Eq. (3.18) we finally have the following upper bound for the dissimilarity measure:

$$\begin{aligned} d(C_i, C_j) \leq & (|C_i| + |C_j|) D_T^2(M_{C_i}, M_{C_j}) \\ & + 2 D_T(M_{C_i}, M_{C_j}) \left[\sum_{x \in C_i} D_T(M_x, M_{C_i}) + \sum_{x \in C_j} D_T(M_x, M_{C_j}) \right], \end{aligned} \tag{3.22}$$

which depends only on pre-computed information. Therefore we can define our approximate dissimilarity measure $\tilde{d}(C_i, C_j)$ as

$$\tilde{d}(C_i, C_j) = (|C_i| + |C_j|)D_T^2(M_{C_i}, M_{C_j}) + 2D_T(M_{C_i}, M_{C_j})\left[\sum_{x \in C_i} D_T(M_x, M_{C_i}) + \sum_{x \in C_j} D_T(M_x, M_{C_j})\right], \tag{3.23}$$

By comparing Eq. (3.23) with Eq. (3.16), we can observe that Eq. (3.23) is indeed more computationally efficient as it involves only the means of the existing clusters and not those of the clusters after merging the fusible pairs. In our algorithm, the costs for all possible mergings at each iteration are thus computed according to Eq. (3.23). The clusters with the minimum estimated merging cost are then combined and the mean of the newly formed cluster is computed as shown in Section 3.3.3. The procedure is then repeated until we reach the desired number of clusters $\mathscr{L}$.

At the end, each cluster represents a group of samples that can be well approximated by a d-dimensional flat. We compute the final flats for each cluster and we use the subspace spanned by the left eigenvectors corresponding to the d dominant singular values of each cluster's data matrix as representative subspace. The overall manifold approximation algorithm is summarized in Algorithm 1.

Algorithm 1 Agglomerative clustering based on differences of tangents (ACDT)

Input: $\mathscr{X}, k, \mathscr{L}, d$
Construct $G(\mathscr{X}, E)$ ▷ *Step 1, Section 3.4.1*
for all $x \in \mathscr{X}$ **do**
  $N_x = \{y \in \mathscr{X} : (x, y) \in E\}$ ▷ *Compute neighborhoods*
  $[N_x^0] = USV^T, M_x = U$ ▷ *Compute tangent spaces*
end for
$n = |\mathscr{X}|$, $\lambda = 0$, $\mathbf{C}_n^* = \{\{x\} : x \in \mathscr{X}\}$ ▷ *Step 2, Section 3.4.2*
for $\lambda < n - \mathscr{L}$ **do**
  $(C_i^{'}, C_j^{'}) = \underset{\substack{C_i, C_j \in \mathbf{C}_{n-\lambda}^* \\ \psi(C_i, C_j) \text{ is true}}}{\arg\min} \tilde{d}(C_i, C_j)$ ▷ *Eq.* (3.23)
  $\mathbf{C}_{n-\lambda+1}^* = (\mathbf{C}_{n-\lambda}^* \setminus \{C_i^{'}, C_j^{'}\}) \cup \{C_i^{'} \cup C_j^{'}\}$
  Compute $M_{C_i^{'} \cup C_j^{'}}$ ▷ *Eq.* (3.4)
  $\lambda = \lambda + 1$
end for
for $C_i \in \mathbf{C}_{\mathscr{L}}^*$ **do** ▷ *Compute the final flats* F_i [2]
  $[C_{m_i}^0] = USV^T$
  $F_i = U$
end for
Output: $\mathbf{C}_{\mathscr{L}}^*, \mathbf{F}$

[2] m_i is the sample mean of C_i, $[C_{m_i}^0]$ is the data matrix formed by the samples in C_i shifted by m_i and U, S, V are the results of its d-rank SVD.

3.4.3 Computational complexity

We now analyze and compare briefly the complexity of both versions of the manifold approximation algorithm, the one using the exact dissimilarity measure of Eq. (3.16) and the other using the approximate measure of Eq. (3.23). The preprocessing step is the same for both schemes and it is skipped in the following analysis. Then, the operations that are time consuming in our scheme are the tangent distance computations and the computation of mean tangents. In the following we will consider that both have similar computational costs.

Computing the cost of a possible merging with Eq. (3.23) requires only the computation of one additional tangent distance at each step. Denoting by $K_{n-\lambda}$ the number of possible mergings in the clustering $\mathbf{C}_{n-\lambda}$ at step λ, the complexity of one step of the greedy merging (line 9 in Algorithm 1) requires $K_{n-\lambda}$ computations of tangent distances. Then, the operation at line 11 also requires one mean tangent computation plus $|C_i^{'} \cup C_j^{'}|$ tangent distance computations for the newly formed cluster. Therefore, the greedy merging (lines 8-12 in Algorithm 1) will be performed with a time complexity of $T_{approx}(n) = O\left(\sum_{\lambda=1}^{n-\mathscr{L}}(1+|C_i^{'} \cup C_j^{'}| + K_{n-\lambda})\right)$ where n is the number of data samples.

On the other hand, if the exact dissimilarity measure was used, Eq. (3.16) would require one mean tangent computation plus $|C_i^{'} \cup C_j^{'}|$ tangent distance computations for every possible merging. Then, after picking the winning merging, no further actions would be required. In total, the scheme would have a time complexity of $T_{exact}(n) = O\left(\sum_{\lambda=1}^{n-\mathscr{L}}(1+|C_i^{'} \cup C_j^{'}|)K_{n-\lambda})\right)$.

To complete our analysis, we need to estimate the number of possible mergings $K_{n-\lambda}$. Since, at each step of the algorithm, we perform one merging operation, we will have exactly $n-\lambda$ clusters at step λ. Moreover, each $C_i \in \mathbf{C}_{n-\lambda}$ will have on average a size equal to $|\tilde{C}_i| = \dfrac{n}{|\mathbf{C}_{n-\lambda}|} = \dfrac{n}{n-\lambda}$ and therefore we have that C_i has at most $k\dfrac{n}{n-\lambda}$ different neighbors. Thus, the number of possible mergings is at most $K_{n-\lambda} \leq \dfrac{1}{2}|\mathbf{C}_{n-\lambda}|k\dfrac{n}{n-\lambda} = \dfrac{1}{2}kn$.

By substituting $K_{n-\lambda}$ from above and $|C_i^{'} \cup C_j^{'}|$ with its average in $T_{approx}(n)$ we get:

$$T_{approx}(n) = O\left(\sum_{\lambda=1}^{n-\mathscr{L}}\left(1+2*\frac{n}{n-\lambda}+\frac{1}{2}kn\right)\right) = O\left(\sum_{\lambda=1}^{n-\mathscr{L}}\frac{1}{2}kn\right) = O\left(n^2\right) \tag{3.24}$$

For the exact dissimilarity measure we have:

$$T_{exact}(n) = O\left(\sum_{\lambda=1}^{n-\mathscr{L}}\left(1+\frac{n}{n-\lambda}kn\right)\right) = O\left(kn^2(H_{n-1}-H_{\mathscr{L}-1})\right) = O\left(n^2\ln n\right) \tag{3.25}$$

which is higher than the average running time of our approximate algorithm[1]. Therefore, we can clearly see that the use of the approximate dissimilarity measure is beneficial for the

[1] $H_n, H_{\mathscr{L}}$ are the harmonic numbers of order n and $\mathscr{L}$ respectively

computational complexity of the algorithm.

3.5 Experimental results

To study the performance of our manifold approximation scheme, we have tested it for both synthetic and real datasets. We have compared our scheme (ACDT) with two other manifold approximation approaches from the literature, namely the Hierarchical Divisive Clustering (HDC) [120] and the Hierarchical Agglomerative Clustering (HAC) [40]. The HDC algorithm starts with considering all the data as one cluster and then hierarchically partitions them by dividing highly non-linear clusters. As a linearity measure, it uses the deviation between the Euclidean and geodesic distances, i.e., each cluster gets a nonlinearity score that is equal to the average ratio of the geodesic distance over the Euclidean one for all the pairs of samples in the cluster. The process continues until all existing clusters have a nonlinearity score that is lower than a given threshold. On the other hand, HAC is a bottom-up algorithm, i.e., each sample is considered at the beginning as a separate cluster and then clusters are merged iteratively until their number reduces to the desired target. At each iteration of the algorithm, the pair of clusters with the minimum distance is merged. The distance between two clusters is measured as the average geodesic distance between the samples of the one cluster and the samples of the other. Our scheme follows also a bottom-up strategy; however our distance measure is completely different than the one in [40]. Instead of relating our merging decisions to the average geodesic distances, we use the variance of tangents to decide on proper mergings. This choice has been motivated by the definition of tangents as the best locally linear approximations of manifolds and has been proven very effective in practice.

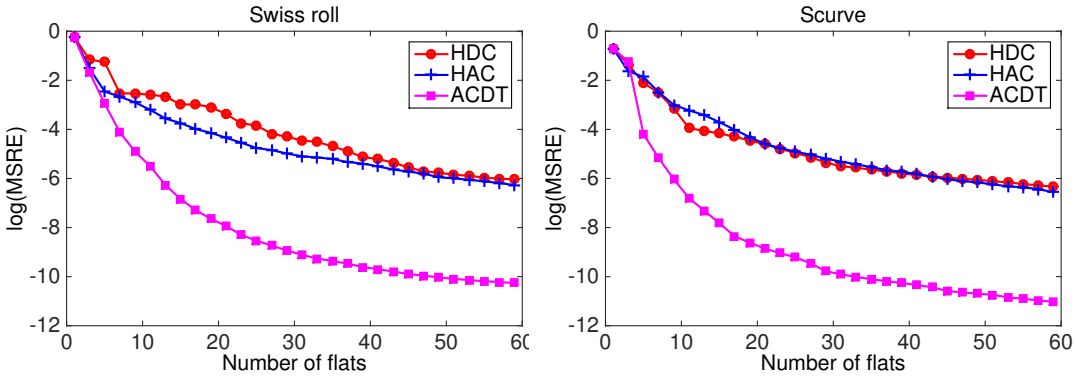
In order to quantify the performance of the compared schemes, we have used the mean squared reconstruction error (MSRE). The MSRE is defined as

$$MSRE = \frac{1}{N}\sum_{i=1}^{N} ||x_i - \hat{x}_i||^2$$

where x_i and $\hat{x_i}$ are respectively a data sample and its projection on the corresponding approximating flat, while N is the total number of signals. For the HDC and HAC algorithms, whose output is a set of clusters and not a set of representative flats, we have computed the corresponding flats by principal component analysis on the data of each cluster. The results of our experiments for all three algorithms are given below for three different datasets.

3.5.1 Synthetic Data

Firstly, we test the performance of our scheme in approximating synthetic manifolds. We use the Swiss roll and the S-curve dataset. The training set for both cases consists of 5000 points, randomly sampled from the manifolds. The neighborhood size k is set equal to 15 in the experiments. We have observed that it is preferable to use low values for k, varying from



(a) Swiss roll (b) S-curve

Figure 3.4 – Mean squared reconstruction error (MSRE) versus the number of flats. The error on the y-axis is shown in logarithmic scale.

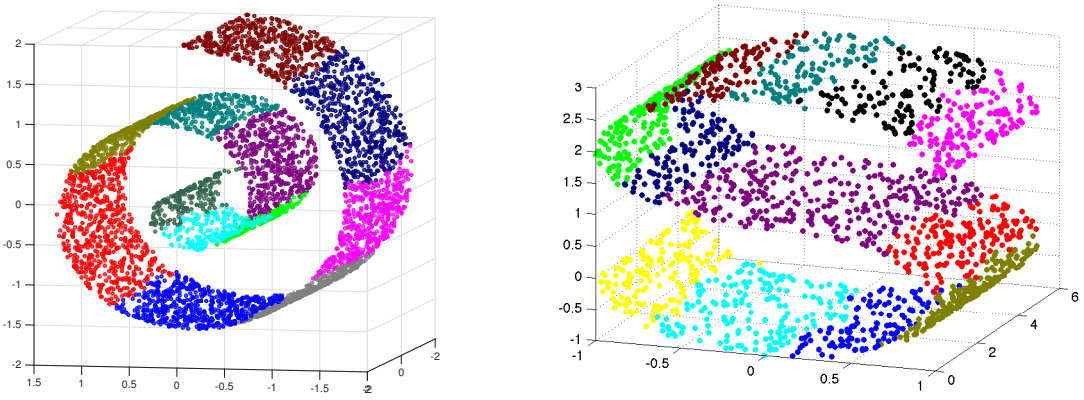
0.5% to 2% of the total number of samples, in order to avoid "short-circuit" effects that distort the manifold structure.

The MSRE versus the number of flats, for our synthetic manifolds, is presented in Figure 3.4. The results are averaged over 10 randomly chosen training sets. From Figure 3.4, we can see that our scheme approximates better the manifold structure than the other approaches. The approximation performance is better even for a small number of flats but the differences are more evident in the mid-range cases where the number of flats is between 15 and 30. For higher number of flats, the difference stabilizes around 50 to 60 flats when the MSREs of the algorithms converge. The effectiveness of our method is mainly due to the use of the difference of tangents for measuring the linearity of sample sets instead of the geodesic-based criteria used by other algorithms [120, 40].

For the sake of completeness, we also give in Table 3.1 the running times for the three algorithms in the case of 60 flats. We can see that the two bottom-up schemes are a bit penalized in terms of complexity as they start with a high number of clusters (equal to the number of points) and proceed with mergings until they reach the desired number of clusters, which is significantly smaller in this experiment. On the other side, HDC has to perform fewer splittings, as it starts with considering all points as one cluster. As far as ACDT is concerned, we would like to note that there is still room for improvement as the code used is far from optimized. For example, a significant gain could be achieved by optimizing the SVD computations but this is beyond the scope of our paper.

	ACDT	HDC	HAC
Running time (sec)	389	15	288

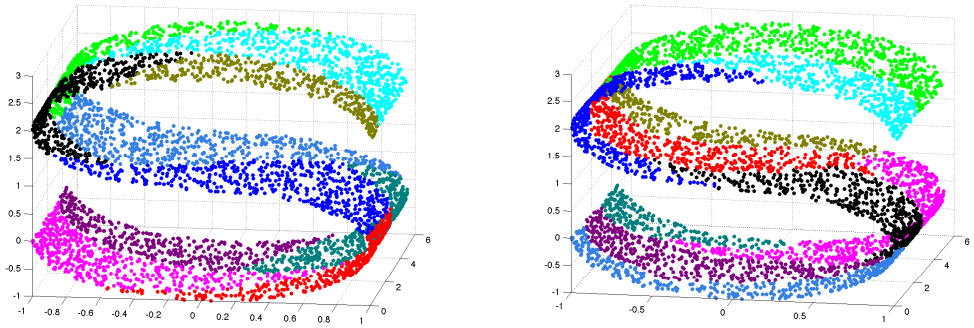
Table 3.1 – Running times for the three algorithms in case of the Swiss role data and 60 flats. The results were obtained on an Intel Core Duo 2.66 GHz, MacBook Pro.



(a) Swiss roll

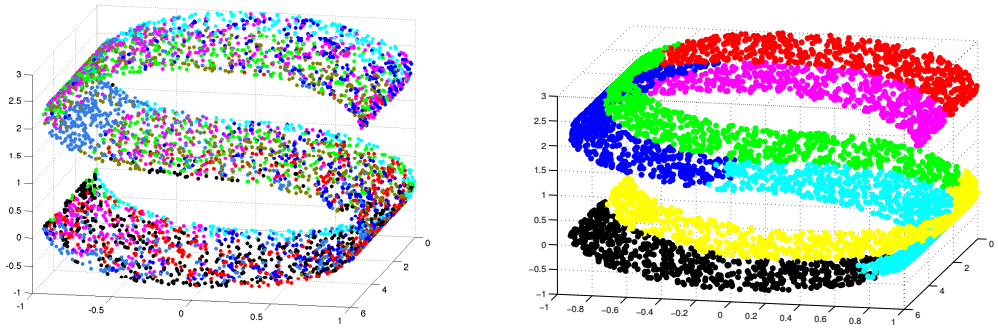
(b) S-curve

Figure 3.5 – The final groups formed by the proposed approximation algorithm with 12 flats. Each color represents a different cluster of points.



(a) HDC

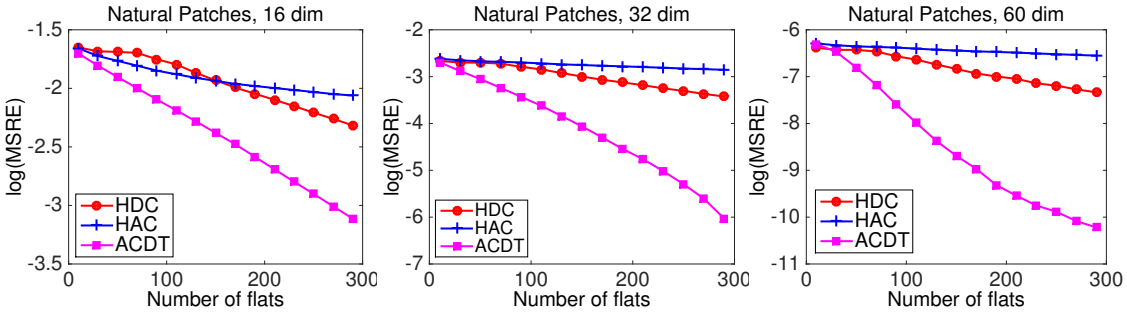
(b) HAC



(c) LSA

(d) Spectral clustering

Figure 3.6 – The final groups formed by the HDC, HAC, LSA and spectral clustering algorithms with 10 flats. Each color represents a different cluster of points.



(a) 16-dimensional flats (b) 32-dimensional flats (c) 60-dimensional flats

Figure 3.7 – MSRE for natural patches for different choices of the flats' dimensionality. The error on the y-axis is shown in logarithmic scale.

Finally, an example of the final groups computed by our algorithm is shown in Figure 3.5 for the case of 12 flats. In this figure, we see that the structure of the manifold is correctly preserved by the proposed manifold approximation algorithm. The final groups for the HDC and the HAC algorithms for the S-curve data are shown in Figure 3.6. Moreover, to strengthen our argument on the inappropriateness of general subspace clustering methods for manifold data, we also provide the results of Local Subspace Affinity (LSA) method [124] and spectral clustering in the same Figure. For spectral clustering we used the same k-nearest neighbor graph as for our own scheme, weighted with tangent distances by the formula $w_{ij} = 1 - \frac{D_T(M_{x_i}, M_{x_j})}{max_{D_T}}$ where max_{D_T} is the maximum tangent distance over the whole dataset. As we can see clearly from the plots, all algorithms fail to uncover clusters that comply with the manifold geometry. The spectral clustering, HDC and HAC achieve better results than LSA but when compared to ACDT it is obvious that they orient their clusters in the wrong way.

3.5.2 Natural patches

We have also tested the performance of our scheme in approximating natural image patches since they are often assumed to form a lower dimensional manifold, e.g. [93]. The manifold samples are taken from the training set of the Berkeley Segmentation Dataset (BSDS) [85]. Each patch is of size 8×8 and captures a square region of a natural image. Before approximating the manifold, we preprocess the patches so that they have zero mean and unit variance. For constructing the manifold we use 10,000 patches and k is set equal to 100.

The approximation performance (in terms of the MSRE) versus the number of flats is presented in Figure 3.7. We have plotted the approximation error for three different choices of the flats' dimensionality, i.e., $d = 16, 32$ and 60 respectively. As we can observe from the plots, in all cases, our scheme approximates significantly better the manifold structure than the other approaches and the differences increase as the dimensionality of flats increases. The performance of the HDC and the HAC schemes is quite similar with the HDC usually outperforming the HAC. These results suggest that our approximation algorithm is very promising even in

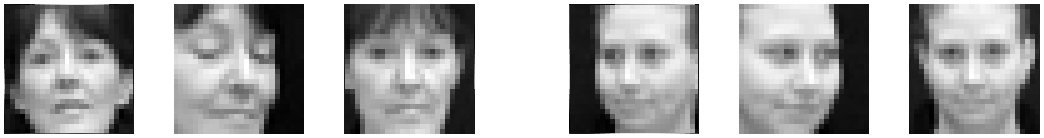


Figure 3.8 – Example faces from the VidTIMIT database after face detection and downsampling. The size of the images is 26×26

cases where the underlying structure of the data cannot be easily identified. The effectiveness of our method is mainly due to the use of the difference of tangents for measuring the linearity of sample sets instead of the geodesic-based criteria used by other algorithms

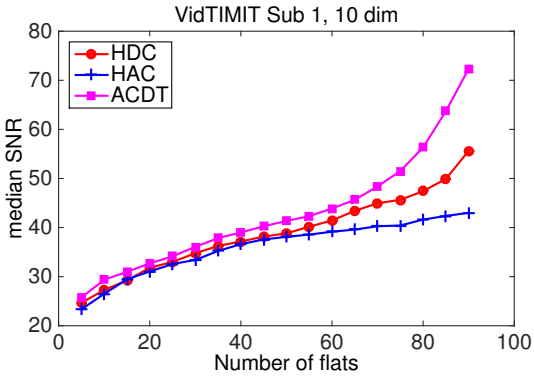
3.5.3 VidTIMIT faces

In a last set of experiments, we have also tested the approximation power of ACDT on faces taken from the VidTIMIT database [102]. This face database contains 3 different video sequences for 43 subjects. In each video sequence, the person performs a head rotation starting from the frontal position and moving sequentially to the right, left, center, up and down. For our experiments, we have first isolated the faces with the P. Viola's face detector [119] from all the video sequences and then downsampled the images to size 26×26. Some resulting example faces are shown in Figure 3.8.

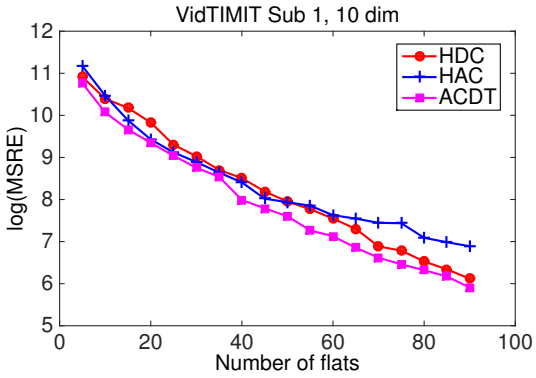
Based on the assumption that all face images belonging to the same subject form a low dimensional manifold, we have used the previous algorithms to approximate this manifold with different number of flats. The dimension of the manifold was set to 10 and the number of neighbors $k = 15$. The results of the approximation for two of the subjects are shown in Figure 3.9. For this experiment,in addition to the MSRE, we also provide results in terms of the median SNR in the image reconstruction. As we can see from these plots ACDT generally outperforms the other two algorithms, although the differences are not extremely big. However, there are sample cases where the performance of the schemes is significantly different. Such an example is shown in Figure 3.10, where we can see that ACDT achieves a significantly better approximation than the other schemes. The reason is that the group that the sample belongs to with ACDT is more uniform than the corresponding group uncovered by HDC and HAC. These groups are shown in Figure 3.11 where it is obvious that the group of ACDT contains mainly frontal poses with open eyes, while the same group in the other algorithms includes also closed eyes and downwards or slightly profile poses.

3.6 Conclusion

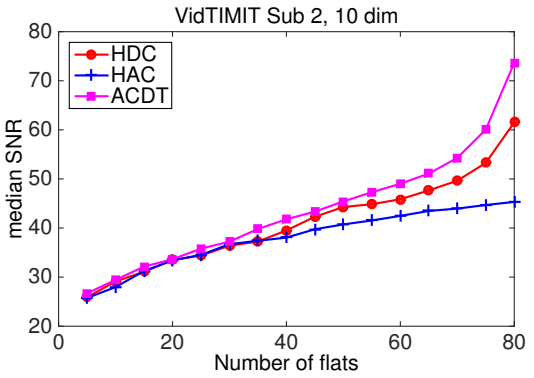
In this chapter, we have considered the problem of manifold approximation with affine subspaces while preserving the underlying structure. For this purpose, we employed a greedy technique that partitions manifold samples into groups based on the difference of local tangents. We have borrowed elements of the constrained clustering theory to motivate the use of



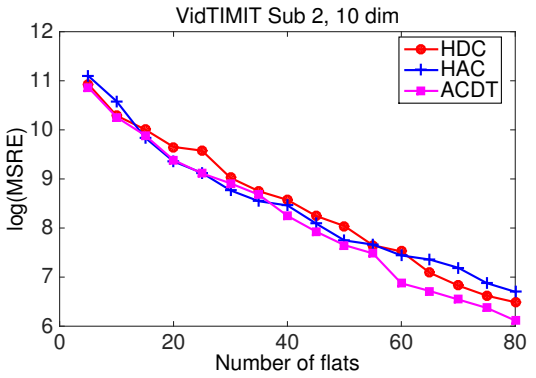
(a) median SNR for Subject 1



(b) MSRE for Subject 1



(c) median SNR for Subject 2



(d) MSRE for Subject 2

Figure 3.9 – Results for the MSRE and median SNR for two subjects in VidTIMIT database.

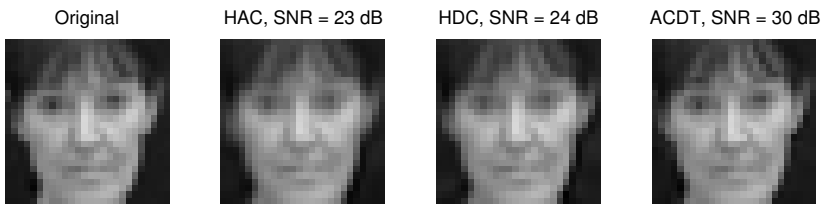


Figure 3.10 – The reconstruction of a sample face based on the approximating flats.

a greedy scheme for manifold approximation. Our method has shown to be quite powerful for manifold approximation where it outperforms state-of-the-art manifold approximation approaches both in terms of preserving the manifold structure and reducing the approximation error.

(a) ACDT

(b) HDC

(c) HAC

Figure 3.11 – The corresponding group for the sample image in Figure 3.10 according to the different approximation schemes.

4 Structured sparse molecule coding

4.1 Introduction

In this chapter, we focus on sparse signal representations and we propose a new signal model to represent signal patterns and higher level structures. Our model represents signals as sparse sets of molecules, which are linear combinations of atoms from a redundant dictionary of elementary functions. It permits to efficiently represent the signal structures as parts or patterns; it builds richer priors than classical structured sparsity models that merely focus on the support of the signal representation and not the actual energy distribution. As such, the traditional priors are not suitable for differentiating patterns with the same support but different distributions, which could actually be very different signal patterns. Such a case is presented in Figure 4.1 where we show how much the image of a face can change when varying the coefficients of its sparse code while keeping the same support.This ambiguity is unfortunately a serious drawback in various applications such as signal recovery and recognition, for example.

To be more specific, we define representative molecules whose prototypes are linear combinations of atoms, or equivalently typical patterns in images. Then, we introduce the idea of molecule realizations in order to take into account the variability of patterns in natural images. The molecule realizations are slightly deformed versions of molecule prototypes with small deviations in the coefficients and possibly in the support of the atoms. However, capturing such changes in the support of the codes is quite challenging. To this end, we form pools of similar atoms in the dictionary, and assume all atoms in a pool carry similar information. Then, we allow atoms in the molecule prototypes to be replaced by similar atoms from their respective pools when forming the molecule realizations. As a result, a given molecule can take various forms that are controlled by the construction of the atom pools. This scheme provides flexibility in the representation of signals with molecules, while preserving the main structural information in the sparse signal approximation. The molecule prototype is essentially expressing a main visual pattern while its realizations allow for signal dependent versions of the main pattern with possibly minor deformations.

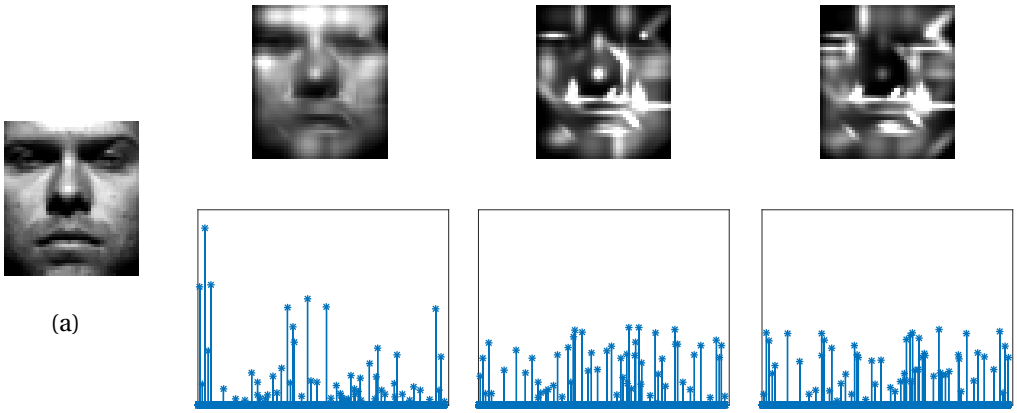


Figure 4.1 – An example of the ambiguity related to the support of the sparse codes. In (a) we show the image of a face and in (b) its sparse approximation with 60 atoms on a dictionary of Gaussian atoms. The next two columns are produced by randomly choosing the values of the coefficients on the same support. The final signal is then normalized. The resulting images are quite different than the original face proving the importance of the coefficients along with the support of the sparse code.

Our efficient structured sparsity model represents a quite unique framework in the literature. In particular, the consideration of the coefficient distribution and the atom pools, as well as the definition of both molecule prototypes and realizations, are important characteristics of our new signal representation model. The coefficients permit to differentiate structures with distinct energy distributions on the same support and thus to facilitate the proper recovery of image information in case of incomplete or inaccurate observations. Another definition of molecule has been previously proposed in [31] to describe a set of coherent atoms in a dictionary, but it is more related to the notion of a group or a pool of atoms than to our original definition of a molecule. The idea of pooling that is used for defining molecules realizations is quite often used under different forms to provide local invariance [66, 67] in the signal representation. In our case however, it provides local invariance to small deformations of a set of atoms with higher resilience to sparse code variability in the identification of typical patterns in images. Finally, the differentiation between the molecule prototypes and molecule realizations in our new model leads to realizations of structures that are signal dependent, like in [98, 128]. Hence, the signal representation is flexible but nevertheless follows a pre-defined structure. The specific characteristics of our scheme make it very suitable for various signal processing tasks and especially signal denoising and inpainting.

The structured sparsity model proposed in this paper is essentially a two-layer architecture with the first layer consisting of the dictionary atoms and the second of the molecules. Deep architectures have been a subject of research for a long time in the machine learning community with very promising recent results [67, 70] and they have recently started becoming more popular in the context of dictionary learning too [109, 101]. In [109], the authors introduced a multilevel dictionary structure where at each level the signals were concentrated near

hyperlines. Molecules could also be considered hyperlines, however in contrast to [109], we allow signals to be composed of more than one molecules in the same level of structure. On the other hand, the structure model in [101, 2] is closer to ours as they authors constrain the dictionary atoms to be sparse combinations of the atoms in a base dictionary. The proposed models however are more rigid than ours, as we include the notions of pools and molecules realizations that enable the proper handling of minor structure deformation in the signals.

The rest of the chapter is organized as follows. In Section 4.2 we describe our model in detail, we discuss different options for the molecule realizations and we exploit the characteristics of atoms pools to design effective similarity measures for detecting energy-based molecule realizations in signals. In Section 4.3 we formally show that our choice of the synthesis dictionary based on molecules realizations provides a good compromise between structure and flexibility. Then, in Section 4.4 we propose a novel constructive sparse coding algorithm of signals with our new structured sparsity model. Finally, in Section 4.5 we show the use of our framework with illustrative experiments in various applications such as compressed sensing, inpainting and denoising. Our results show that our structured sparsity prior leads to better reconstruction performance than classical sparsity priors due to its flexible molecule-based representation.

4.2 Structured image model

We present now our new structured sparsity model for images whose multi-level structure permits to represent visual patterns or typical signal parts as combinations of elementary atoms in a dictionary. In other words, we define molecules as linear combinations of atoms to represent groups of structurally similar signal patterns. We define the concept of molecule prototypes along with molecule realizations that are slightly deformed versions of the prototypes aiming at capturing additional signal variability. We first present our new model and then discuss the concept of molecule realizations in more detail. We start by introducing a simple case of possible deformations, namely the error-based realizations that allow for flexibility only in the coefficients of the molecule prototypes. Next, we extend our definition of molecule realizations to the energy-based ones that allow changes in both the coefficients and the support of the prototype. To this end, we introduce the notion of pools of atoms, which is central for computing energy-based molecule realizations. Based on this notion, we then introduce a new structural difference function that is later used to compare visual patterns when computing image representations.

We first provide an example to illustrate our structured sparsity model. Our model is built on the concepts of molecule prototypes and realizations. The prototype is a representative pattern for a group of molecule realizations, which are slightly deformed versions of a typical image part. The top left image in Figure 4.2 shows a molecule prototype, which is an orthogonal angle formed by two edge-like atoms from the dictionary of elementary atoms. In other words, the molecule prototype is represented by a particular linear combinations of atoms, as

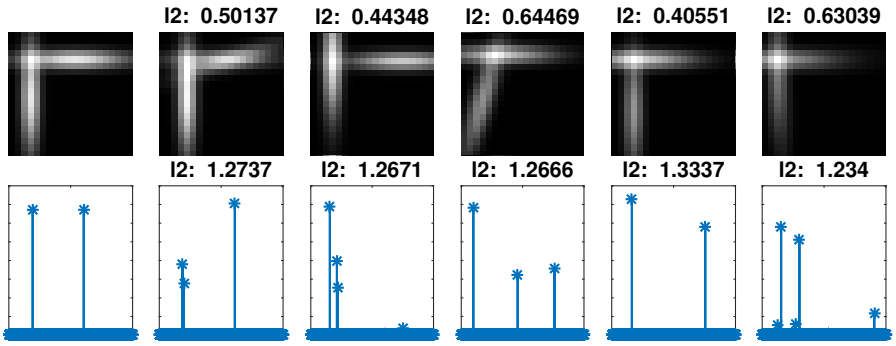


Figure 4.2 – Illustrative example of a molecule prototype and its realizations. In the first row, the molecule prototype (on the left) represents a near orthogonal crossing of edges while the molecule realizations describe visual patterns that are similar to the prototype. The l_2 distance between the prototype and the realizations in the image domain is given on top of each realization. In the second row, we show the corresponding sparse codes of the images in (a). The l_2 distance of the sparse codes seen as vectors in $\Re^N$ is given on top of each figure. As we can see, none of the metrics depicts accurately the structural similarity among the patterns.

shown in the first bottom atomic energy distribution plot in Figure 4.2. The molecule could however appear with small deformations in actual images, and such molecules realizations are illustrated in the rest of the images in Figure 4.2. They look quite similar to the molecule prototype and preserve to some extent its structural characteristics, but they are not constructed with the exact same atoms, as illustrated by their respective coefficient distribution plots in the second row.

4.2.1 Multi-level structure

We now describe our new signal model in more details. We consider a set of signals $X \in \mathbb{R}^{N\times B}$ and a base dictionary $D \in \mathbb{R}^{N\times K}$ of elementary functions or atoms d_k with $1 \leq k \leq K$, whose linear combinations can effectively represent the signals X. We assume that the occurrence of atoms in the signal representation is not completely independent but that atoms rather have the tendency to form typical visual patterns. In other words, there are some linear combinations of atoms that tend to appear more frequently than others, possibly with slight changes either in the energy distribution or atom sets. The most frequent atom combinations are represented by a set of molecule prototypes $M = \{m_l, l \in \{1,..,Q\}\}$ where each prototype is defined as a sparse set of atoms with specific coefficient values, i.e.,

$$m_l = \sum_{k=1}^{K} c_{\pi,l}(k)\, d_k = D c_{\pi,l}, \quad ||c_{\pi,l}||_0 < n \tag{4.1}$$

where n is the sparsity level of the molecules. We assume $c_{\pi,l}(k) \geq 0, \forall k \in [1,\ldots,K]$ and we define the support $\Gamma_{\pi,l}$ of the molecule m_l to be the set of atoms d_k with $c_{\pi,l}(k) > 0$ i.e., $\Gamma_{\pi,l} = \{d_k \in D,\, c_{\pi,l}(k) > 0\}$. The non-negativity of coefficients will be explained in more detail

Term	Description	Definition
Molecule	group of structurally similar signal patterns	prototype & realizations
Molecule prototype	main visual pattern of the molecule	linear combination of atoms specified by the coefficient vector $c_{\pi,l}$
Molecule realization	possibly deformed versions of the main pattern	linear combination of atoms specified by a coefficient vector $c_{x,l}$ with $\Delta(c_{\pi,l}, c_{x,l}) < t, \forall l$

Table 4.1 – Description and definition of the concepts of molecules and their prototype and realizations.

in Section 4.2.3. We can further write all the molecule prototypes in a matrix form as

$$M = DC_\pi, \text{ with } C_\pi = \left[c_{\pi,1}\, c_{\pi,2} \cdots c_{\pi,Q}\right]. \tag{4.2}$$

We consider that the molecules correspond to the most important parts in the signals, but that they may appear as realizations that are similar but not identical to the prototypes. Equivalently, we consider a signal $x \in X$ to be a sparse non-negative combination of molecules realizations plus some bounded noise. We define $c_{x,l}$ as the vector of atom coefficients that expresses the realization of the molecule m_l in x. We further consider that the difference between a molecule realization and the corresponding prototype is small, i.e., $\Delta(c_{\pi,l}, c_{x,l}) < t, \forall l$, where the function Δ measures the structural difference between molecules. The parameter t is a threshold value on the structural difference and its value permits to control the flexibility of our new multi-level model in capturing the variability in typical visual patterns. The signal can therefore be written as

$$x = \quad DC_x a + \eta, \text{ with } C_x = \left[c_{x,1}\, c_{x,2} \cdots c_{x,Q}\right] \text{ and } \Delta(c_{\pi,l}, c_{x,l}) < t, \forall l \in [1, \ldots, Q] \tag{4.3}$$

We further consider that the approximation error is bounded (i.e., $||\eta||_2 < H$), the atom and molecule coefficients are defined as $a_i \geq 0, \forall i$ and $c_{x,i}(k) \geq 0, \forall (k,i)$ and the representation is sparse, i.e., $||a||_0 \leq s$ for some sparsity threshold s.

The image model in Eq. (4.3) corresponds to a sparse decomposition of x into molecule realizations, or equivalently the expansion of the signal x into dictionary atoms whose coefficients are given by $C_x a$. The grouping of atoms into representative molecules is driven by the choice of the structural difference function Δ that quantifies the deviation of molecule realizations from the corresponding prototypes. A summary of the newly introduced concepts of molecules is given in Table 4.1. In the rest of this section, we present more details about the definition of molecule realizations and we provide the corresponding formula for the structural difference $\Delta(c_{\pi,l}, c_{x,l})$.

4.2.2 Error-based realizations

The matrix C_π serves as a rich prior about the signal that specifies simultaneously the support and the coefficients of the molecule prototypes. To allow however some flexibility in the coefficients in the molecule realizations we can incorporate a constrained error matrix to capture such deviations from the prototype coefficients while preserving their support. To be more specific, we denote the error vectors $E_{x,l} \in \mathbb{R}^K$, $\forall l \in [1,\ldots,Q]$ and we have

$$c_{x,l} = c_{\pi,l} + E_{x,l}, \forall l \in [1,\ldots,Q] \tag{4.4}$$

To ensure the preservation of the support we demand that $\Gamma_{x,l} \subseteq \Gamma_{\pi,l}$ where $\Gamma_{x,l}$ and $\Gamma_{\pi,l}$ are the supports of $c_{x,l}$ and $c_{\pi,l}$ respectively. Therefore, from Eq. (4.4) we get that the support of $E_{x,l}$ should also follow the same restriction, i.e., $\Gamma_{E_{x,l}} \subseteq \Gamma_{\pi,l}$. Therefore,

$$c_{x,l} = c_{\pi,l} + E_{x,l}, \text{ with } \Gamma_{E_{x,l}} \subseteq \Gamma_{\pi,l}, \forall l \tag{4.5}$$

In this case, the structural difference $\Delta(c_{\pi,l}, c_{x,l})l$ can be the usual l_1 or l_2 norm of the difference between $c_{\pi,l}$ and $c_{x,l}$ i.e, the norm of the vector $E_{x,l}$. The choice between the two norms depends on the application and it is related to whether the number of erroneous coefficients in the realizations is important or not. In our work [65], we have applied the l_1 norm as structural difference, i.e.$\Delta(c_{\pi,l}, c_{x,l}) = ||c_{\pi,l} - c_{x,i}||_1 = ||E_{x,l}||_1$, with interesting results to denoising experiments.

4.2.3 Energy-based realizations

Even though the definition of molecule realizations in Eq. (4.5) allows some flexibility in the coefficients, it does not cover cases where deviations in the support of the prototypes appear as well, such as the ones that we show in Figure 4.2. In order to extend our definition to capture such additional deviations, we introduce the concept of atom *pools* which is central for computing molecule realizations that can capture both types of variation. The atom *pools* are groups of similar atoms in the dictionary, and we use them to define a structural difference metric Δ that can account for simultaneous deviations in both the coefficients and the support of the prototypes. This measure is later used to compare visual patterns when computing image representations.

Pools of atoms

In our framework, the signal is represented as a linear combination of atoms taken from a redundant dictionary. The redundancy of the dictionary helps in building sparse representations but also leads to the fact that many atoms may carry similar information. In particular, a specific image feature can be well captured by a specific atom d_i in the dictionary. But the same feature might also be well represented by atoms that are similar to d_i, as illustrated in Figure 4.3. Depending on the actual image representation method, the same visual feature

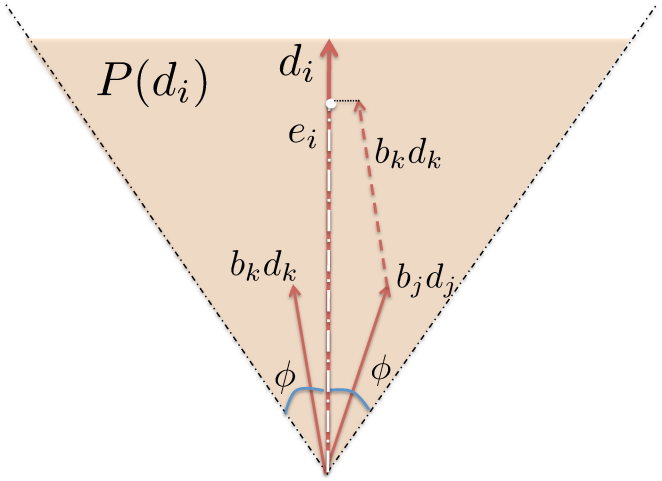


Figure 4.3 – The representation of an atom d_i and its pool $P(d_i)$ in $\mathbb{R}^N$. The pool is defined by the atoms with $cos\,\phi > 1-\epsilon$. Then, $b_k d_k + b_j d_j$ is one possible realization of the atom d_i with energy $e_i = b_k\langle d_i, d_k\rangle + b_j\langle d_i, d_j\rangle$.

can therefore be coded in various ways. We would like to make sure that our structured sparse image model is able to take this phenomenon into account.

We define the notion of atom pools in order to represent atoms that are similar. More specifically, in a dictionary D, each atom d_i can be represented as a unit norm vector in the signal space $\mathbb{R}^N$. Then, there might be other atoms d_j in D that are very similar to d_i, i.e., $\langle d_i, d_j\rangle > 1-\epsilon$, with ϵ the approximation threshold on the similarity of two atoms. In this case, the energy of the projection of d_j on d_i is significant, so that a visual feature may be equivalently well represented by the atoms d_i or d_j. We characterize this phenomenon by introducing the notion of pools of atoms: each atom d_i is related to a pool $P(d_i)$ of atoms d_j 's that are most similar to d_i. In other words, a pool is defined as

$$P(d_i) = \{d_j, 1 \le j \le K, \mid \langle d_i, d_j\rangle > 1-\epsilon\} \tag{4.6}$$

Equipped with this definition, we can now measure the difference between alternative representations of the same visual features. In particular, we can estimate the actual energy corresponding to the atom d_i in a signal represented by the sparse code b that does not actually include the atom d_i. In other words, looking at the sparse signal decomposition $x = Db$ with $b_i = 0$, we would like to know how much of the energy is actually aligned along the direction represented by the atom d_i. It mainly corresponds to the energy captured by the coefficient of all the atoms in the pool $P(d_i)$. We can therefore approximate the energy of the signal in the direction of d_i as

$$e_i(b) = \sum_{j \in P(d_i)} b_j \langle d_i, d_j\rangle = S_i\, b \tag{4.7}$$

where

$$S_i(j) = \begin{cases} \langle d_i, d_j \rangle & \text{if } d_j \in P(d_i) \\ 0 & \text{if } d_j \notin P(d_i) \end{cases} \tag{4.8}$$

The vector S_i expresses essentially the pairwise relationships between the atom d_i and the rest of the atoms in the dictionary D. The energy estimate above is very useful in computing the structural difference between molecules that is explained below. The value of $e_i(b)$ is essentially the length of the projection of the vector $v_i(b) = \sum_{j \in P(d_i)} b_j d_j$, the realization of d_i, in the direction of d_i. When the entries of b are non-negative, v_i is guaranteed to lie in the geometric space defined by the pool $P(d_i)$ and as a result the error $||d_i - v_i||_2^2$ is bounded (the proof is provided in A.1). In the rest, we will adopt this assumption of non-negativity without loss of generality. Finally, an example of the pool of an atom, as well as a possible non-negative realization of the atom from its pool, is shown in Figure 4.3.

Definition of energy-based realizations and their structural difference

Based on the above definition of atom pools, we can now define energy-based molecule realizations with deviations on both the support and the coefficients. A molecule realization of this kind can be defined as the deformation of a molecule prototype whose original atoms could be each substituted by atoms from their respective pool. Equivalently, a molecule realization is essentially a molecule prototype that can be realized through linear combinations of atoms in the pools of the initial prototype. As a result, a molecule realization has a similar energy as the prototype when measured on atom pools but not necessary exactly the same coefficient values or the same support on the atom level.

Following the notation we introduced for the error-based realizations, we can rewrite Eq. (4.4) for the energy-based realizations as

$$c_{x,l} = c_{\pi,l} + E_{x,l}, \text{ with } \Gamma_{x,l} \subseteq \Gamma_{P_{\pi,l}}, \forall l \in [1, \ldots, Q] \tag{4.9}$$

where, as before, $\Gamma_{x,l}$ and $\Gamma_{\pi,l}$ are the supports of $c_{x,l}$ and $c_{\pi,l}$ respectively and $\Gamma_{P_{\pi,l}} = \bigcup_{d_k \in \Gamma_{\pi,l}} P(d_k)$ is the union of the active pools in the ith molecule prototype. The difference with Eq. (4.4) is that an energy-based molecule realization is constrained by specific energy levels on the pools of the atoms in the support of the molecule prototype $\Gamma_{P_{\pi,l}}$ instead of the support $\Gamma_{\pi,l}$ and as a result the realizations are allowed to have non-zero values in the union of the pools of these atoms.

This fact makes it difficult to measure the similarity between the patterns represented by the molecule prototype and its realizations. For example, the l_2 norm in both the image and sparse code domain fail to uncover the structural similarity between the instances, as it does not take into account the actual features represented by the atoms nor their interplay. The inability of the l_2 norm in capturing the similarity of molecules can be observed by checking

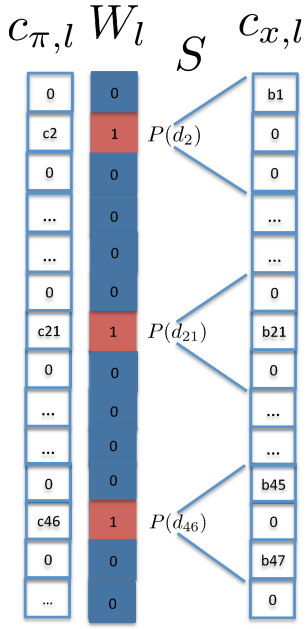


Figure 4.4 – Illustration of a molecule prototype and a possible realization. The vector W_l is the indicator function of the support $\Gamma_{\pi,l}$ of the molecule prototype $c_{\pi,l}$. The structural difference between $c_{\pi,l}$ and $c_{x,l}$ is then $\Delta_(c_{\pi,l}, c_{x,l}) = ||W_l \times (c_{\pi,l} - S\, c_{x,l}))||_2^2 = (c_2 - \langle d_1, d_2\rangle\, b_1)^2 + (c_{21} - b_{21})^2 + (c_{46} - \langle d_{46}, d_{45}\rangle\, b_{45} - \langle d_{46}, d_{47}\rangle\, b_{47})^2$

the norms in Figure 4.2. However, while the l_2 norm cannot be trusted to compare a molecule prototype $c_{\pi,l}$ and a deformation $c_{x,l}$ in the atomic level, the same is not true in the pool level: by definition energy-based realizations have similar energies with the prototypes when measured at the pool level. Therefore, we can define a structural difference measure that compares the energies in the pools of $c_{x,l}$ and $c_{\pi,l}$. Then, if the energies are comparable, the structural difference can be considered to be small.

To be more specific, using the formula for the energy level of an atom based on its pool given in (4.7), we can write the structural difference Δ as:

$$\begin{aligned}\Delta_(c_{\pi,l}, c_{x,l}) &= \sum_{k\in\Gamma_{\pi,l}} (c_{\pi,l}(k) - e_k(c_{x,l}))^2 \\ &= \sum_{k\in\Gamma_{\pi,l}} (c_{\pi,l}(k) - S_k c_{x,l})^2 \\ &= ||W_l \times (c_{\pi,l} - S\, c_{x,l}))||_2^2 \end{aligned} \tag{4.10}$$

where $S = [S_1\, S_2 \cdots S_K]$, with S_i from Eq. (4.8). The indicator vector W_l denotes the inclusion of dictionary atoms in the support $\Gamma_{\pi,l}$ of the molecule m_l, i.e.,

$$W_l(k) = \begin{cases} 1 & \text{if } d_k \in \Gamma_{\pi,l} \\ 0 & \text{if } d_k \notin \Gamma_{\pi,l} \end{cases} \tag{4.11}$$

Note that atoms that participate in the same molecule are assumed to not have overlapping pools which is equivalent to assuming that the atoms in a prototype are quite incoherent. As we will see in Section 4.3 this is a desired property that leads to lower coherence on the dictionary and thus better recovery guarantees. In general, the lower the structural difference $\Delta(c_{\pi,l}, c_{x,l})$, the more compatible the molecule realization and its prototype. Finally, we show an example of a molecule prototype and one possible realization in the atomic level in Figure 4.4 along with the corresponding structural difference function.

4.3 Recovery Analysis

The proposed model presented in Eq. (4.1) defines signals to be be formed as a composition of molecule prototypes with small, controlled deformations. The molecules are further defined as linear combinations of a set of basic atoms. According to this model, one could approximate signals in three different ways, namely as linear combinations of elements in three different dictionaries: the atomic dictionary D, the molecule prototype dictionary DC_π and the dictionary of molecule realizations. In the rest of this section, we analyze the pros and cons of each option in accurately representing signals.

On the one hand, the benefit of the atomic dictionary, is its flexibility since it includes all possible atoms present in signals. However, the lack of any structure makes it less appropriate for recovering signals under challenging conditions, in the presence of intense noise or when information is missing, as the sparsity prior may prove to be insufficient for a satisfactory reconstruction. On the other hand, it is known that the inclusion of more structure in the dictionaries facilitates significantly the task of signal restoration even under severe degradation. The dictionary of molecule prototypes as well as that of molecule realizations have both the advantage of providing structured priors. However, this advantage comes at a price in both cases.

The dictionary of molecule prototypes, might not be always sufficient for retrieving the right structure in the signals. We can rewrite a signal given from Eq. (4.1) as :

$$x = D\,C_x\,a + \eta = D\,(C_\pi + E_x)\,a + \eta \approx D\,C_\pi\,a + D\,C_\pi\,\tilde{a} + \eta = D\,C_\pi\,(a + \tilde{a}) + \eta = D\,C_\pi\,b + \eta$$

where $D\,C_\pi\,\tilde{a}$ is the best approximation of $D\,E_x\,a$ in the dictionary of molecule prototypes DC_π. The vector a is an exact sparse representation. However, E_x can take various forms so that the vector $\tilde{a}$ does not necessarily have a sparse nature in DC_π. Therefore, the structure of b can be significantly different from that of a resulting in a false recovery of the signal structure. The source of the above problem is the lack of flexibility in the dictionary DC_π: it defines patterns through the prototypes to assist the retrieval of degraded signals but at the same time the dictionary elements are quite rigid and restrictive.

Therefore, it appears that building a dictionary with all possible molecule realizations, denoted as DC_x, could be a better and more flexible alternative with a compromise between structure

and flexibility. However, building a dictionary with all possible molecule realizations results in a very coherent representation. As we have seen in Section 4.2.1, the molecule realizations are essentially small deformations of a molecule prototype. Therefore, all realizations of the same prototype are highly similar. The recovery performance of a dictionary is known to deteriorate as the sparsity of the signals decreases and the coherence of the dictionary increases. To put it more formally, a known recovery constraint for BPDN (Basis Pursuit Denoising) [22] or OMP (Orthogonal Matching Pursuit) [112] is given by

$$k \leq \frac{1}{2}(\frac{1}{\mu_x} + 1). \tag{4.12}$$

where μ_x is the coherence of the underlying dictionary and k is the sparsity of the signal, i.e., the number of elements in the signal. The coherence μ_x equals the maximum absolute inner product between two distinct vectors in the dictionary, i.e.,

$$\mu_x = \max_{d_j, d_k \in DC_x, j \neq k} |\langle d_j, d_k \rangle| \tag{4.13}$$

Therefore, the more coherent the dictionary DC_x , the more sparse the signals should be in order to be able to recover them.

We can analyze how the coherence μ_x of the dictionary DC_x is affected by the presence of multiple realizations for each molecule prototype. Since the realizations of the same molecule prototype are very similar, μ_x can be lower bounded using the maximum distance r between any realization and the corresponding molecule prototype. The theoretical bound, $L_x \leq \mu_x$, is given by

$$L_x = 1 - 2r^2 \tag{4.14}$$

To quantify this result, we can compare the molecule realization dictionary with the case of a dictionary DC_u that contains only one molecule realization per molecule prototype. The restriction on the allowed number of instances per prototype allows for a theoretical upper bound on the coherence μ_u of the dictionary DC_u, i.e., $U_u \geq \mu_u$ with

$$U_u = \mu(1 - 2r^2) + 2r\sqrt{(1-\mu^2)(1-r^2)} \tag{4.15}$$

where μ is the coherence of the dictionary of molecule prototypes DC_π. In practice the coherence μ_u is expected to be close to μ. Both theoretical bounds depend on the distance r which is driven by the characteristics of the atoms pools as well as the internal structure of the molecules. The latter is measured by the maximum similarity between atoms belonging to the same molecule, denoted as μ_M. To improve the readability of the section we have moved the exact expressions for r as well as the proofs for the bounds in the A.2.

From the expression for L_x we can see that the smaller the r is, the worse the μ_x is expected to be. On the other hand, when r is small, U_u gets closer to μ. In order to present these

dependencies more concretely, we show in Figure 4.5 some plots of μ_x and μ_u for various settings. At the first row, we present the bounds L_x and U_u computed based on Eq. (4.14) and (4.15) respectively while at the second row we show the mean values of μ_x and μ_u computed experimentally for different values of the molecule prototype coherence μ over random generations of the dictionaries DCu and DCx. For simplicity, in our calculations we have assumed that the number of atoms in all molecules is the same, denoted as n. The pool angle ϕ was set to 10 degrees while we varied the maximum in-molecule atomic similarity μ_M. In both rows, the red line refers to the coherence of the DC_x dictionary, the blue line to the coherence of DC_u and the yellow to that of molecule prototypes DC_π.

From the figures, according to the values of the bounds L_x and U_u, the benefit of the use of DC_u over DC_x is more prominent when the molecule prototypes are not very coherent (lower values of μ). In this case, the lower bound for μ_x, L_x, is higher than the upper bound for μ_u, U_u, so that μ_u is guaranteed to be lower than μ_x. This benefit depends also on the coherence of the atoms belonging to the same molecules: it is larger when μ_M is low. However, the analysis of the experimental mean shows that in practice the coherence μ_u of the dictionary DC_u lies very close to the coherence of the initial molecule prototype dictionary DC_π, while μ_x lies always close to 1. Therefore, we observe that restricting the number of realizations in the dictionary to one per molecule prototype preserves the dictionary coherence quite well while the inclusion of more than one molecule realizations per prototype pushes the dictionary coherence towards 1.

To sum up, from the above discussion we can see that deciding which dictionary to use for signal decomposition is not trivial. The underlying atomic dictionary D lacks structure, the dictionary of molecule prototypes DC_π lacks flexibility while the dictionary of all molecule realizations suffers from inefficient size and high coherence. To alleviate this issue, we propose an iterative decomposition scheme that searches for the best molecule realizations using at each iteration a synthesis dictionary with strictly one molecule realization per molecule prototype, denoted as DC_u above. In this way, at each iteration we have a guarantee for the coherence of the used dictionary while through the iterations we expect to recover the right signal structure. The details of the exact problem formulation as well as the proposed solution are presented in the next Section.

4.4 Adaptive molecule coding algorithm

We now formulate the problem of decomposing a signal into a sparse set of molecule realizations. From now on, we will mainly refer to energy-based molecule realizations as they are the most generic ones. We assume that the signal x follows the model in Eq. (4.3), or equivalently that the signal can be well approximated by a sparse linear combination of molecule realizations represented by C_x along with their respective coefficients a. Each molecule realization in C_x is an energy-based realization of the corresponding molecule prototype in C_π. The signal approximation can then be computed by solving the adaptive molecule coding problem

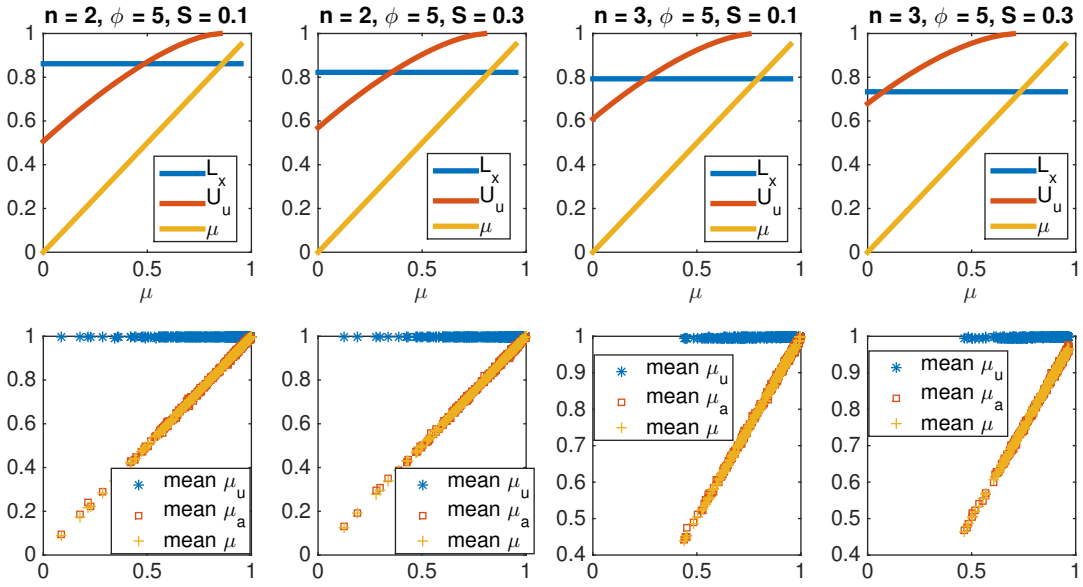


Figure 4.5 – Comparison plots for the coherence of the dictionaries DC_x and DC_u containing many VS one realizations per molecule prototype respectively. The plots are for different values of the number of atoms per molecule n, the size of the atoms pools ϕ as well as the maximum similarity of atoms in the same molecule μ_M. In the first row we plot the theoretical bounds while in the second the average coherence observed over random generations of the dictionaries DC_x and DC_u.

written as follows:

$$\{\hat{a}, \hat{C}_x\} = \underset{\substack{a,\, a(l)\geq 0,\, \forall l \\ C_x,\, C_x(k,l)\geq 0\, \forall k,l \\ \Gamma_{x,l}\subseteq\Gamma_{P_{\pi,l}},\, \forall l}}{\arg\min} \left[||x - DC_x a||_2^2 + \lambda_1 ||a||_1 + \sum_{l, a(l)>0} \left(\lambda_2 \Delta(c_{\pi,l}, c_{x,l}) + \lambda_3 ||c_{x,l}||_1 \right) \right] \tag{4.16}$$

where each $c_{x,l}$ is a molecule realization for the molecule prototype $c_{\pi,l}$ with $l \in [1,\ldots,Q]$. The $\Gamma_{x,l}$ is then the support of $c_{x,l}$ and $\Gamma_{P_{\pi,l}} = \bigcup_{d_k \in \Gamma_{\pi,l}} P(d_k)$ is the union of the active pools in the ith molecule prototype with support $\Gamma_{\pi,l}$. The first term in the objective function in Eq. (4.16) is the error of the approximation of the signal with a sparse set of molecule realizations. The second term favors a sparse approximation with the l_1 norm of the coefficient vector a. The last term drives the form of the molecule realizations: the term $\Delta(c_{\pi,l}, c_{x,l})$ tends to favor molecules realizations that are close to prototypes while the l_1 norm on the molecule realization codes $c_{x,l}$ ensures their sparsity. The constraint on the support $\Gamma_{x,l}$ is necessary from the definition of energy-based molecule realizations given in equation (4.9). Finally, the weight parameters λ_i's permit to balance the different terms of the objective function.

By substituting the structural difference function from Eq. (4.10) in Eq. (4.16) we get:

$$\{\hat{a}, \hat{C_x}\} = \underset{\substack{a,\, a(l)\geq 0,\, \forall l \\ C_x,\, C_x(k,l)\geq 0\, \forall k,l \\ \Gamma_{x,l}\subseteq\Gamma_{P_{\pi,l}},\, \forall l}}{\arg\min} \Bigg[||x - DC_x a||_2^2 + \lambda_1 ||a||_1 + \lambda_2 \sum_{l,a(l)>0} ||W_l \times (c_{\pi,l} - S\, c_{x,l})||_2^2 + \lambda_3 \sum_{l,a(l)>0} ||c_{x,l}||_1 \Bigg] \tag{4.17}$$

where W_l is given in Eq. (4.11). For a given dictionary D, a set of pools represented by S and a set of molecule prototypes written as C_π, the objective function in Eq. (4.17) is minimized when the variables a and C_x form a structured sparse approximation of x. However, the above optimization problem cannot be solved easily as it is not jointly convex for both variables a_x and C_x. However, when one of the variables is fixed, the problem is convex with respect to the other one. Therefore, we adopt an alternating optimization technique with two steps for solving the optimization problem in Eq. (4.17). The two steps are computed as follows.

1. We first fix the set of molecules realizations, and solve the sparse coding problem for the coefficient vector a. Given C_x, the solution for a can be found as:

$$\hat{a} = \underset{a,\, a(l)\geq 0,\, \forall l}{\arg\min} \left[||x - DC_x a||_2^2 + \lambda_1 ||a||_1 \right] \tag{4.18}$$

2. Then, we fix the coefficient vector, and find the set of molecule realizations that minimize the objective function of the coding problem. Given a, the solution for C_x can be found as

$$\hat{C_x} = \underset{\substack{C_x,\, C_x(k,l)\geq 0\, \forall k,l \\ \Gamma_{x,l}\subseteq\Gamma_{P_{\pi,l}},\, \forall l}}{\arg\min} \Bigg[||x - DC_x a||_2^2 + \lambda_2 \sum_{l,a(l)>0} (||W_l \times (c_{\pi,l} - S\, c_{x,l})||_2^2 + \lambda_3 \sum_{l,a(l)>0} ||c_{x,l}||_1 \Bigg], \tag{4.19}$$

The first problem is essentially an l_1 regularized sparse coding problem which is convex on a. It can be solved with many different algorithms, e.g., [84, 7]. In our case we have chosen to solve it with the method of alternating direction method of multipliers (ADMM) [17]. Following the findings in [24], we have also employed the method of reweighted l_1-minimization to get to a sparser solution. Note that, at the very first iteration of the global algorithm, C_x is initialized with C_π, while it is later updated during the solution of the second step of the alternating algorithm.

The second problem is also convex. As for the first problem, we have chosen to solve it with ADMM [17]. In order to solve it more efficiently, we have transformed it into a more convenient form that allows for the optimization over one vector of coefficients b instead of the whole

matrix C_x. Since the support of each molecule realization $\Gamma_{x,l}$ is restricted to the union of the pools of the active atoms in the corresponding molecule prototype, i.e., $\Gamma_{P_{\pi,l}}$, many of the entries in matrix C_x are constrained to be zero. The vector b represents the possible non-zero entries in C_x, i.e. the coefficients of the atoms in $\cup_{l,a(l)>0}\Gamma_{P_{\pi,l}}$. Essentially it expresses the flexibility that is allowed in the molecule realizations once the molecules are chosen.

To complete our problem transformation, we introduce the vector $\tilde{C}$ that expresses the expected energy in the atoms pools. It is created by concatenating into vector form the entries in C_π that correspond to the energy expected in each pool of active atoms. Equivalently, the corresponding dictionary of atoms $\tilde{D}$ is created by concatenating the atoms in each of the active pools. Finally, the new vector of relationships $\tilde{S}$ between atoms in $\tilde{D}$ replaces the vector S. With these modifications, the problem in Eq. (4.19) can be equivalently expressed as:

$$\hat{b} = \arg\min_{b} ||x - \tilde{D}\, b||_2^2 + \lambda_2 ||\tilde{C} - \tilde{S}\, b||_2^2 + \lambda_3 ||b||_1 \text{ with } b(k) \geq 0, \forall k \tag{4.20}$$

Solving this problem is more efficient in terms of time and space than solving the equivalent problem in Eq. (4.19) as the size of the vector b is usually much smaller than that of the whole dictionary D.

Finally, we iterate between the two optimization problems until the value of the signal reconstruction doesn't change much. Although this alternate optimization technique does not have any optimality guarantee, it gives good results in practice and therefore offers an effective constructive solution to the sparse coding problem of Eq.(4.17). Since the algorithm has several constraints on the structure and sparsity the final molecule realizations cannot be completely different from the predefined molecule prototypes and as a result the quality of the signal reconstruction depends significantly on the initialization of the molecule structure. However, the design and learning of good molecule prototypes is beyond the scope of this chapter which is mainly focused on the sparse coding step. We will however, discuss it more in the next chapter. Finally, as long as the parameters of the algorithm are concerned, the values for the λ's were chosen according to each specific task based on a small validation set. The value for the parameter r required for the ADMM method was set to 1 for all the experiments. The pseudocode of the complete sparse coding scheme, called Adaptive Molecule Coding (AMC), is presented in Algorithm 2.

4.5 Experimental results on signal restoration

Next, we have evaluated the effectiveness of our model for various image restoration tasks on both synthetic and real data. In signal restoration, a high quality signal x needs to be reconstructed from its degraded measurements y. The problem can be modeled in a generic form as

$$y = Hx + v \tag{4.21}$$

Algorithm 2 Adaptive molecule coding (AMC)

function AMC($x, D, C_\pi, S, \lambda_1, \lambda_2, \lambda_3, \epsilon$)
  $\hat{a} = \arg\min_a \left[||x - D\,C_\pi\,a||_2 + \lambda_1 ||a||_1 \right], a \geq 0$ ▷ *Initialize a*
  while true **do** ▷ *Alternate optimization*
    $(\tilde{D}, \tilde{S}, \tilde{C}) = transform(D, C, S, \hat{a})$ ▷ *Create new variables for Eq.* (4.20)
    $\hat{b} = \arg\min_b \left[||x - \tilde{D}\,b||_2^2 + \lambda_2 ||\tilde{C} - \tilde{S}\,b||_2^2 + \lambda_3 ||b||_1 \right], b \geq 0$ ▷ *Solve for b*
    $\hat{C_x} = transform^{-1}(\hat{b}, C, \hat{a})$ ▷ *Reconstruct C_x from b*
    $w = 1./\hat{a}$ ▷ *Set new weights for re-weighted l_1*
    $\hat{a} = \arg\min_a \left[||x - D\,\hat{C_x}\,a||_2 + \lambda_1 ||w. * a||_1 \right], a \geq 0$ ▷ *Solve for a*
    if $\hat{C_x}\hat{a} - C_p\,a_p < \epsilon$ **then** return ▷ *If signal coding did not change significantly, stop*
    else
      $a_p = \hat{a}, \quad C_p = \hat{C_x}$
    end if
  end while
  return $\hat{a}, \hat{C_x}$
end function

where H is a degrading operator and υ is additive noise.

4.5.1 Synthetic Data

For the case of synthetic data, we have used a dictionary of gaussian anisotropic atoms with mother function $\phi(x, y) = A\exp(-(x/2)^2 - y^2)$. We have sampled the image plane for two scale levels [0.5 1] with a step size 1 for translation and $\pi/6$ for rotation. The atoms of the dictionary were combined according to 10 predefined molecules contained in C_π. The size of the signals and the molecules was 10×10. Each molecule was randomly constructed to contain 2, 3 or 4 atoms of equal energy. Then each signal was created as a random combination of a few molecule realizations (2,3 or 4).

To produce a molecule realization we have followed the definition of energy-based realizations given in Section 4.2.3. To be more specific, for each atom in the molecule prototype we produced an approximation using the atoms in the atom's pool. The approximating atoms were chosen randomly, their total number drawn from a geometric distribution with $p = 0.7$ (so that the approximation is a sparse combination of atoms) while their coefficients were adjusted so that the projection of their combination to the atom direction is close to the original coefficient value. Finally, for each restoration task, the appropriate operator was applied to get the testing data.

We have compared our method with the l_1-l_2 group norm [58] that assumes that the atoms are forming groups and penalizes the l_1 norm on the groups instead of the atoms by substituting the atom coefficients with the l_2 norm on each group (the algorithm is denoted as A_{12} in the rest). To define each group $g_i \in \mathcal{G}$ we have used the support of the corresponding molecule m_i. The atoms that didn't belong to any group, were considered as separate groups of size 1.

The resulting optimization problem was:

$$\hat{b} = \arg\min_{b}\{||y - H\,D\,b||_2 + \lambda \sum_{g_i \in \mathcal{G}} ||b_{g_i}||\} \tag{4.22}$$

where b is the signal decomposition in the atomic level and b_{g_i} is its restriction on g_i. The decomposition $\hat{a}$ in groups (or equivalently molecules in our case) is computed as the l_2 norm of the coefficients in each group i.e., $\hat{a}_i = ||\hat{b}_{g_i}||$.

As we have discussed before in Section 4.3, one alternative for the synthesis dictionary is the dictionary of molecules prototypes. This approach is similar to the sparse coding step in [101] that assumes a double sparsity structure prior where the learned atoms are constrained to be linear combinations of a set of base atoms. This way, the learned atoms are similar to our molecule prototypes. However, the proposed sparse coding does not allow the atoms to further adjust to the signals. Therefore, their approach is equivalent to sparse coding with l_1 regularization on the molecule dictionary, i.e., the outcome of:

$$\hat{a} = \arg\min_{a}\{||y - H * D_\pi * a||_2 + \lambda||a||_1\} \tag{4.23}$$

where $D_\pi = DC_\pi$ is the molecule dictionary.In the rest, we denote this algorithm as A_m.

Finally, we have also compared our scheme with simple sparse coding on D, i.e.,

$$\hat{a} = \arg\min_{a}\{||y - H * D * a||_2 + \lambda||a||_1\} \tag{4.24}$$

The method is denoted A_1 in the rest.

The performance of the algorithms is compared using various measures. To quantify the performance in terms of the signal recovery we have computed both the mean square reconstruction error of the signal approximation (MSRE), i.e., $\frac{\sum_i ||x_i - \hat{x}_i||^2}{N}$ where $\hat{x}$ is the signal reconstruction and N is the number of signals , as well as the mean sparsity ratio of the recovered representations where the sparsity ratio is computed as the l_0 norm of the recovered representation in D over the l_0 norm of the true atomic representation. Finally, we are also interested in how effective are the schemes in detecting the correct molecules. Therefore, we have also computed the accuracy of the molecule detection, which is the ratio of the correctly categorized molecules ($TP + TN$) over all the molecule instances ($P + N$).

Denoising

To start with, we have tested the performance of the schemes under noise. In this case, $H = I$ and v is white gaussian noise. The results, for different noise levels, are shown in Figure 4.6. For each noise level, the results were averaged over 5 different molecule matrices and 1000 signal instances per matrix. The parameters for each algorithm, chosen based on a small validation set, were: $\lambda_1 = 0.01, \lambda_2 = 1, \lambda_3 = 0.1$ for AMC and $\lambda_1 = 0.1$ for all the rest. From

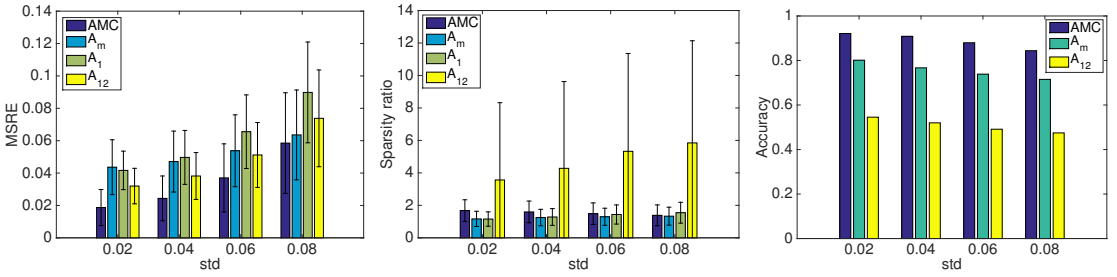


Figure 4.6 – The results for denoising on synthetic data with different coding schemes. The performance is evaluated with the MSRE of the reconstructed signals as well as the sparsity ratio and the accuracy of the recovered representations.

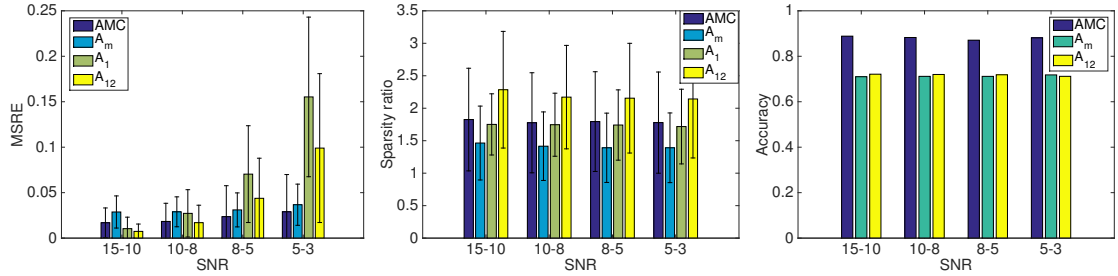


Figure 4.7 – The results for inpainting on synthetic data with different coding schemes. The performance is evaluated with the MSRE of the reconstructed signals as well as the sparsity ratio and the accuracy of the recovered representations.

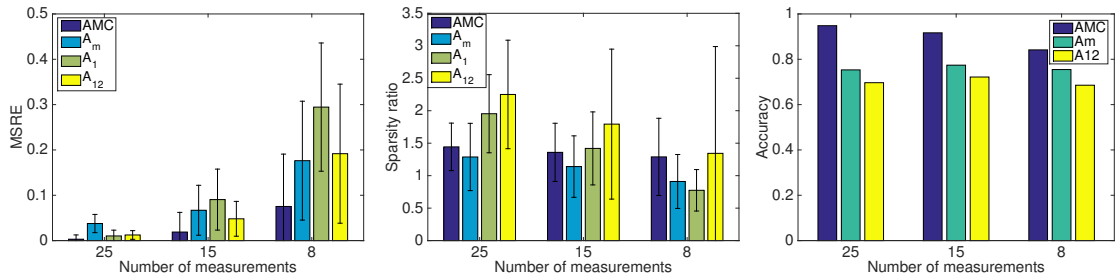


Figure 4.8 – The results for compressed sensing on synthetic data with different coding schemes. The performance is evaluated with the MSRE of the reconstructed signals as well as the sparsity ratio and the accuracy of the recovered representations.

Figure 4.6 we can observe that as the noise increases the effectiveness of the structure is more prominent: the MSRE of A_1 progressively deteriorates compared to the other 3 schemes that use a structured prior. Moreover, for the highest noise level the A_m scheme which is the one with the least flexible structure prior, almost reaches the best performance. However, our scheme manages to perform best for all the noise levels by uncovering signal representations with small MSREs, accurate molecule detection, and satisfactory sparsity (A_m has a fixed sparsity level for each molecule, therefore it is expected to have the lower value as the most constrained one).

Inpainting

Next, we have tested the performance of the schemes for inpainting. In this case, we have created a set of signals by omitting the signal values in a randomly chosen square region. We have tried three different sizes for the region: 3×3, 4×4 and 5×5. Then, the signals were divided into 4 sets based on their SNR. The signal recovery problem was solved over the known regions of the signals: each signal x was expressed as $x' = P_x .* x$ where P_x is the mask denoting the known region. In this case, $H = P_x .* I$ resulting in masking each dictionary atom. No extra noise was added to the data. The values for the parameters were $\lambda_1 = 0.001, \lambda_2 = 1, \lambda_3 = 0.1$ for AMC and $\lambda = 0.01$ for all the rest. The results are shown in Figure 4.7. Again, we can observe the benefits from the flexible prior that our scheme provides compared to the rest: the MSRE is always the lowest, the accuracy is the highest while the sparsity ratio is satisfactory, usually the lowest after A_m which is the most constrained one. In case of highly disturbed signals (lowest SNR) the Am also outperforms the rest, proving the importance of structure in applications were there is a significant amount of missing information.

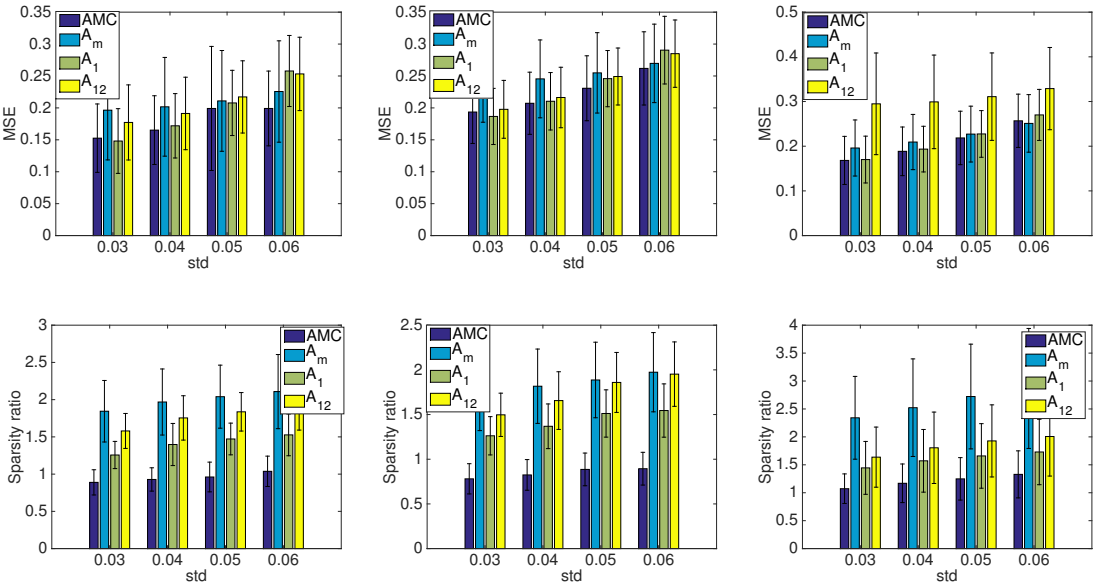
Compressed Sensing

Finally, we have compared the recovery performance of the schemes for compressed sensing. The measurement process was performed by setting $H = \Phi$ where Φ is a random projection matrix. The entries of Φ were independent realizations from a standard normal distribution. We have checked three different sizes for Φ namely 25, 15 and 8 measurements. For each number of measurements the results were averaged over 5 different instances of matrix Φ. The values of the parameters were $\lambda_1 = 0.01, \lambda_2 = 10, \lambda_3 = 0.01$ for AMC and A_1 while $\lambda_1 = 1$ for Am and $\lambda_1 = 0.01$ for A_{12}. The results for the different number of measurement are shown in Figure 4.8. Our scheme significantly outperforms the rest as the number of measurements decreases while keeping a high accuracy on molecule detection. The sparsity ratio is almost stable for all sizes of measurement matrix and quite close to 1 which is the desired value.

4.5.2 Denoising of digit images

Next, we have used our adaptive molecule coding scheme to perform denoising on MNIST images [72]. The images have been downsampled to 14×14 and normalized. In order to better fit the signal model the digits were further coarsely pre-aligned to avoid big discrepancies in the position and the orientation. The molecule prototypes were extracted using the algorithm presented in [101] from 1000 examples per digit while for the testing we used 100 examples per digit. The denoising performance was tested over different noise levels and measured by the mean squared reconstruction error and the mean sparsity ratio. The parameters were fixed according to a small validation set and their values were $\lambda_1 = 0.001, \lambda_2 = 0.01, \lambda_3 = 0.01$ for AMC and $\lambda_1 = 0.01$ for the rest of the schemes.

The results of our experiments are presented in Figure 4.9. We have experimented with both



(a) Digit 0 (b) Digit 9 (c) Digits 0,1,2,3

Figure 4.9 – Results for denoising on data from MNIST digits for various levels of noise. On the first row we plot the MSE and on the second the sparsity ratio of the results. In the first two columns we present the results obtained when each digit was treated separately while on the third row we simultaneously denoised digits from different classes. The results were obtained with 20 molecule prototypes per digit.

denoising each digit separately using molecules extracted only for its class as well as denoising with molecules extracted from many classes simultaneously. In the first two columns we show the results we obtained for digits 0 and 9 separately while in the third column we plot the results for the case of denoising digits 0, 1, 2 and 3 with molecules extracted for all 4 digits together. From the plots we can see that AMC is the scheme that manages to perform well for all different noise levels. As expected the benefits from rich structure priors are more prominent in the presence of severe noise, where A_m, the scheme with the most restrictive prior, outperforms A_1 and A_{12} that have loser priors. However, for lower noise levels the performance of Am is not sufficiently good due to the rigidity of its prior. Our scheme on the other hand performs well in all cases as it adapts to the signals almost as succesfully as A_1 in the less noisy cases, while preserving the structure as A_m in the more noisy cases. Finally, it is also important to note that AMC is the scheme that achieves on average a sparsity ratio close to one, meaning that it is highly efficient as it achieves a good signal restoration using only as many components as it is necessary.

4.5.3 Restoration of image patches

Finally, in image restoration it is often the case that the non-local similarities that different regions of the image may exhibit are used to enhance the restoration process [33, 81]. The idea

of 'nonlocally centralized' sparse codes is not very far from the idea of molecule prototypes. Therefore, we have followed the same intuition to define molecules prototypes based on the non-local similarity of patches and use their deformed versions to further enhance the image recovery from compressed measurements.

To be more specific, when only sparsity is used as a prior for the recovery of the patches x_i of an image X, the recovery problem for each patch can be written as:

$$\hat{a}_i = \arg\min_{a_i} ||y_i - \Phi D a_i||_2^2 + \lambda_1 ||a_i||_1 \tag{4.25}$$

where a_i is the decomposition of the patch x_i in the dictionary D and y_i is the measurements acquired for this patch. The recovered image $\tilde{X}$ is then created by the recovered patches $\tilde{x}_i$.

However, when taking into account the non-local similarity of the patches, a molecule prototype can be extracted for every patch and further enhance the recovery by restricting the code of each patch to be a realization of the prototype. The corresponding coding problem is then:

$$\hat{c}_{x,i} = \arg\min_{c_{x,i}} ||y_i - \Phi D c_{x,i}||_2^2 + \lambda_2 ||W_i \times (c_{\pi,i} - S * c_{x,i})||_2^2 + \lambda_3 ||c_{x,i}||_1 \tag{4.26}$$

where $c_{\pi,i}$ is the molecule prototype for $\tilde{x}_i$ and $c_{x,i}$ is the patch dependent molecule realization. In order to obtain $c_{\pi,i}$ we search the image $\tilde{X}$ for the most similar patches to $\tilde{x}_i$ and we build a set Ω_i as in [33]. Then, based on the sparse codes of the patches in Ω_i we extract a molecule prototype for $\tilde{x}_i$. The prototype extraction algorithm is a greedy procedure that identifies a small number of atoms to account for most of the energy in the sparse codes in Ω_i while taking into account the atoms pools. It is an iterative procedure that at each step adds in the support of the molecule prototype the atom with the most energy in its pool. The energy of the atoms falling in the already chosen pools is considered covered and the algorithm iterates until a sufficient amount of the energy is covered. In this way, we extract a molecule prototype $c_{\pi,i}$ that accepts as realizations all the patches in Ω_i.

To show that our proposed coding scheme is suitable for enhancing the recovery of the original image, we have compared it to the λ_1 based sparse coding presented in Eq. (4.25) which only imposes sparsity as structure. Moreover, following the ideas in [33], we have also implemented a scheme where the imposed structure is defined as the mean sparse code over similar patches. The corresponding optimization problem is then:

$$\tilde{a}_i = \arg\min_{a_i} ||y_i - \Phi D a_i||_2^2 + \lambda_2 ||\hat{a}_i - a_i||_2^2 + \lambda_3 ||a_i||_1 \tag{4.27}$$

where $\hat{a}_i$ is the mean sparse code obtained from the sparse codes of the patches in Ω_i.

We have tested the performance of the above schemes on the images 'House' and 'Barbara'. Each image was divided in 10×10, non-overlapping patches. As a base dictionary D we have used a DCT overcomplete dictionary with 256 atoms. For solving the coding problem in Eq. (4.26) we have used the Algorithm 2, namely the part that solves for C_x given a, as in this

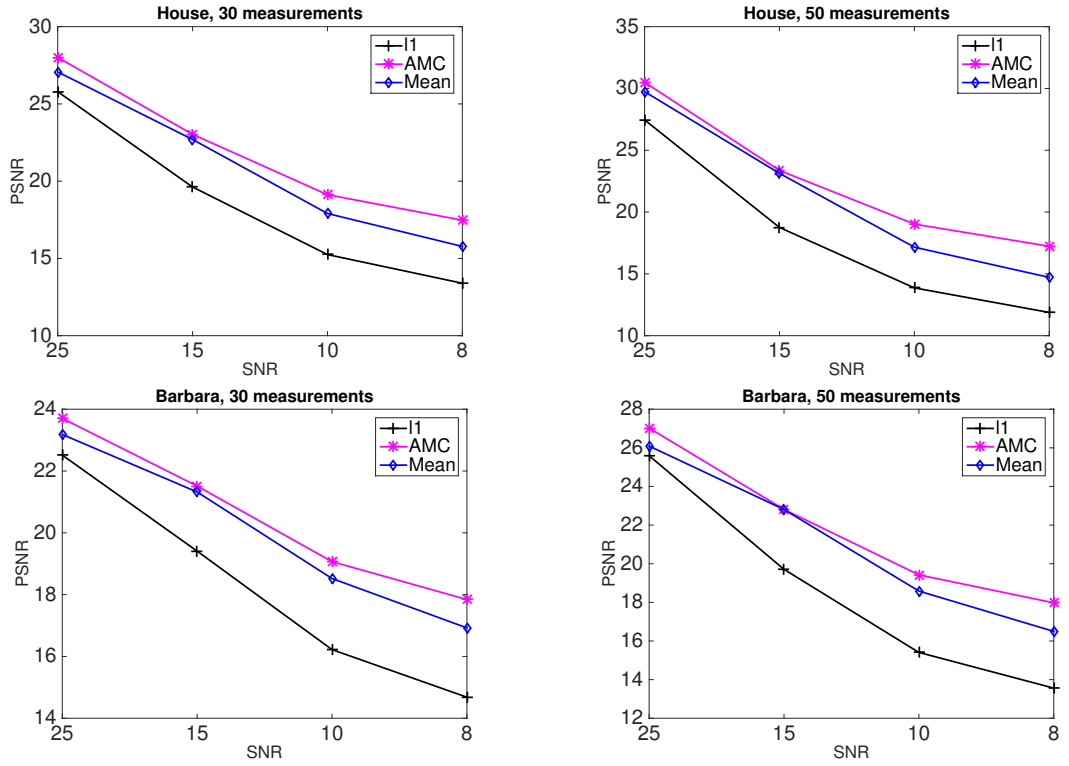


Figure 4.10 – Results for image recovery with compressed measurements. The values of the parameters were set to $\lambda_1 = 10$ and $\lambda_2 = \lambda_3 = 1000$.

case for each patch there is only one molecule prototype and as result the vector of molecule coefficients is set to 1. The entries of Φ were independent realizations from a standard normal distribution. We have checked two different size for Φ, namely 30 and 50 measurements, while for each number of measurements the results were averaged over 5 different instances of the matrix Φ. The measurements were further corrupted with noise.

In the Figure 4.10 we show the PSNR of the recovered images based on the three different schemes for various levels of noise and the two different number of measurements. From the results we can verify that the non-local similarity of the patches is very helpful for the image restoration as the λ_1 sparse coding has a much lower PSNR than the other two schemes. Moreover, our molecule based coding scheme manages to extract more effectively the structural similarities of the patches than the mean sparse code as it achieves better PSNR results for the majority of settings. Therefore it is proven that the idea of molecule prototypes and realizations based on atoms pools is a powerful one providing correct priors for patch based restoration of images.

4.6 Conclusions

In this chapter we have presented a new two-layer structure model for signals. We have defined our structural elements, the molecules, as linear combinations of atoms and we have distinguished between molecule prototypes and molecule realizations to add more flexibility in the model. For the molecule realizations, we started with allowing small errors on the sparse coefficients and then we extended our definition to allow deviations on both the support and the coefficients of the prototypes based on the notion of pools of atoms. We have presented a new algorithmic scheme for adaptive molecule coding (AMC) and we have conducted experiments on both synthetic and real data that proved the effectiveness of our model for various restoration tasks.

In this chapter, our goal with our new structure model has been to provide a better modeling strategy for higher level patterns while allowing for invariance to small deformations. In the next chapter, we use these concepts further to move the structure analysis from the signal to the sparse code domain. To this end, instead of considering that the molecules are given a priori, we devise a scheme for representing the codes with molecule realizations and learn the corresponding molecule prototypes at the same time, directly in the sparse code domain.

5 Structure learning from sparse codes

5.1 Introduction

In this chapter we address the problem of learning the signal structure. We focus on the signals' sparse codes and we aim at uncovering structured representations for the codes without transforming them back to the signal domain. Although sparse codes have been employed successfully in various tasks like signal restoration and classification, they usually require the explicit knowledge of the atoms of the underlying dictionary and the coding algorithm. However, this can prove quite restrictive in practice as the dictionary is not always known explicitly. Moreover, in this way we lose the opportunity to work with sparse data which is easier from a computational point of view than working with the original signals. Additionally, biological evidence supports the use of sparsity to produce signal representations that can fully describe the corresponding signals without reconstructing it. Our goal here is therefore to provide a way to understand and analyze the sparse codes with minimal information requirements on the underlying dictionary and the coding algorithm. In this way, we can bring them a step closer to being standalone signal representations and facilitate their use in various settings including cases where it is desirable to minimize the system resources like in sensor networks.

To this end, we use the structure model that we have developed in the Chapter 4 to enable a structured representation of sparse codes. We define a sparse code to be a linear combination of a few molecule realizations based on a set of molecule prototypes. Based on the structural difference measure for molecules that we have defined earlier, we devise a scheme for representing a given code with molecule realizations and learn the corresponding molecule prototypes directly from the sparse codes. Our algorithm requires only minimal knowledge of the underlying dictionary, namely the correlations between features or the matrix of the atoms' pools. As a result, we can recover the signal structure independently of the exact atom form and of the actual sparse coding algorithm that produced the codes.

The rest of the chapter is organized as follows. In Section 5.2 we present our problem formulation for learning the structure from the sparse codes and we propose an optimization solution

that alternates between steps of sparse code representation and structure update to solve it. Then, in Section 5.3 we present our matching pursuit representation algorithm for the sparse codes and in Section 5.4 we give the details of the structure update step. Finally, in Section 5.5 we present the results of our learning scheme and we compare it to traditional dictionary learning techniques to extract the structure from synthetic data, digit and object images.

5.2 Structure learning in the sparse code domain

In this section we present the problem of learning the underlying structure of signals based on their sparse codes. To be more specific, we assume that we have a set of sparse codes X where each sparse code $x \in X$ can be considered as a vector in $\mathbb{R}^N$ with $||x||_0 \leq T_S$ where T_S stands for the maximum allowed sparsity level. We want to learn a set of M molecule prototypes $C_\pi = [c_{\pi,1}\, c_{\pi,2}\, \ldots\, c_{\pi,M}] \in \mathbb{R}^{N\times M}$ such that each sparse code can be represented well as a linear combination of a few molecule realizations. We use the definition of energy-based molecule realizations given in Section 4.2.3 and we adopt the corresponding structural difference introduced in Eq. (4.10) to constrain the molecule realizations to lie close to the molecule prototypes. In other words we want to solve the following problem:

$$\begin{aligned} \hat{C_\pi} \quad = \quad & \underset{C_\pi \geq 0}{\arg\min} \sum_{x\in X} \underset{\substack{a_{x,i}\geq 0,\, c_{x,i}\geq 0 \\ \Delta(c_{\pi,i},c_{x,i})\leq T \\ \Gamma_{x,i}\subseteq\Gamma_x\cap\Gamma_{P_{\pi,i}},\,\forall i\in[1,\ldots,M]}}{\arg\min} ||x - C_x a_x||_2^2 \\ \text{such that} \quad & ||a_x||_0 \leq T_M \end{aligned} \tag{5.1}$$

where $C_x = [c_{x,1}\, c_{x,2}\, \ldots\, c_{x,M}]$ is the matrix of the molecule realizations for signal $x \in X$ and $a_x \in \mathbb{R}^M$ is the vector of corresponding coefficients. The threshold parameters T_M, T permit to control the sparsity of the representation and the flexibility to deformations respectively and they are dependent on the application at hand.

As in the previous chapter, we constrain the representations to be positive, i.e., we assume that the molecule prototypes and their realizations have only non-negative entries and are also combined with non-negative coefficients. The support of each molecule realization $c_{x,i}$, namely $\Gamma_{x,i}$, is constrained by the definition in Section 4.2.3 to be a subset of the union of the active pools in the ith molecule prototype $\Gamma_{P_{\pi,i}} = \bigcup_{k\in\Gamma_{\pi,i}} P(k)$, where $\Gamma_{\pi,i}$ is the support of the ith molecule prototype defined $\Gamma_{\pi,l} = \{d_k \in D,\, c_{\pi,l}(k) > 0\}$. In the learning problem of Eq. (5.1), we constrain the $\Gamma_{x,i}$ further in order to comply with the sparse nature of the code x. To this end, we further restrict it to be a subset of the support of x, Γ_x. When the two conditions are combined, we get $\Gamma_{x,i} \subseteq \Gamma_x \cap \Gamma_{P_{\pi,i}},\ \forall i, x$. In this way, $\Gamma_{x,i}$ is by definition sparse and we no longer need to constrain the l_0 or l_1 norm of $c_{x,i}$ to enforce sparsity. However, we still need a proper norm constraint to control the sparsity of the coefficients a_x. In this formulation, we opt for the l_0 norm, upper bounded by the constant T_M. Finally, to ensure that the $c_{x,i}$'s are proper molecule realizations $\forall i$, we constrain the structural difference $\Delta(c_{\pi,i}, c_{x,i})$ to be small, namely less or equal to a given threshold T.

In Eq. (5.1), we can substitute the structural difference $\Delta(c_{\pi,i}, c_{x,i})$ of Eq. (4.10) to get the following formulation:

$$\begin{aligned} \hat{C_\pi} \quad = \quad & \underset{C_\pi \geq 0}{\arg\min} \sum_{x \in X} \underset{\substack{a_{x,i} \geq 0,\, c_{x,i} \geq 0 \\ ||W_i \times (c_{\pi,i} - S\, c_{x,i})|| \leq T \\ \Gamma_{x,i} \subseteq \Gamma_x \cap \Gamma_{P_{\pi,i}},\, \forall i \in [1,\ldots,M]}}{\arg\min} ||x - C_x a_x||_2^2 \\ \text{such that} \quad & ||a_x||_0 \leq T_M, \end{aligned} \tag{5.2}$$

where W_i is the indicator function of the support $\Gamma_{\pi,i}$ which has been previously defined in Eq. (4.11). Moreover, the matrix S, defined in Eq. (4.8), represents the information about the atoms' pools and it is the only information that we need for the underlying dictionary. With the involvement of W_i and S, the l_2 norm in $\Delta(c_{\pi,i}, c_{x,i})$ adjusts to the core characteristic of the energy-based realizations that is essentially to resemble the corresponding prototype at the pool level instead of the atomic level.

The problem in Eq. (5.2) is highly complicated and non-convex as it requires to solve for the set of code representations for all $x \in X$, namely $AX = \{a_x\}_{x \in X}, CX = \{C_x\}_{x \in X}$ as well as the molecule prototypes C_π. In order to solve it, we adopt the technique of alternating optimization described in Algorithm 3. Alternating optimization [14] is a common approach for treating complex optimization problems over many variables that replaces the difficult joint optimization over all variables with a sequence of easier optimizations involving subsets of the variables. In our case, we divide the learning problem in Eq. (5.2) into two sub-problems: the representation of the sparse codes as linear combinations of molecule realizations given the molecule prototypes and the update of the molecule prototypes based on the codes' representations. The first problem is solved with a matching pursuit algorithm adapted to the nature of the sparse codes and the molecule realizations. The details of the scheme are given in Section 5.3. The second problem, namely the update of the molecule prototypes, is solved with the algorithm presented in Section 5.4. We iterate over these two steps until we reach convergence.

Finally, to check the convergence of the alternating optimization algorithm for the molecule learning we use a slightly modified version of our previously defined structural difference measure $\Delta(c_{\pi,i}, c_{x,i})$ to compare the old and the new molecule prototype structure. Since the measure is not symmetric, given two molecule prototypes $c_{\pi,k}$ and $c_{\pi,m}$, we check both the differences $\Delta(c_{\pi,k}, c_{\pi,m})$ and $\Delta(c_{\pi,k}, c_{\pi,m})$ and average them, i.e., we set the symmetric measure of difference between molecule prototypes to be:

$$\begin{aligned} \Delta_\pi(c_{\pi,k}, c_{\pi,m}) &= \frac{1}{2}(\Delta(c_{\pi,k}, c_{\pi,m}) + \Delta(c_{\pi,k}, c_{\pi,m})) \\ &= \frac{1}{2}(||W_m \times (c_{\pi,m} - S * c_{\pi,k})||^2 + ||W_k \times (c_{\pi,k} - S * c_{\pi,m}||^2) \end{aligned}$$

where $W_k(i) = 1$ if only $i \in \Gamma_{\pi,k}$ and $W_m(i) = 1$ if only $i \in \Gamma_{\pi,m}$ are the indicator functions of the support sets of the two prototypes respectively. In order for the comparison to be meaningful, the two prototypes need to be normalized. In our work, we used the l_2 norm for that purpose

Algorithm 3 Molecule learning in sparse code domain (MLSC)

function LEARNSPARSE($X, S, M, T_A, T_M, T_E, T, \delta$)
  $C_\pi = \mathscr{I}, Stop = 0$ ▷ *Initialize structure and flag*
  while !Stop **do**
    for $x \in X$ **do** ▷ *MP for codes with current structure (5.3)*
      $[a_x, C_x]$ = MPSPARSE($C_\pi, x, S, T_M, T, \delta$) ▷ *Alg. 5*
    end for
    $AX = \{a_x\}_{x \in X}, CX = \{C_x\}_{x \in X}$
    $C_{\pi,N}$ = UPDATESTR($X, AX, CX, S, T_A, T_E, T$) ▷ *Update structure, Alg. 8 (5.4)*
    $diff$ = STRDIFFERENCE($C_\pi, C_{\pi,N}$) ▷ *Compare old and updated structure Alg. 4*
    if $diff < \delta$ **then** $Stop = 1$ **end if** ▷ *If structures similar enough, stop iterations*
    $C_\pi = C_{\pi,N}$
  end while
  return C_π
end function

Algorithm 4 Structural difference between sets of prototypes

function STRDIFFERENCE(G_π, F_π)
  for $i = 1 : |F_\pi|, j = 1 : |G_\pi|$ **do** ▷ *Compute cost for every pair of molecule prototypes*
    $W_{g,i} = g_{\pi,i} > 0, W_{f,j} = f_{\pi,j} > 0$
    $cost(i,j) = \frac{1}{2}\left(||W_{g,i} \times (g_{\pi,i} - S * f_{\pi,j})||^2 + ||W_{f,j} \times (f_{\pi,j} - S * g_{\pi,i})||^2\right)$
  end for
  $[Assign, diff]$ = HUNGARIAN($cost$) ▷ *Find the prototype assignment [68]*
  return $diff, Assign$ ▷ *that minimizes the cost between the two structures*
end function

i.e., we assume $||c_{\pi,m}|| = ||c_{\pi,k}|| = 1$.

Finally, in our learning problem the structure we optimize for is a set of molecule prototypes. In order to compare two sets of molecule prototypes, $G_\pi = \{g_{\pi,i},\ i \in [1,M]\}$ and $F_\pi = \{f_{\pi,i},\ i \in [1,M]\}$, we define:

$$\Delta_{S,\pi}(G_\pi, F_\pi) = \frac{\min_{P_{GF}} \sum_{k=1}^{M} \Delta_\pi(g_{\pi,k},\ f_{\pi,P_{GF}(k)})}{M} \tag{5.3}$$

where $P_{GF} : [1,M] \to [1,M]$ is a bijection that assigns each molecule in G_π to exactly one molecule in F_π. The best P_{GF} for two given sets G_π, F_π can be computed with the Hungarian algorithm [68]. The pseudocode of the function is given in Algorithm 4.

5.3 Sparse code representation

We present now the method used to solve the first step in the structure learning algorithm of Eq. (5.2) where the molecule prototypes are fixed. In this case, the problem in Eq. (5.2) becomes equivalent to representing a sparse code $x \in \mathbb{R}^N$ as a set of molecule realizations

written as:

$$[\hat{a}_x, \hat{C}_x] = \underset{\substack{a_{x,i}\geq 0,\, c_{x,i}\geq 0 \\ ||W_i \times (c_{\pi,i} - S\, c_{x,i})|| \leq T \\ \Gamma_{x,i} \subseteq \Gamma_x \cap \Gamma_{P_{\pi,i}},\, \forall i \in [1,\ldots,M]}}{\arg\min} ||x - C_x a_x||_2^2 \quad \text{such that} \quad ||a_x||_0 \leq T_M \tag{5.4}$$

This problem is similar to the one we have presented in Eq. (4.17) for decomposing a signal into molecule realizations. The main difference relies in the data approximation term where instead of the signal we now have its sparse code x and we do not have any explicit dictionary D. Nevertheless the problem remains non-convex and we could follow the same alternating optimization procedure as the one introduced in Chapter 4.

However, the sparse nature of the codes in combination with their representation as vectors in $\mathbb{R}^N$ where the similarities of the underlying atoms are not taken into account complicates the procedure. For example, in the Algorithm 2 if we try to initialize a_x by solving the problem in Eq. (5.4) with $C_x = C_\pi$, we may get easily a zero vector as solution. This happens when the code x and the set of prototypes in C_π do not have overlapping supports and therefore none of the prototypes can be used to decrease the error in the approximation of x. The code x might have indeed very similar atoms to the ones in the prototypes, but that is not sufficient as atom similarities are ignored when the dictionary D does not appear in Eq. (5.4) .

To overcome this difficulty, we propose a different solution path. The above observation suggests that we need to adjust the molecule prototypes to the code, i.e., solve for the molecule realizations, simultaneously with the coefficients while taking into account atom similarities. This resembles the strategy proposed by matching pursuit algorithms [84, 30, 112]. Therefore, we will follow the same greedy recipe. The basic idea is to start with an empty code representation and a residual code equal to the original sparse code and to pick at each iteration the molecule with the realization that best fits the residual code and remove it form the residual. The iterative decomposition procedure continues until either the maximum number of allowed molecules in the representation T_M is reached, or the residual cannot be reduced anymore. The pseudocode of the scheme is presented in Algorithm 5.

Finding the molecule realization that best approximates the residual code, however, is not trivial as a simple inner product solution between the residual and the molecule prototype would again be challenged by potential non-overlaps in the support. To resolve this issue, we propose a new method for 'projecting' the residual code to the direction of the molecules which takes into account the atoms' pools and the sparse nature of the codes. To this end, in Section 5.3.1 we present the exact problem formulation for the projection. We take some steps into simplifying it before we present the closed form solution to this simplified version in Section 5.3.2.

Algorithm 5 MP for representing sparse codes with molecule realizations (MPSPARSE)

function MPSPARSE($C_\pi, xS, T_M, T, \delta$)
  $r = x, \hat{a}_{x,i} = 0, \forall i \in [1, M]$ ▷ *Initialize residual and code representation*
  for $j = 1 : T_M$ **do** ▷ *Until reaching the maximum number of molecules*
    for $i = 1 : M$ **do** ▷ *Find the projections of the current residual to the molecules*
      $[\hat{a}_{r,i}, \hat{c}_{r,i}, \tilde{r}_i] =$ PROJECT2MOL$(r, c_{\pi,i}, S, T)$ ▷ *Alg. 6*
    end for
    $[min_r, min_i] = \min_i (||\tilde{r}_i||^2)$ ▷ *Find the molecule with the minimum residual*
    if $||r||^2 - min_r < \delta$ **then** ▷ *If not significant reduce in residual, stop*
      $j = j - 1$, BREAK
    end if
    $\hat{a}_{x,min_i} = \hat{a}_{r,min_i}, \hat{c}_{x,min_i} = \hat{c}_{r,min_i}$ ▷ *Otherwise, pick the molecule*
    $r = \tilde{r}_{min_i}$ ▷ *and update residual*
  end for
  return $\hat{a}_x, \hat{C}_x$ ▷ *Return the decomposition coefficients and realizations*
end function

5.3.1 Sparse code projection to molecules

The problem of finding the realization of a molecule prototype $c_{\pi,i}$, $i \in [1, M]$ that can best approximate the residual code r or equivalently the problem of projecting the residual sparse code to the direction of $c_{\pi,i}$, can be written as:

$$\{\hat{a}_{r,i}, \hat{c}_{r,i}\} = \underset{\substack{a_{r,i} \geq 0,\, c_{r,i} \geq 0 \\ ||W_i \times (c_{\pi,i} - S\, c_{r,i})|| \leq T \\ \Gamma_{r,i} \subseteq \Gamma_r \cap \Gamma_{P_{\pi,i}}}}{\arg\min} ||r - c_{r,i}\, a_{r,i}||_2^2 \tag{5.5}$$

where the index i refers to the ith molecule prototype in C_π. For the sake of simplicity, in the rest of this Section we omit the index i and we set:

$$c_\pi = c_{\pi,i}, \quad w = W_i, \quad b = a_{r,i}, \quad v = c_{r,i} \tag{5.6}$$

to be the ith molecule prototype, the indicator function of its support as well as the projection coefficient of the residual to the molecule direction and the realization that achieves it respectively. After the changes in notation, the problem of Eq. (5.5) becomes:

$$\{\hat{b}, \hat{v}\} = \underset{\substack{b \geq 0,\, v \geq 0 \\ ||w \times (c_\pi - S\, v)|| \leq T \\ \Gamma_v \subseteq \Gamma_r \cap \Gamma_{P_\pi}}}{\arg\min} ||r - v\, b||_2^2 \tag{5.7}$$

where Γ_v is the support of v, Γ_r is the support of r and Γ_{P_π} is the union of active pools in the prototype c_π.

This problem is not jointly convex on both variables b, v. Thus, it is not easy to find its global minimum. To avoid getting caught in poor local minima, we carefully simplify the formulation so that the problem becomes convex while remaining true to the underlying structure of the

variables. In Eq. (5.7) the differences between the vectors are measured in two different ways with $||r - v\,b||_2^2$ at the atomic level and $||w \times (c_\pi - S\,v)||$ at the pool level. In particular, the structural difference between the molecule prototype and its realization can be written as :

$$||w \times (c_\pi - S\,v)|| = ||c_w - v_w|| \tag{5.8}$$

where $c_w = w \times c_\pi$ and $v_w = w \times (S\,v)$. Essentially, c_w and v_w are non-zero only at the active entries of W, i.e., the active pools in the prototype c_π. Therefore we could consider them as the restrictions of the corresponding vectors c_π, v in the pool level.

Moreover, based on the constraint $\Gamma_v \subseteq \Gamma_r \cap \Gamma_{P_\pi}$, we observe that for each pool in the prototype c_π we know exactly which atoms to pick in order to create the realization v, namely the common atoms between the support of the residual code Γ_r and the pool. Hence, we are only missing the amount of energy in each active pool. As a result, we can simplify the problem in Eq. (5.7) by first solving it for the vectors with non-zero entries only for the active pools of c_π, marked by w, and then expanding the solution back to the original atomic level with the aid of Γ_r. We denote the restriction of the residual code r to the pool level by $r_w = w \times (S\,r)$. The simplified formulation of Eq. (5.7) is then:

$$\{\hat{b}, \hat{v}_w\} = \underset{\substack{b \geq 0\ v_w \geq 0,\\ ||c_w - v_w|| \leq T}}{\arg\min}\ ||r_w - v_w\,b||_2^2 \tag{5.9}$$

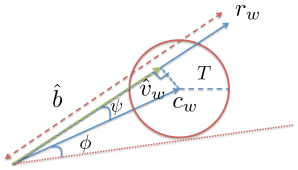
The problem in Eq. (5.9) can be treated as the projection of a point to a convex set. To be more specific, the constraint $||c_w - v_w)|| \leq T$ indicates that v_w belongs to hypersphere H with radius T while the constraint $v_w \geq 0$ restricts v_w to live in the non-negative orthant J. Both H and J are convex sets and so is their intersection $\Omega = H \cap J$. Based on the convexity of Ω and the positivity of the coefficients b, it can be proved that the vector $z = v_w\,b$ belongs to a closed convex cone C_Ω, $\forall\, b \geq 0$, $v_w \in H$. The proof is given in the Appendix A.3. After this observation, the problem can be written as

$$\hat{z} = \underset{z \in C_\Omega}{\arg\min}\, ||r_w - z||_2^2 \tag{5.10}$$

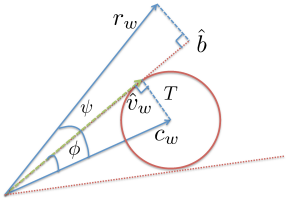
which is exactly the problem of projecting a point to a closed convex set and according to the projection theorem, it is convex and it has a unique solution [18].

5.3.2 Solution for the code projection to molecule direction

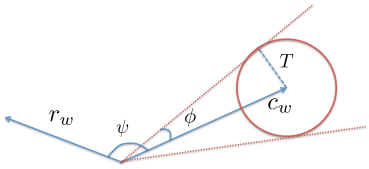
The projection of a point in the intersection of convex sets (POCS) is a well-studied problem [39]. A traditional solution approach suggests to sequentially pass over the individual sets and project onto each one a deflected version of the previous iterate [47]. However, in our case due to the small value of T in applications and the positivity of c_w, it is usually the case that the hypersphere H lies entirely into J, i.e. $\Omega = H \cup J = H$. In the rest, we will make this assumption and find an efficient way to project a point on C_Ω. In the opposite case, a more generic POCS



(a) Case 1 : the residual falls in the area of the cone.



(b) Case 2a: the residual falls outside the cone, non-zero projection.



(c) Case 2b: the residual falls outside the cone, zero projection.

algorithm should be employed instead.

To compute the solution we look at our convex cone C_Ω, which can be alternatively defined as the conical hull of the hypersphere H. As a result, it can also be described as the set of vectors whose angle from c_w is less or equal to ϕ, the maximum between any vector v_w in C_Ω and c_w. The maximum angle ϕ can be computed as $\phi = \cos^{-1}(\frac{\sqrt{||c_w||^2 - T^2}}{||c_w||})$ as it is shown in Appendix A.4. Therefore, we can easily deduce the relative position of r_w with respect to the cone by comparing ϕ with the angle between the vectors r_w and c_w, denoted by ψ. Then we can distinguish two cases, namely r_w is inside the cone C when $\psi <= \phi$ and r_w is outside the cone C when $\psi > \phi$. In the rest, we give the details of the solution in each case.

1. $\psi <= \phi : \mathbf{r_w} \in \mathbf{C}_\Omega$. In this case, the vector r_w is inside the convex cone C. Therefore, the vector $\hat{z}$ that minimizes Eq. (5.10) is equal to r_w. For this case, there are many different decompositions of z as $z = v_w\, b, b > 0, v_w \in H$. For simplicity, we pick v_w to be the projection of c_w in the direction of r_w, i.e., $\hat{v}_w = \frac{\langle c_w, r_w \rangle}{||r_w||^2} * r_w$. Then $\hat{b} = \frac{||r_w||^2}{\langle c_w, r_w \rangle}$. The geometric representation of this case on the plane defined by r_w and c_w is shown in Figure 5.1a.

2. $\psi > \phi : \mathbf{r_w} \notin \mathbf{C}_\Omega$. In this case, the vector $\hat{z}$ has a different direction than the r_w. Due to the symmetry of C around c_w, we can constrain our search for $\hat{z}$ on the plane defined by r_w and c_w. Then we can distinguish two possible cases:

   (a) $\psi - \phi < \pi/\mathbf{2}$: An example of such a case is shown in Figure 5.1b. In this case, $\hat{v}_w$ can be described by its relative position to the vectors r_w and c_w. More specifically we can write:

   $$\begin{aligned} &\hat{v}_w = l_1\, r_w + l_2\, c_w, && v_w \text{ is on the plane defined by } r_w, c_w \\ &||\hat{v}_w|| = \cos\phi, && v_w \text{ is } \phi \text{ radiants from } c_w \\ &\langle \hat{v}_w, r_w \rangle = \cos(\psi - \phi)||r_w||\,||\hat{v}_w||, && v_w \text{ lies } \psi - \phi \text{ radians away from } r_w \end{aligned}$$

   The above equations constitute a 2×2 system system of equations for l_1, l_2 which can be solved by substitution. The exact numeric solution is presented in the pseudocode of the Function FINDV(ψ, ϕ, r_w, c_w) in the Algorithm 7. Finally, the

coefficient $\hat{b}$ is the projection of r_w to the direction of $\hat{v}_w$, i.e., $\hat{b} = \frac{\langle r_w, \hat{v}_w \rangle}{||\hat{v}_w||^2}$.

(b) $\psi - \phi >= \pi/\mathbf{2}$. In this case, shown in Figure 5.1c, $\hat{z} = 0$. Then, $\hat{b} = 0$ and $\hat{v_w} = 0$ too.

Finally, to complete the solution, we need to get the complete vector $\hat{v}$ by expanding its restriction to the pools $\hat{v}_w$ back to the atomic domain. We can do it by iterating over the active pools of c_π. For the jth pool of the prototype c_π, we define $\gamma_j = \Gamma_r \cap P(\Gamma_\pi(j))$ to be the common support between the residual r and the pool. From the jth entry in $\hat{v}_w$ we know the available energy in this pool. In order to mantain the structure in the entries of the residual r, we will enforce the same proportions in the entries of the molecule realization $\hat{v}$. To this end, we set

$$\hat{v}(k) = \begin{cases} \frac{r(k)}{r_w(j)} \hat{v}_w(j) & \text{if } k \in \gamma_j, \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

This way, we have $\frac{\hat{v}(k)}{\hat{v}_w(j)} = \frac{r(k)}{r_w(j)}, \forall k \in \gamma_j, \forall j$. The pseudocode of the expansion procedure is presented Function FINDV(ψ, ϕ, r_w, c_w) in the Algorithm 7. Finally, using the values for $\hat{b}$ and $\hat{v}$ that we have computed for the projection, we can compute the new residual as $\tilde{r} = r - \hat{b} * \hat{v}$.

To sum up, in this section we have presented a greedy matching pursuit algorithm for representing a sparse code as a few molecule realizations. To allow the matching pursuit to adapt to the specific characteristics of the sparse codes and to the molecule structure, we have designed a new scheme for discovering the realization of a molecule prototype that best approximates a residual sparse code. To achieve an effective and efficient solution we simplified the original problem formulation to a form that could be solved accurately based on the geometrical properties of convex sets. The pseudocode of the functions involved in our scheme are presented in the Algorithms 6 and 7 for the PROJECT2MOL and the functions FINDV and EXPANDV respectively.

5.4 Structure update

We consider now the problem of updating the set of molecule prototypes C_π. Since the molecule realizations are strongly connected to their prototypes through the structural constraint $||W_i \times (c_{\pi,i} - S\, c_{x,i})|| \leq T$, $\forall i, x$ we need to update the prototypes and their realizations at the same time to ensure the constraint. To achieve that, we use the code representations discovered so far to update each molecule, both prototype and realizations, alone. Specifically, for each molecule $i \in [1, M]$, we consider that all $c_{\pi, j \neq i}$ and $c_{x, j \neq i}, a_{x, j \neq i}, \forall x$ are known. Then, we solve the problem of Eq. (5.2) for the prototype $c_{\pi,i}$ and the corresponding code representations $c_{x,i}, a_{x,i}$ for all the codes x that use the ith molecule, i.e., $\forall x$, $a_{x,i} > 0$. We denote this set by $X_i = \{x \in X,\ a_{x,i} > 0\}$ and we call it the supporting set of the ith molecule. Then, based

Algorithm 6 Projection to molecule

function PROJECT2MOL(r, c_π, S, T)
  $\hat{b} = 0, \tilde{r} = r, \hat{v} = 0$ ▷ *Initialization*
  $w = c_\pi(k) > 0, \Gamma_\pi = \{k : w(k) == 1\}$ ▷ *Support of the molecule prototype*
  $\Gamma_{P_\pi} = \bigcup_{k\in\Gamma_\pi} P(k)$ ▷ *Support of active pools*
  $r_w = w \times (S\, r),\ c_w = w \times c_\pi$ ▷ *Vectors in pools' level*
  if $r_w == 0$ **then** $return$ **end if** ▷ *If no overlap with molecule prototype, return*
  $\phi = \cos^{-1} \frac{\sqrt{||c_w||^2 - T^2}}{||c_w||},\ \psi = \cos^{-1} \frac{\langle r_w, c_w\rangle}{||r_w||||c_w||}$ ▷ *Angles ϕ, ψ*
  if $\psi < \phi$ **then** ▷ *Case 1*
    $\hat{b} = \frac{||r_w||^2}{\langle r_w, c_w\rangle}$
    $\hat{v}_w = \frac{\langle c_w, r_w\rangle}{||r_w||^2} * r_w$
  else ▷ *Case 2*
    if $\psi < \pi/2$ **then** ▷ *Case 2a*
      $\hat{v}_w = \text{FINDV}(\psi, \phi, r_w, c_w)$ ▷ *Alg. 7*
      $\hat{b} = \frac{\langle r_w, \hat{v}_w\rangle}{||\hat{v}_w||^2}$
    else ▷ *Case 2b*
      return
    end if
  end if
  $\hat{v} = \text{EXPANDV}(\hat{v}_w, \hat{b}, r, \Gamma_\pi)$ ▷ *Alg. 7*
  $\tilde{r} = r - \hat{b}\,\hat{v}$ ▷ *New residual*
  return $\hat{b}, \hat{v}, \tilde{r}$
end function

on Eq. (5.2), our update problem for the ith molecule becomes:

$$\hat{c}_{\pi,i} \quad = \quad \underset{c_{\pi,i}\geq 0}{\arg\min} \sum_{x\in X_i} \underset{\substack{a_{x,i}\geq 0\, c_{x,i}\geq 0 \\ ||W_i\times(c_{\pi,i} - S\, c_{x,i})||\leq T \\ \Gamma_{x,i}\subseteq\Gamma_{e_x}\cap\Gamma_{P_{\pi,i}}}}{\arg\min} ||e_x - c_{x,i}\, a_{x,i}||_2^2 \tag{5.12}$$

where e_x is the code residual with respect to the i-th molecule i.e., the part of the code that is not covered by the other molecules i.e.,

$$e_x = x - \sum_{j\neq i} C_{x,j}\, a_{x,j} \tag{5.13}$$

Moreover, Γ_{e_x} is the support of the residual e_x while $\Gamma_{x,i}$ and $\Gamma_{P_{\pi,i}}$ are the support of the molecule realization $c_{x,i}$ and the union of active pools of the prototype $c_{\pi,i}$ respectively. As in the previous Section, for the sake of simplicity, in the rest of this Section we will omit the index i of the molecule and refer to it as c_π instead of $c_{\pi,i}$. Equivalently, we will set:

$$w = W_i, \quad b_x = a_{x,i}, \quad v_x = c_{x,i}, \quad Q = X_i \tag{5.14}$$

Algorithm 7 Projection to molecule: auxiliary functions

function FINDV(ψ, ϕ, r_w, c_w)) ▷ *Find the vector $\hat{v}_w$ (Case 2a)*
  $A = \cos(\psi - \phi) * \cos\phi * \frac{||c_w||}{||r_w||}$, $B = \frac{\cos\psi * ||c_w||}{||r_w||}$
  $l_2 = \frac{\sin\psi}{\cos\phi \sin(\psi - \phi)}$
  $l_1 = A - l_2 * B$
  $\hat{v}_w = l_1 * r_w + l_2 * c_w$
  return $\hat{v}_w$
end function

function EXPANDV($\hat{v}_w, \hat{b}, r_w, S, \Gamma_\pi$) ▷ *Expand $\hat{v}_w$ to get $\hat{v}$ (Case 2)*
  for $j \in \Gamma_\pi$ **do** ▷ *For every active pool in the prototype*
    $\gamma_j = \{k : r(k) > 0 \wedge S(j,k) > 0\}$ ▷ *Find common support between pool and r*
    if $|\gamma_j| > 0$ **then**
      $\hat{v}(k) = \frac{v_w(j)}{r_w(j)} * r(k), \forall k \in \gamma_j$ ▷ *Distribute energies in the atoms in the pool*
    end if ▷ *while preserving the ratios of energies in r*
  end for
  return $\hat{v}$
end function

to be indicator function of the support of the ith molecule, the coefficient and the realization of the ith prototype for the representation of code x and the support set of the molecule respectively. After the changes in notation the problem (5.12) becomes:

$$\hat{c}_\pi = \underset{c_\pi > 0}{\arg\min} \sum_{x \in Q} \underset{\substack{b_x \geq 0,\, v_x \geq 0 \\ ||w \times (c_\pi - S\, v_x)|| \leq T \\ \Gamma_{v_x} \subseteq \Gamma_{e_x} \cap \Gamma_{P_\pi}}}{\arg\min} ||e_x - v_x b_x||_2^2 \tag{5.15}$$

where Γ_{v_x} is the support of v_x and Γ_{P_π} is the union of active pools in the prototype c_π.

Although updating one molecule at a time results in a simpler problem formulation than the one in Eq. (5.2) that solves for all the molecules simultaneously, the new optimization problem in Eq. (5.15) still presents some challenges. A major one is the dependence of the constraints on the molecule support through both w and Γ_{P_π}. The inclusion of these variables into the problem formulation transforms it into a mixed-integer optimization problem, rendering its exact solution inefficient and time consuming as it can require a full search of the variable space.

To overcome this difficulty, we build an approximate solution to Eq. (5.12) in successive steps. To start with, in Step A, we decide on the support Γ_π of the prototype c_π and we solve for w and Γ_{P_π}. Then, in Step B, we solve for the coefficients of the prototype on the chosen support Γ_π. The two steps are discussed below. The pseudocode of the overall procedure for updating the structure is given in Algorithm 8 while the details for updating each molecule are presented in Algorithm 9.

Algorithm 8 Structure update

function UPDATESTR($X, AX, CX, S, T_A, T_E, T$)
  for $i = 1 : M$ **do** ▷ *Update molecules, one by one*
    $X_i = \{x \in X : a_{x,i} > 0\}$ ▷ *Find codes that use the molecule i*
    if $|X_i| == \emptyset$ **then** ▷ *If molecule not used at all,*
      $X_i = \{x_p, p = randperm(|X|, 1)\}$ ▷ *pick randomly the X_i*
    end if
    $AX_i = \{a_x\}_{x \in X_i}, CX_i = \{C_x\}_{x \in X_i}$
    $[\hat{C}_{\pi,i}, CX_i, AX_i] =$ UPDATEMOL$(X_i, i, CX_i, AX_i, T_A, S, T_E, T)$ ▷ *Update molecule (Alg. 9)*
  end for
  return $\hat{C}_\pi$
end function

Step A: Solve for molecule support

In the first step, we solve for the support of the prototype c_π, denoted as Γ_π. Our solution is based mainly on the constraint $\Gamma_{v_x} \subseteq \Gamma_{E_x} \cap \Gamma_{P_\pi}$, which indicates that the active pools in the molecule prototype Γ_{P_π} should cover as many of the non-zero entries in E_x as possible. Otherwise, the realizations v_x cannot be non-zero in these positions and the approximation error increases. However, given a maximum number of pools that can be included in the prototype, denoted as T_A, deciding which pools to pick to maximize the covered energy in the residuals e_x is an NP-hard problem [106]. An exact solution could be devised with the use of dynamic programming. However, such an approach would not be efficient for our scheme as the algorithm would be invoked multiple times for one structure update, causing an important computational overhead. Instead, we propose a greedy solution that approximates the optimal support Γ_π efficiently through iterations. The algorithm starts with an empty support set and at each iteration adds the most 'energetic' pool to it. Essentially, it follows the same strategy as the matching pursuit algorithms as it tries to pick a few pools to cover most of the coefficients' energy in the residual codes e_x.

To be more specific, we start by setting $r_x = e_x, \forall x \in Q$. Then we project the residuals into the pools using the matrix S and we sum over all the codes in Q to get the total 'energy' in each pool i.e.,

$$E_P = \sum_{x \in Q} S\, r_x \tag{5.16}$$

where S is the matrix representing the atoms' pools, defined in Eq. (4.8). From this vector, we pick the strongest pool to add to the support. Then, we update the residuals r_x by zeroing out the coefficients into the chosen pool, as these entries are now in the allowed support. We iterate until the maximum number of atoms per prototype, namely T_A, is reached or until the energy left in the residual codes r_x's is very small, namely less than a threshold T_E. During the iterations, we also make sure that the chosen support fulfills the molecule

assumption introduced in Section 4.2.3 which states that a molecule prototype should not include overlapping pools. To this end, we introduce a vector of forbidden atoms for the support that is updated at each iteration based on the newly added pool.

Step B: Solve for molecule coefficients

After we decide on the support Γ_π, we solve for the exact coefficients of the prototype c_π. To this end, we use the same method introduced in Section 5.3.1 to simplify the problem of Eq. (5.15) by projecting the vectors to the pool levels. We define:

$$\begin{aligned} c_w &= w \times c_\pi \\ v_{x,w} &= w \times (S\, v_x) \\ e_{x,w} &= w \times (S\, e_x) \end{aligned}$$

In reality, we have $c_w = c_\pi$ since the vector w is the indicator function of c_π. Then the difference between the molecule prototype and its realization is written $||w \times (c_\pi - S\, v_x)|| = ||c_w - v_{x,w}||$ and the problem formulation becomes:

$$\hat{c}_w = \underset{c_w \geq 0}{\arg\min} \sum_{x \in Q} \underset{\substack{b_x \geq 0,\, v_{x,w} \geq 0 \\ ||c_w - S\, v_{x,w})|| \leq T}}{\arg\min} ||e_{x,w} - v_{x,w} b_x||_2^2 \tag{5.17}$$

The optimization problem with unknowns $v_{x,w}, b_x \forall x \in Q$ as well as the vector c_w is not convex. Geometrically, the problem in Eq. (5.17) is equivalent to finding the location of the hypersphere H of radius T that we introduced in Section 5.3.2 so that its conical hull C minimizes the sum of squared distances between the vectors $e_{x,w}$ and their projections on C. However, in contrary to the problem in Eq. (5.9), the one in Eq. (5.17) cannot be solved in a closed form. To avoid tedious alternation techniques, we will provide an approximate solution instead. We can observe that when the value of T is small, which is usually the case, the hypersphere shrinks to a very small area around c_w and the conic hull reduces to the vector c_w. Thus, we can set $c_w \approx v_{x,w} \forall x \in Q$ and Eq. (5.17) becomes:

$$\hat{c}_w = \underset{c_w \geq 0}{\arg\min} \sum_{x \in Q} \underset{b_x \geq 0}{\arg\min} ||e_{x,w} - c_w b_x||_2^2 \tag{5.18}$$

We denote with $B_Q = [b_{x_1} b_{x_2} \dots b_{x_{|Q|}}]$ the vector with the coefficients of all codes in Q and with $E_w = [e_{w,1} e_{w,2} \dots e_{w,|Q|}]$ the matrix with the corresponding error vectors. We can re-write the Eq. (5.18) as :

$$\hat{c}_w = \underset{c_w > 0, B_Q > 0}{\arg\min} ||E_w - c_w B_Q||_F^2 \tag{5.19}$$

If E_w is a non-negative matrix, the problem in Eq. (5.19) is the non-negative matrix factorization of E_w of rank 1. In the opposite case, the negative entries in E_w can not be approximated

due to the positivity constraints on c_w and B_Q. However, we can still solve Eq. (5.19) with NMF [13] after we zero out the negative entries of E_w. Finally, we have $\hat{c}_\pi = \hat{c}_w$ as the vectors c_w and c_π are identical due to $c_w = w \times c_\pi$ where w is the indicator function of the prototype c_π.

Algorithm 9 Molecule update

function UPDATEMOL($Q, i, AX_i, CX_i, T_A, S, T_E, T$)
  $e_x = x - \sum_{j \neq i} C_{x,j} a_{x,j}$ ▷ *Compute error vectors for index set*
  $r_x = e_x, Forb = \emptyset, \Gamma_\pi = \emptyset$ ▷ *Initiliaze residuals, molecule support, allowed pools*
  for $k = 1 : T_A$ **do** ▷ **Step A: support** *(5.4)*
    $E_P = \sum_{x \in Q} S\, r_x$ ▷ *Project residuals in the pools, sum over all instances*
    $[maxV, index] = max(E_P \times forb)$ ▷ *Find the most energetic, allowed pool*
    if $maxV < T_E$ **then** $break$ **end if** ▷ *If not enough energy left, stop*
    $\Gamma_\pi = \Gamma_\pi \cup \{index\}$ ▷ *Add the chosen pool to support*
    $\Gamma_{P_\pi} = \cup_{k \in \Gamma_\pi} P(k)$ ▷ *Find covered pools*
    $r_x(m) = 0, \forall m \in \Gamma_{P_\pi}, \forall x \in Q$ ▷ *Update residuals*
    $forb = \{j : P(j) \cap \Gamma_{P_\pi} \neq \emptyset\}$ ▷ *Update allowed atoms: no overlaps with chosen pools*
  end for ▷ **Step B: coefficients** *(5.4)*
  $w = \mathbb{1}(\Gamma_\pi)$ ▷ *Indicator function of chosen support*
  $e_{w,x} = w \times (S\, e_x)$ ▷ *Restrict errors to chosen support*
  $E_w = [e_{w,x_1} e_{w,x_2} \ldots e_{w,x_{|Q|}}], E_w^+ = max(E_w, 0)$ ▷ *Build error matrix*
  $\hat{c}_\pi = NMF(E_w^+, 1)$ ▷ *Pick the direction for the molecule*
  for $j = 1 : |Q|$ **do** ▷ *Code representations for $x \in Q$*
    $[b_x, v_x]$ = PROJECT2MOL$(e_x, \hat{c}_\pi, S, T)$ ▷ *Algorithm 6*
  end for
  $AX_i = \{b_x,\ x \in Q\}, CX_i = \{v_x,\ x \in Q\}$
  return $\hat{c}_\pi, AX_i, CX_i$
end function

Once $\hat{c}_\pi$ has been completed through steps A and B above, we complete the solution to the problem in Eq. (5.15) and solve for $b_x, v_x\ \forall x \in Q$ with our algorithm that solves and finds the best molecule realization for representing a residual code as described in Algorithm 6.

To sum up, in this Section we have presented our scheme for updating the molecule prototypes after finding the sparse representations of the codes. Faced with a complicated optimization problem in Eq. (5.15), we have designed a series of steps that permits us to find an approximate solution efficiently. In the rest of the Chapter, we present the results of our learning scheme for various data.

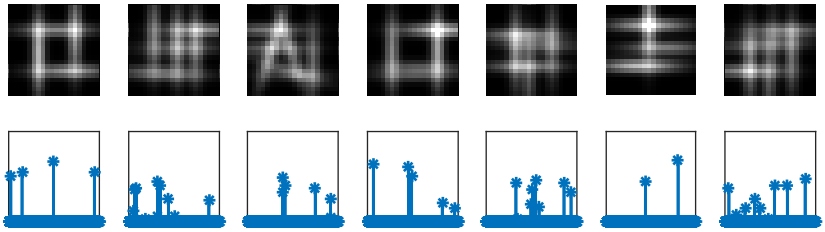


Figure 5.2 – Examples of synthetic signals composed each by 1,2 or 3 molecule realizations. In the first row we have the images and in the second the distribution of energy in the sparse code domain. The corresponding molecule prototypes are shown in Figure 5.3.

5.5 Experimental results

5.5.1 Synthetic Data

Experimental settings

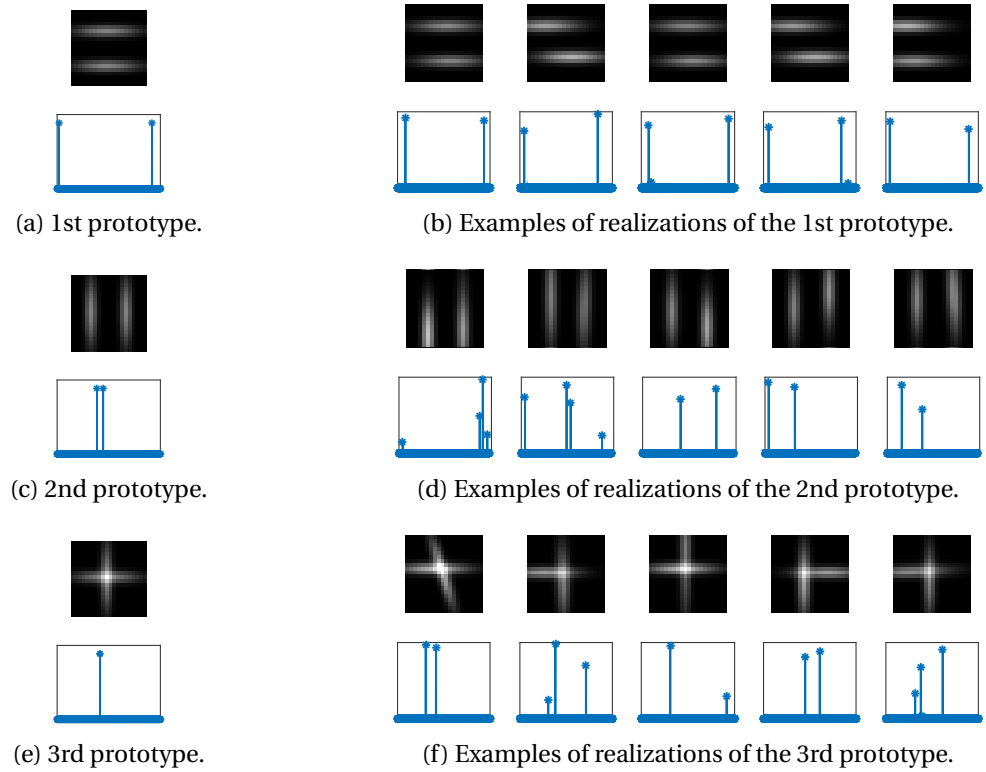
In this section we have experimented with synthetic data to check whether our learning algorithm, namely MLSC, manages to correctly recover the underlying structure when presented with a set of sparse codes. The underlying dictionary consists of gaussian anisotropic atoms i.e.,

$$D = \{\phi_u : u = (\tau_x, \tau_y, r, \sigma) \in U\} \tag{5.20}$$

where $\phi(x,y) = A\exp(-(x/h)^2 - y^2)$ is the gaussian mother function and $\phi_u(x,y) = \phi(x',y')$ with $x' = \cos r\,(x-\tau_x) + \sin r(y-\tau_y)$, $y' = (1/s)(-\sin r\,(x-\tau_x) + \cos r(y-\tau_y))$ is the transformation between the mother function and an atom ϕ_u. In the definition, h stands for the anisotropy level, r is the rotation parameter, τ_x and τ_y denote translations in x and y directions, and σ represents the scale.

The atoms of the dictionary are combined according to M predefined molecules contained in structure matrix C. Each molecule is randomly constructed to contain 2, 3 or 4 atoms of equal energy. To produce a molecule realization we use the same procedure as in the previous chapter to model patterns that are quite similar in structure but may have different sparse representations. In short, for each atom in the molecule prototype we produced an approximation using the atoms in the atom's pool. The atoms are chosen randomly, their total number drawn from a geometric distribution with $p = 0.7$ (so that the approximation is a sparse combination of atoms) while their coefficients are adjusted so that the projection of their combination to the atom direction is close to the original coefficient value.

Each training signal is created as a random combination of a few molecule realizations (2,3 or 4). During the structure learning process, only the corresponding sparse codes of the signals, as well as the matrix S with the atoms' pools is considered to be known, and the knowledge of the complete dictionary D is not necessary. Finally, the values of the parameters T, T_E for



(a) 1st prototype.

(b) Examples of realizations of the 1st prototype.

(c) 2nd prototype.

(d) Examples of realizations of the 2nd prototype.

(e) 3rd prototype.

(f) Examples of realizations of the 3rd prototype.

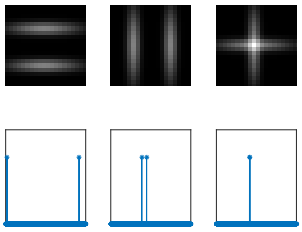
Figure 5.3 – The molecule prototypes for the example described in Section 5.5.1. In each subfigure, the top line shows the images and the bottom the distribution of their coefficients in the sparse code domain. In Figures 5.3a, 5.3c, 5.3e we plot the molecule prototypes while in the Figures 5.3b, 5.3d, 5.3f we have some possible molecule realizations.

MLSC are both set to 0.1 for this set of experiments.

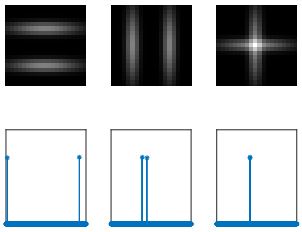
Illustrative example

We first present an illustrative example to highlight the notions of molecule prototypes and realizations. For this case, we use 3 molecule prototypes of 20×20 pixels each, i.e., $M = 3$. The underlying gaussian dictionary D is created by sampling the image plane with a step size 1 for translation and $\frac{\pi}{10}$ for rotation. The anisotropy h in Eq. (5.20) was set equal to 6. The images and the sparse codes of the 3 molecule prototypes are shown in Figure 5.3. Some examples of the training signals composed of 1 to 3 molecule realizations, along with their sparse codes, are shown in Figure 5.2. From the figures, we can observe that the sparse codes of the same molecules can be very different in the various realizations.

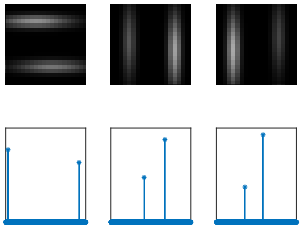
We then apply our MLSC structure learning algorithm on 500 training signals to recover M molecule prototypes. The results of the learning are shown in both the image and code domain in Figure 5.4 along with the results of alternative algorithms, namely the traditional dictionary learning method K-SVD [1] and its recent expansion to sparse structured atoms, the double



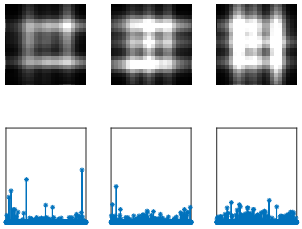
(a) Original molecule structure.



(b) Recovered structure with MLSC.



(c) Recovered structure with DS.



(d) Recovered structure with K-SVD.

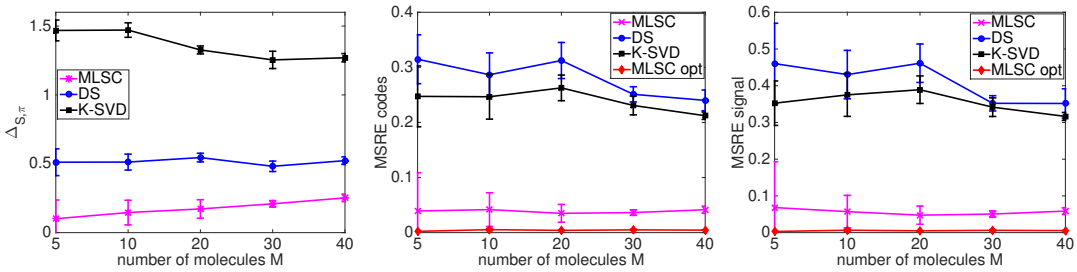
Figure 5.4 – Structure learning results for the example in Section 5.5.1. At the top left we see the original molecule prototypes and then at the top right the molecules learned by our scheme. At the bottom left we have the results of the DS algorithm and finally at the bottom right the results of the K-SVD. In each Figure, the top line shows the images and the bottom shows the distribution of their coefficients in the sparse code domain.

sparsity algorithm (DS) [101]. From this figure, we can observe that K-SVD, as expected, finds molecules that are not sparse since it is not designed to handle constraints on the structure of the atoms. DS on the other hand correctly identifies sparse molecules. However, the lack of the notion of the pools and of the relation between molecule prototypes and realizations, results in mixing the second and third molecule. Our scheme however, correctly identifies the underlying molecules.

Results over different numbers of molecules

Moving a step further, we test our learning scheme on synthetic data built from bigger molecule dictionaries. For this case, we use the same settings as in the previous example, except that the dictionary D has atoms of size 14×14. The anisotropy is set equal to 2 and we have sampled the image plane for two scale levels $[0.5 \quad 1]$ with a step size 2 for translation and $\frac{\pi}{6}$ for rotation.

We have experimented with various values for the number of molecules M in the structure model. For each of the dictionaries we create a training set of sparse codes used for the learning and a testing set used for evaluating the results. As in the previous example, we compare our results with those of double sparsity algorithm (DS) and K-SVD. We measure the performance with the structural difference for molecule sets introduced in Eq. (5.3) and the mean square reconstruction error (MSRE) of the testing set when coded with the learned structures. The MSRE is computed in both the sparse code and the image domain.



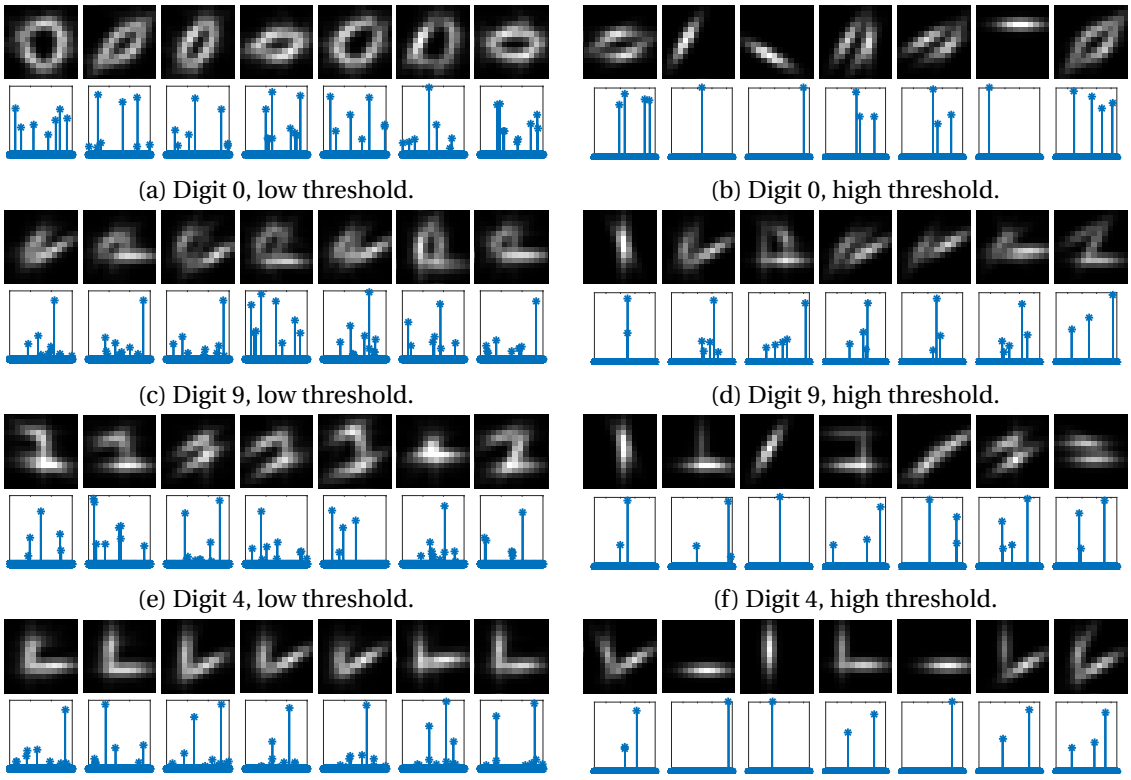
(a) Structural difference (b) MSRE is sparse code (c) MSRE in signal domain

Figure 5.5 – The evaluation of the structures learnt by the different schemes over the number of molecules M in the dictionary for the case of synthetic data. In (a) we plot the structural difference between the learned models and the optimal structure and in (b) and (c) the MSRE in both the sparse code and image domain that is achieved with the learned structures when coding the testing set.

The results of the experiments, averaged over 5 different structure instances for each M, are shown in Figure 5.5. From the plot in Figure 5.5a, we can see that our scheme manages to uncover the structure that is the most relevant one in terms of our measure of structural difference of Eq. (5.3). In the same plot, we observe that K-SVD performs very poorly in terms of this measure, as it is expected since it does not take into account the sparse nature of the molecules. Double sparsity on the other hand, performs better than K-SVD but still worse than our scheme. In the next two plots in Figures 5.5b and 5.5c, we plot the MSRE for the testing set in both the sparse code and image domain. For reference, we also plot the MSRE achieved when the optimal structure is used in the coding. The scheme is denoted as 'MLSC, opt'. The qualitative behavior in both domains is the same for all schemes with our scheme being the closest to the performance of the optimal structure. The interesting twist however is that the MSRE achieved by the structure learnt with K-SVD is a bit better than that of the double sparsity scheme. This observation means that the non-sparse nature of the molecules of K-SVD that is completely wrong in terms of structure, permits a better approximation performance of the signals. The sparse and strict molecules of the DS scheme are more accurate but perform worse in the approximation. Therefore, the two performance measures are not equivalent and satisfying both tasks, namely correct structure and good approximation performance, seems challenging. Luckily, our scheme performs well on both aspects.

5.5.2 Digit images

Next, we have used our algorithm to learn the structure of MNIST images [72]. The images have been downsampled to 14×14 and normalized. In order to better fit the signal model the digits are coarsely pre-aligned to avoid big discrepancies in the position and the orientation. For learning the molecule prototypes we use 1000 examples per digit and for the testing 500 examples per digit. In order to find a sparse representation of the digits, we use again the



(a) Digit 0, low threshold.

(b) Digit 0, high threshold.

(c) Digit 9, low threshold.

(d) Digit 9, high threshold.

(e) Digit 4, low threshold.

(f) Digit 4, high threshold.

(g) Digit 7, low threshold.

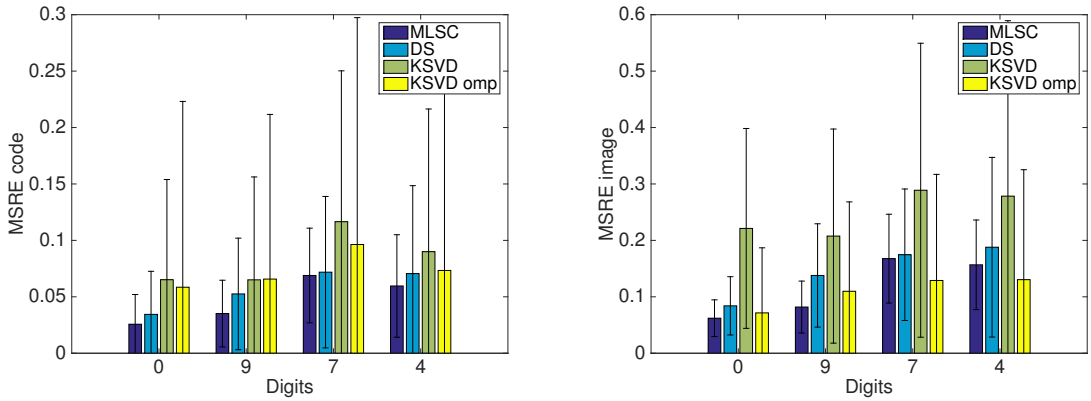
(h) Digit 7, high threshold.

Figure 5.6 – Examples of the molecules learned with MLSC for various digits and different values of the threshold parameter T_E in Algorithm 8. In the top row of each figure we plot the images of the molecules and in the bottom the corresponding distribution of coefficients in the sparse code domain.

dictionary of gaussian anisotropic defined in Eq. (5.20) with two levels of anisotropy h set to 2 and 4 and two scale levels [0.5 1]. The translation parameters τ_x, τ_y are sampled with a step size 1 and the rotation parameter r with step size $\frac{\pi}{6}$.

In a first test, we investigated the influence of the energy threshold T_E in the Algorithm 8 on the sparsity of the learned molecules as well as on their appearance. We have experimented with two different values for T_E a higher one set to 0.1 and a lower one set to 0. The rest of the parameters are set to $M = 20$, $T_A = 10$, $T_M = 5$, $T = 0.2$. The results of the learning for different digits are shown in Figure 5.6. Recall that learning is performed in the sparse code domain, but in order to interpret the results more easily we also plot the corresponding images. As expected when the value of T_E is low, the molecules are less sparse and resemble more images of full digits, encoding more complicated patterns. On the other hand, when T_E is high, fewer atoms are included in the molecules, which end up representing parts of the digits.

In order to quantify the quality of the extracted molecules, we compare their approximation performance in both the sparse code and image domain with the molecules learned from



(a) Sparse code domain.

(b) Image space.

Figure 5.7 – Comparison of the MSRE on the testing set of various digits in both the sparse code and image domain. The 20 molecule prototypes are extracted with 3 different schemes namely MLSC, DS and K-SVD and the coding is performed with the MPSPARSE (Alg. 5) for the first two and the MPSPARSE (Alg. 5) and the OMP for the K-SVD.

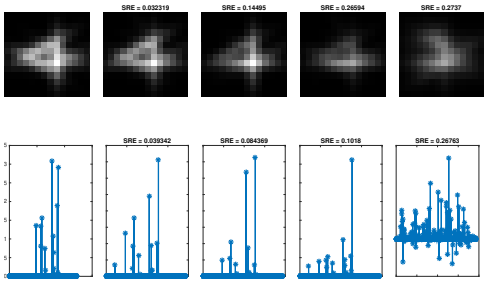
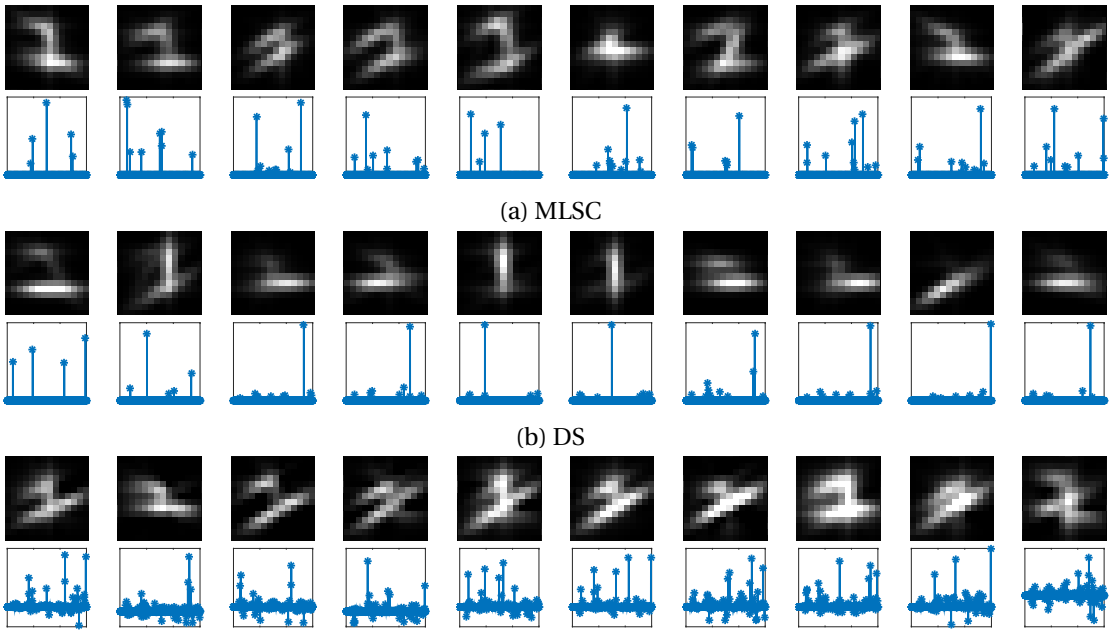


Figure 5.8 – Approximation example for an instance of the digit 4. Starting from the left we have the testing signal, and then its approximation with the structure extracted with the MLSC, the DS and the K-SVD. The square resonstruction error is written on top of each approximation.

DS and K-SVD algorithms. We have kept the same values of parameters M, T_A, T_M and T in all algorithms and we have set T_E in our algorithm to the value allowing the best performance, which proved to be 0. For the K-SVD algorithm, we plot both cases of coding with our MPSPARSE algorithm in Algorithm 5 and the classical OMP as the nature of the molecules extracted by KSVD is not always proper for our scheme (negative values, non-sparse). For the DS, on the other hand, we plot only the results with the MPSPARSE algorithm as the molecules learned with the DS are compatible with our algorithm. The MSRE for the testing set of each digit is shown in Figure 5.7. Moreover, examples of the molecule prototypes included in the dictionaries of the different schemes for the case of the digit 4 are shown in Figure 5.9.

From the Figure 5.7, we can verify that the molecules learned with MLSC, approximate the testing codes better than the molecules learned with the competing schemes in the sparse code domain. However, the MSRE for the corresponding images does not always confirm this



(a) MLSC

(b) DS

(c) K-SVD

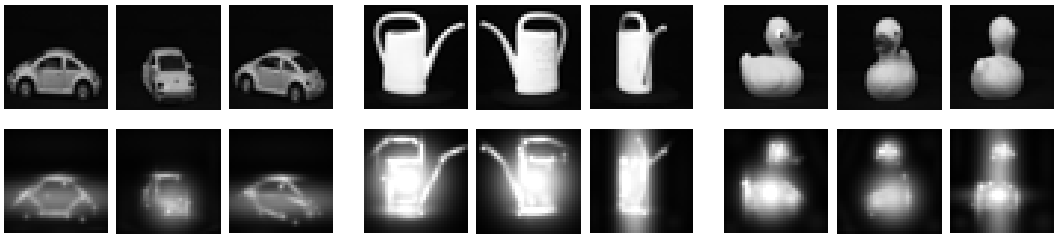
Figure 5.9 – Examples of the molecules learned with the MLSC, DS and the K-SVD for the digit 4. In the top line of each sub-figures we plot the images of the molecule prototypes and in the bottom their sparse codes. These prototypes are part of the structure used to compute the MSRE's in Figure 5.7.

as there are digits where the K-SVD molecules give good approximation error in the image domain. This difference is due to the nature of the K-SVD algorithm that extracts molecules with both positive and negative entries. However, the results of this scheme are highly variable and not very consistent across the testing set.

Moreover, in Figure 5.9 we can see that the molecules learned with MLSC and DS have often similar supports in the sparse code domain. However, the DS molecules have usually a couple of high coefficients while the rest of the support has quite small values in contrast to MLSC that produces more well-balanced molecules. As a result, we can verify that the values of the coefficients is important as it affects significantly the performance. Finally, in Figure 5.8 we present a concrete example of the approximation achieved by the different schemes for an instance of the digit 4 where we observe a degradation of the approximation as we move from MLSC to the others: DS approximation captures less details of the digit while K-SVD approximation is rather blurry and much less well shaped.

5.5.3 Object images

Finally, we have also experimented with images of objects. For that purpose we use the Amsterdam Library of Object images (ALOI) [44] which is a color image collection of one-thousand small objects. Each object is recorded under different viewing angles and illumination settings.



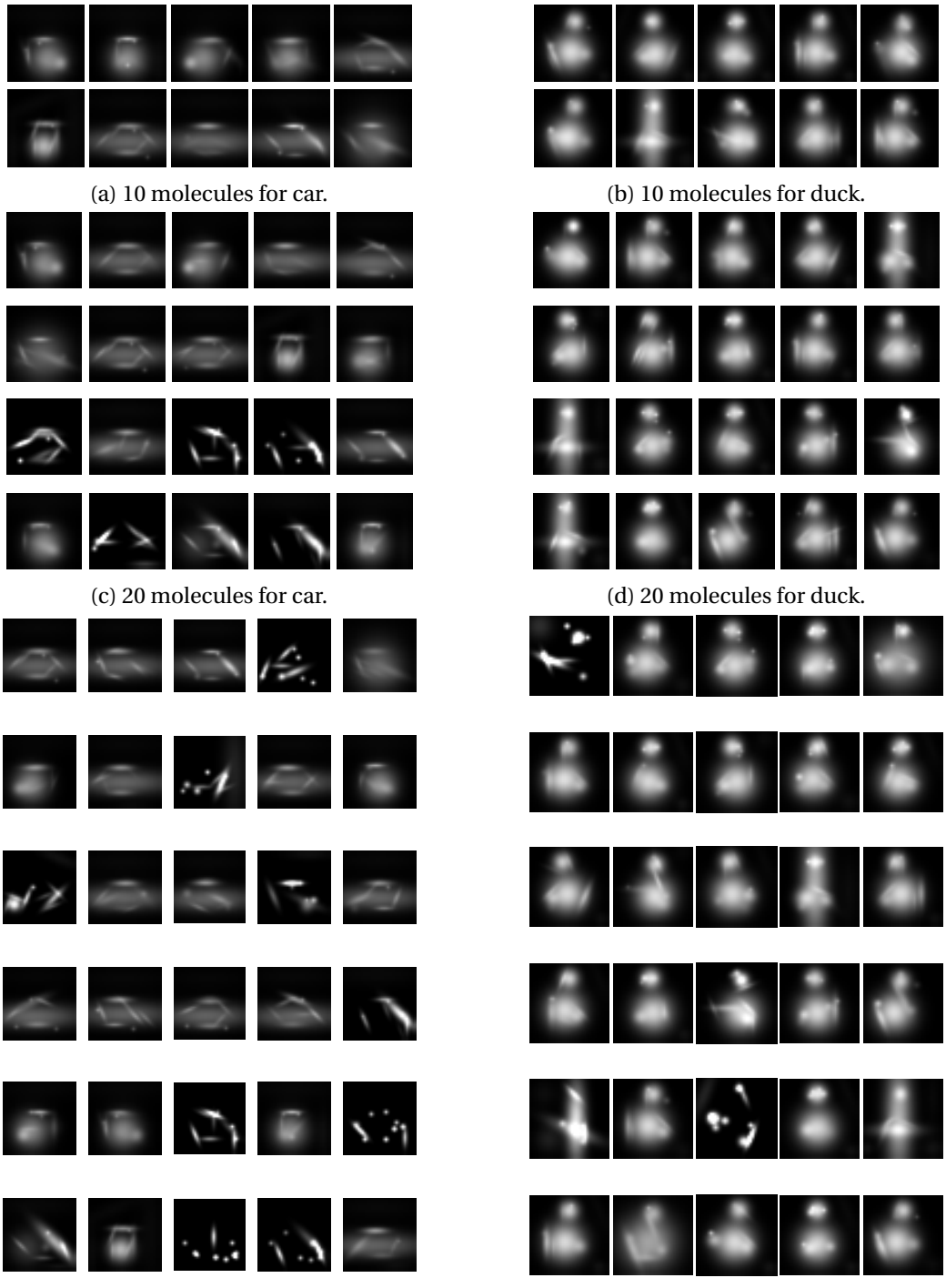
(a) Car (b) Watering Can (c) Duck

Figure 5.10 – Some examples of the images of three objects in ALOI dataset along with their sparse approximation.

In our experiments, we use the grayscale version of the images downsamled to the size 35×35. For the sparse representation of the images we use as before the dictionary of gaussian atoms from Eq. (5.20) with the anisotropy h set to 1 and 4 and the scale s to $2^{[0:0.5:4]}$ as in [41]. The translation parameters τ_x, τ_y are sampled with a step size of 0.5 and the rotation parameter r with $\frac{\pi}{8}$. However, to reduce the computational overhead that a dictionary of that size induces after finding the sparse codes of the images, we discard the atoms of the dictionary that have not been used for sparse coding along with the corresponding dimensions in the codes. Some instances of the objects 'Car', 'Watering Can' and 'Duck' from the dataset are shown in Figure 5.10.

In a first test, we investigate the changes in the appearance of the learned molecules as the number of molecules M in the structure model changes. We try with $M = 10, 20, 30$ for each object and the results for the 'Car' and 'Duck' objects are shown in Figure 5.11. The rest of the parameters are set to $T_A = 10, T_M = 5, T = 0.2, T_E = 0$. As before, the learning is performed in the sparse code domain, but in order to interpret the results more easily we plot the corresponding images as well. From the Figure, we can observe that when M is low, almost all molecules look like blurry shapes of the objects in the different viewpoints. However, as M increases, new molecules handling the details of the objects in the different viewpoints start to emerge allowing for more accurate object approximation.

Finally, as before, we compare the approximation performance of the molecules learned with the MLSC with the ones learned with the DS and the K-SVD algorithms. For the comparison, we keep the same values of parameters T_A, T_M, T and T_E as above and we have set M to 30. Since the number of objects per category is small in the ALOI dataset, we use all the instances for training and we report the MSRE for these images in Figure 5.12. For the K-SVD algorithm, we plot both cases of coding with our MPSPARSE algorithm and the classical OMP as the nature of the molecules extracted by KSVD is not always proper for our scheme (negative values, non-sparse). From the Figure 5.12, we can verify that the molecules learned with our scheme, approximate the testing signals better than the molecules learned with the competing schemes in both domains. This can also be verified from the Figure 5.13 where we show the approximation of a 'Duck' instance from all algorithms in both the image and the sparse



(a) 10 molecules for car.

(b) 10 molecules for duck.

(c) 20 molecules for car.

(d) 20 molecules for duck.

(e) 30 molecules for car.

(f) 30 molecules for duck.

Figure 5.11 – Examples of the molecules learned with our scheme for different number of molecules M in the structure model. In the left side we have the molecules for the object car and in the right side the ones for the object duck.

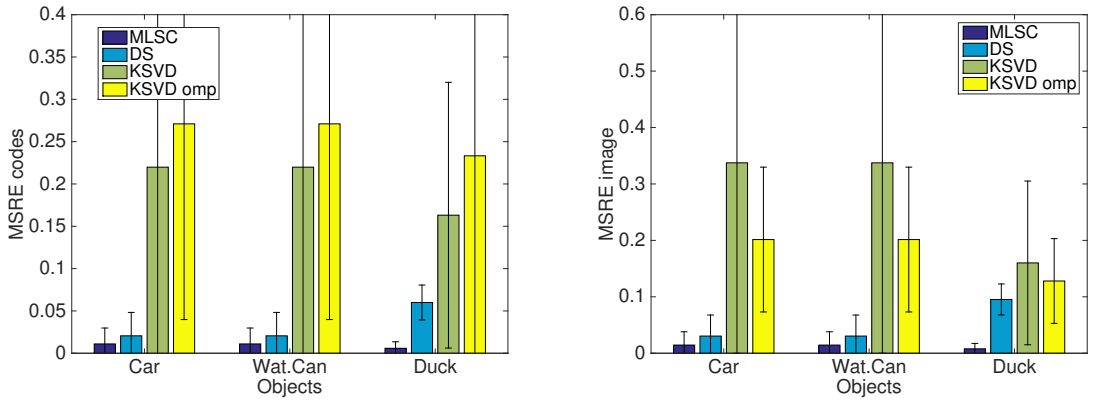


Figure 5.12 – Comparison of the MSRE on ALOI dataset in both the sparse code and image domain. The 30 molecule prototypes are extracted with 3 different schemes namely MLSC, DS and K-SVD and the coding is performed with the MPSPARSE (Alg. 5) for MLSC and DS and with both the MPSPARSE (Alg. 5) and the OMP for the K-SVD.

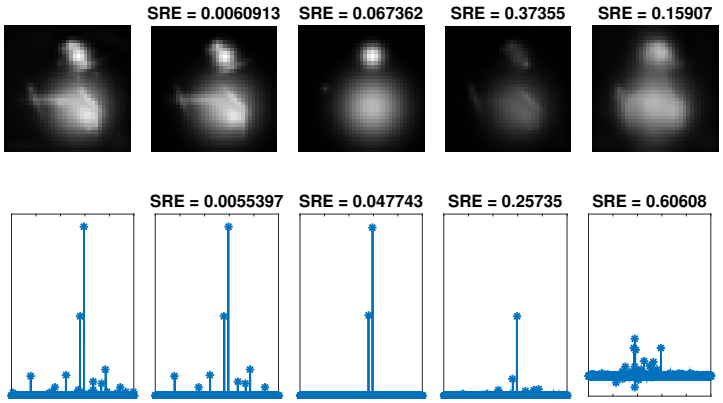


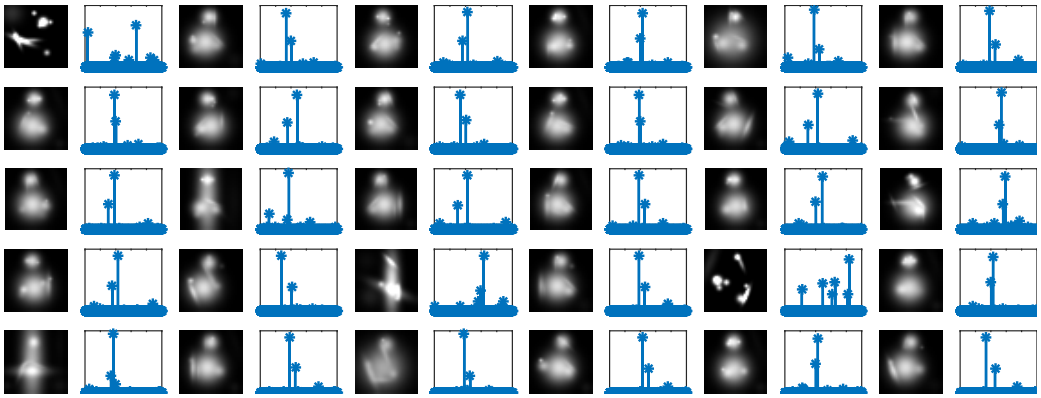
Figure 5.13 – Approximation example for an instance of the object duck.

domain .

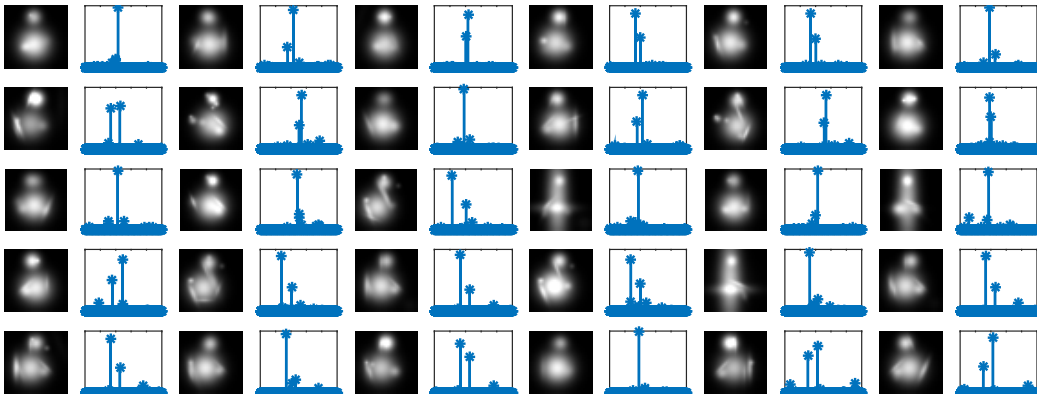
Finally, the molecule prototypes in the dictionaries of the different schemes for the object 'Duck' are shown in Figure 5.14. From the Figure we can observe that with this dataset many of the molecules learned with the MLSC and the DS are similar. However, from the DS molecules we luck the 'details' molecules appear with our scheme as M increases. As a result, the MSRE with the MLSC molecules is better than the one with DS. The same observation can also be made by the approximation example in Figure 5.13: the DS approximation has gotten less details about the duck figure while the SMC approximation is more detailed and accurate.

5.6 Conclusions

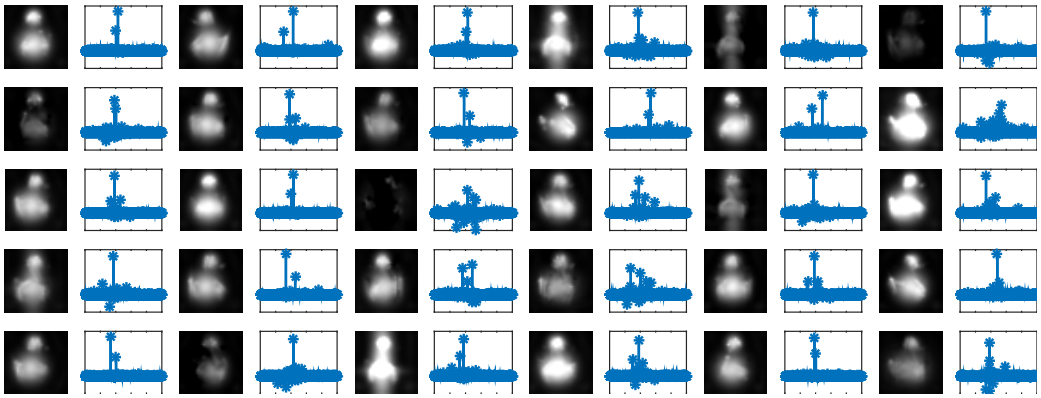
In this chapter, we have presented our algorithm for learning structure from the signals' sparse codes. We have used the structure model that we introduced in the Chapter 4 to formulate the structure learning problem directly in the sparse code domain. In order to deal efficiently with the resulting complex optimization problem we have alternated between steps of finding the representation of the codes based on the current molecule structure and then updating the structure based on the codes' representation. For each step, we have carefully analyzed the simplified optimization problem to get an approximate, yet effective solution. As a result, our scheme is very efficient and it also requires only minimal knowledge about the underlying dictionary, namely the 'correlation' matrix S of the atoms' pools. We have tested our scheme in learning the structure of various datasets like synthetic, digit and object images. From our experiments we have verified the superior performance of our scheme compared to other existing learning techniques that are however not designed explicitly for the sparse domain.



(a) SMC



(b) DS



(c) K-SVD

Figure 5.14 – Molecule dictionaries for $M = 30$ for the different learning schemes for the duck object. In each case, the images of each prototype are plotted along side their sparse representation.

6 Conclusions

6.1 Summary of the thesis contributions

In this thesis we have explored several ways to highlight the structure in low-dimensional signal representations. In particular, we have focused on two popular signal models, namely the manifolds and the sparse representations, and we have proposed techniques to enhance them by either exploring their structural form if already present or by introducing more structure in case of lack.

For the case of manifolds, we have designed a new manifold approximation technique to uncover the structure that is already embedded in the model but it is usually not known explicitly. Motivated by the local linear nature of the manifolds, we have employed affine subspaces, the flats, as approximation functions and we have used the difference of tangents to uncover groups of points that comply with the low dimensionality of the flats. As a result, our algorithm guarantees the preservation of the manifold's local linear structure. Moreover, by using elements from the theory of constrained clustering we have given a thorough justification for the greedy nature of our scheme.

In the case of sparse representations, we have addressed two different issues. First, to enable the differentiation of structures on the same support but with distinct energy distributions, we have focused on defining more informative priors than the traditional ones. To this end, we have proposed a new structured sparsity prior, the molecules, which take into account both the coefficients and the support of the codes. To make our new prior more flexible we have defined both molecule prototypes and molecule realizations, such that the molecules can adjust to the data by getting signal-dependent forms. In order to compare the molecules, we have designed a new structural difference measure based on the comparison of the corresponding sparse codes. Moreover, we have also proposed a sparse coding scheme adapted to our new structure model that permits an effective decomposition of signals into molecule realizations.

Finally, we have used our structure model of molecule prototypes and realizations to formulate the structure learning problem directly in the sparse code domain. Based on the difference

measure between the prototypes and their realizations, we have designed an algorithm that requires only minimal knowledge of the underlying dictionary, namely the matrix of the atoms' pools. As a result, the signal structure can be recovered independently of the exact form of the atoms in the underlying dictionary. Finally, in order to deal efficiently with the complex optimization problem, we have divided in into smaller sub-problems that we have carefully analyzed and simplified to get an approximate but effective solution.

6.2 Discussion

To conclude this thesis, we discuss some observations that we made while working for representation learning with manifolds and sparse priors.

Firstly, we highlight that our approximation scheme and the resulting flats for the manifolds could be seen as a dictionary with group structure. To be more specific, we could build a dictionary D by concatenating the bases for all flats in the approximation. Then, each signal on the manifold has a decomposition using only the base elements of one flat, so it is K-sparse where K is the flat dimensionality. And since only one flat participates in each signal's decomposition, the representation is 1-sparse in the level of groups while inside each group the coefficients are generally non-zero. This is exactly the definition of group sparsity with the $l_1 - l_2$ prior. However, the two models are not exactly equivalent, as the dictionary D can generate more signals than the ones that belong to the manifold. Nevertheless, it is an interesting insight as it also provides a link to our work with molecules as structural elements in dictionaries: we get the molecule prior when we shrink the flats to 1-D hyperlines and we relax the restriction for the signals to belong to only one group.

Secondly, we would like to mention that the manipulation of manifolds poses some challenges. The most fundamental one is the assumption of local smoothness for the manifold. This assumption is reasonable, but it can prove to be quite tricky under the light of the curse of the dimensionality. In high dimensional spaces, the non-linear nature of the manifold may require huge amounts of data to uncover its true underlying structure. In our work, we have used the k-NN nearest neighbor graph to model the manifold structure. This classical technique provides reasonable results in many cases. However, it is not very reliable in high-dimensional spaces and it is very sensitive to noise. And although there has been recently some work on the reliable estimation of the local geometry and the tangent spaces for manifolds [113, 123], the problem still remains open.

Then, we would like to comment shortly on the depth of the architecture that we have used in our structured sparsity model. The quest for depth and multiple layers in representations is not new, as it is inspired from the architecture of many biological systems processing natural stimuli. Many advances in the field have emerged recently in the machine learning community with wide adoption of deep convolutional networks. These models, while achieving impressive performance is some tasks, do unfortunately not provide much insight on how to get a meaningful and effective signal representation. In this thesis, we have tried to follow the trend

for deep data models by defining a second representation level for sparse representations. The components of our model are explicit and intuitive. Although our performance in applications is not as impressive as the deep nets ones, we obtained some promising results. One interesting direction for further exploration therefore consists in the expansion of our structured model to more than one layer. However, since the molecule realizations allow for independent deformations in the atoms of the molecule, the challenge consists in the coordination between the individual molecule deformations so that the pattern modeled by the molecule prototype is not completely deformed into another shape but rather transformed. In other words, the model should be expanded in order to provide transformation invariance.

Seen through the prism of transformation invariance, molecules could be very useful tools for signal analysis as the ability to represent complex patterns in a transformation invariant way is essential for applications like signal classification and recognition. For transformation invariant molecule learning an interesting direction could be the use of Lie operators [94]. Moreover, the molecule learning could be further enhanced by the use of terms that explicitly encourage the molecules' discriminative power with techniques similar to the ones applied in discriminative dictionary learning [97]. Additionally, the underlying dictionary of atoms could also be learned from the data to further improve both the representative and discriminative power of the system. In our work we assumed that the atomic dictionary is given a priori, however a joint learning of the atoms and the molecules is an interesting direction for research.

Finally, in our work we have used a specific difference measure for the sparse codes between molecules and molecule realizations based on our definition of deformations and on the properties of the underlying dictionary. However, the matrix S, modeling the similarities in the pools of the dictionary, could be alternatively learned from the data; that would be particularly useful in case of absence of any information about the dictionary. Moreover, it would allow the matrix to adapt to data by possibly allowing different sizes of pools for the atoms. Additionally, the structural difference measure could also assume different forms where the various methods about metric learning [69, 10] could be employed.

A Supplementary proofs

A.1 Bound on error of atom realization

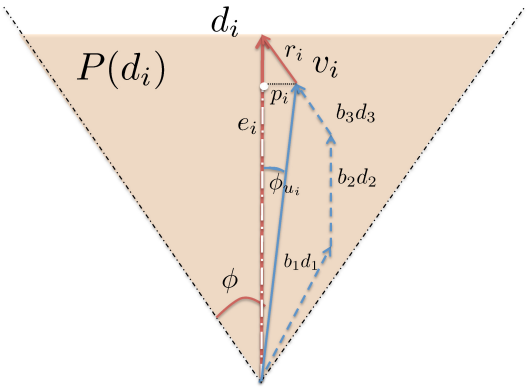


Figure A.1 – An example of the realization of the atom d_i from vector $v_i = b_1d_1 + b_2d_2 + b_3d_3$ with $d_1, d_2, d_3 \in P(d_i)$ and $b_1, b_2, b_3 > 0$.

As we have mentioned in Section 4.2.3, if we constrain the atoms that participate in the realization of the atom d_i to lie in its pool $P(d_i)$ and have non-negative coefficients we can guarantee that the resulting approximation has a bounded error, i.e., $||d_i - v_i||_2^2 \leq L$. To see why, let $v_i = \sum_{j \in P(d_i)} b_j d_j$. Then, from Figure A.1 we have:

$$||d_i - v_i||_2^2 = ||r_i||^2 = ||p_i||^2 + (1 - e_i)^2 = e_i^2 \tan^2 \phi_{u_i} + (1 - e_i)^2 \tag{A.1}$$

However for the angle between v_i and d_i we have:

$$\cos \phi_{u_i} = \frac{\langle v_i, d_i \rangle}{||v_i||} = \frac{\sum_{j \in P(d_i)} b_j \langle d_j, d_i \rangle}{||\sum_{j \in P(d_i)} b_j d_j||} \geq \frac{(1-\epsilon) \sum_{j \in P(d_i)} b_j}{\sum_{j \in P(d_i)} |b_j|} = 1 - \epsilon$$

if $b_j \geq 0, \forall\, j \in P(d_i)$ and $||d_i|| = 1$. Therefore, when we allow only non-negative coefficients in the approximation, v_i belongs to $P(d_i)$.

Moreover, since $\cos \phi_{u_i} \geq 1 - \epsilon$, then $\sin \phi_{u_i} \leq \sqrt{\epsilon(1-\epsilon)}$ and therefore $\tan \phi_{u_i} \leq \sqrt{\frac{\epsilon}{1-\epsilon}}$. Finally,

from Eq. A.1 we get:

$$||d_i - v_i||_2^2 \leq (1-e_i)^2 + e_i^2 \frac{\epsilon}{(1-\epsilon)} \tag{A.2}$$

A.2 Recovery analysis

In this section, we present the theorems that provide the lower and upper bounds on the coherence of dictionaries DC_x and DC_u discussed in Section 4.3. The dictionary DC_x is a dictionary that contains more than one realization per molecule prototype while the dictionary DC_u is restricted to one realization per prototype. To evaluate their coherences denoted as μ_x and μ_u respectively we first need to examine the distance between a molecule prototype $m_{\pi,l} = D\, c_{\pi,l}$ and its possible realizations $m_{x,l} = D\, c_{xi,l}$. The corresponding upper bound is presented in the next Theorem.

Theorem 1
Let $||c_{\pi,l}||_0 \leq n, \forall\, l$ *and* $\phi = acos(1-\epsilon)$ *where* ϵ *is the parameter used in the pool definition in Eq.* (4.6)*. Moreover, let the error* $|c_{\pi,l}(i) - e_i|$ *between the energy in an atom* d_i *of a molecule prototype and the energy on its pool in any of the molecule realizations be bounded by* $|c_{\pi,l}(i) - e_i| \leq E\, c_{\pi,l}(i), \forall\, l, i \in \Gamma_{\pi,l}$*, where* E *is a positive constant. Finally, let* μ_M *stand for the in-molecule coherence defined as the maximum coherence between the atoms that belong to the same molecule, i.e.,* $\mu_M = max_l \left(max_{i,j \in \Gamma_{\pi,l}, i \neq j} | < d_i, d_j > | \right)$ *and assume that* $\mu_M \leq \frac{1}{n-1}$*. Then, the distance between any molecule prototype* $m_{\pi,l}$ *and any of its realizations* $m_{x,l}$ *is bounded by*

$$||m_{x,l} - m_{\pi,l}|| \leq \sqrt{\frac{((1+E)^2\, tan^2\phi + E^2)n}{1-(n-1)\mu_M}}$$

Proof. For the molecule prototype $m_{\pi,l} = \sum_{i \in \Gamma_{\pi,l}} c_{\pi,l}(i) d_i$ a molecule realization can be written as :

$$m_{x,l} = \sum_{i \in \Gamma_{\pi,l}} v_i = \sum_{i \in \Gamma_{\pi,l}} (e_i\, d_i + p_i) = m_{\pi,l} + \sum_{i \in \Gamma_{\pi,l}} \left(p_i - [c_{\pi,l}(i) - e_i]\, d_i \right)$$

where an example of an approximation vector v_i for an atom d_i is shown in Figure A.2. Therefore:

$$||m_{x,l} - m_{\pi,l}|| = || \sum_{i \in \Gamma_{\pi,l}} \left(p_i - [c_{\pi,l}(i) - e_i]\, d_i \right) || \leq \sum_{i \in \Gamma_{\pi,l}} ||p_i - (c_{\pi,l}(i) - e_i) d_i|| \tag{A.3}$$

by the triangle inequality. However, p_i is orthogonal to the direction of d_i. Therefore:

$$||p_i - (c_{\pi,l}(i) - e_i) d_i|| = \sqrt{||p_i||^2 + ||(c_{\pi,l}(i) - e_i) d_i||^2} = \sqrt{e_i^2\, tan^2\phi_v + (c_{\pi,l}(i) - e_i)^2}$$

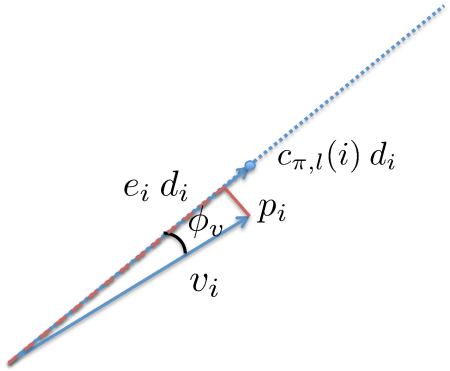


Figure A.2 – An example of the approximation of the atom d_i from vector v_i deviating by ϕ_v in direction. The desired energy level is c_{li} while the projection of v_i gives an energy of e_i.

Substituting in Eq. (A.3), we get:

$$||m_{x,l} - m_{\pi,l}|| \leq \sum_{i \in \Gamma_{\pi,l}} \sqrt{e_i^2 tan^2\phi_v + (c_{\pi,l}(i) - e_i)^2} \leq \sqrt{(1+E)^2 tan^2\phi + E^2}\, ||c_{\pi,l}||_1 \tag{A.4}$$

since $|e_i| \leq E\, c_{\pi,l}(i), \forall l, i \in \Gamma_{\pi,l}$, and $c_{\pi,l}(i) \geq 0, \forall l, i$. For the $||c_{\pi,l}||_1$, given $||c_{\pi,l}||_0 \leq n$, we have :

$$||c_{\pi,l}||_1 \leq ||c_{\pi,l}||_2 \sqrt{n} \tag{A.5}$$

To bound the l_2 norm, we use the Rayleigh quotient $R(M,x) = \frac{x^T M x}{x^T x}$ and its bound $\lambda_{min}(M) \leq R(M,x)$. In our case, $M = D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}}$ where $D_{\Gamma_{\pi,l}}$ is the matrix of the atoms participating in molecule $m_{\pi,l}$. Then, for $x = c_{\pi,l}$ we have :

$$\lambda_{min}(D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}}) \leq \frac{1}{||c_{\pi,l}||^2} \Leftrightarrow ||c_{\pi,l}|| \leq \frac{1}{\sqrt{\lambda_{min}(D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}})}} \tag{A.6}$$

where λ_{min} is the minimum eigenvalue of $D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}}$. Finally, from the Gershgorin circle theorem applied on $D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}}$, which is the Gram matrix of $D_{\Gamma_{\pi,l}}$ that contains the inner products of the atoms in $\Gamma_{\pi,l}$, we get:

$$|\lambda - 1| \leq max_{i \in \Gamma_{\pi,l}} \sum_{j \neq i, j \in \Gamma_{\pi,l}} |< d_i, d_j >|$$

Since $\mu_M = max_l \left(max_{i, j \in \Gamma_{\pi,l}, i \neq j} |< d_i, d_j >| \right)$ we get that $\forall l$:

$$1 - (n-1)\mu_M \leq \lambda_{min}(D_{\Gamma_{\pi,l}}^T D_{\Gamma_{\pi,l}})$$

Assuming $1 - (n-1)\mu_M > 0 \Leftrightarrow \mu_M \leq \frac{1}{n-1}$ and substituting in Eq. (A.6), we have :

$$||c_{\pi,l}|| \leq \frac{1}{\sqrt{1 - (n-1)\mu_M}} \tag{A.7}$$

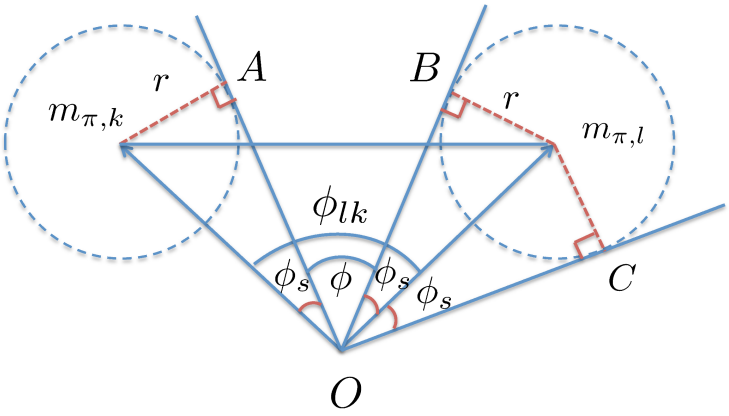


Figure A.3 – The geometry of the molecule prototypes and the region of their realizations restricted on the plane $Om_k m_l$ defined by the center of the axis and the two prototypes. The region of the realizations is restricted by a sphere of radius r. The angle ϕ_s shows the maximum angle between the molecule prototype and any of the realizations, while ϕ is the angle between the two prototypes.

Combining Eq. (A.5),(A.7) and (A.4) we finally get that :

$$||m_{x,l} - m_{\pi,l}|| \leq \sqrt{\frac{((1+E)^2 tan^2\phi + E^2)n}{1-(n-1)\mu_M}}$$

□

With an established bound for the distance $||m_{x,l} - m_{\pi,l}||$ between a molecule prototype and its realizations, we can prove the following theorem which provides a lower bound for the coherence μ_x of any dictionary DC_x with more than one realizations per prototype.

Theorem 2

When the distance between any molecule prototype and its realizations is bounded by $||m_{x,l} - m_{\pi,l}|| \leq r$ *with* $r < \frac{\sqrt{2}}{2}$*, the coherence* μ_x *of any dictionary* DC_x *with more than one molecule realizations per molecule is*

$$\mu_x \geq 1 - 2r^2 = L_x \tag{A.8}$$

Proof. The coherence of the dictionary DC_x is :

$$\mu_x = max_{x,l,y,k} \frac{|< m_{x,l}, m_{y,k} >|}{||m_{x,l}|| * ||m_{y,k}||} = max_{x,l,y,k} |\cos\phi_{m_{x,l},m_{y,l}}|$$

where $m_{x,l}, m_{y,l}$ are realizations of the molecule prototypes $m_{\pi,l}$ and $m_{\pi,k}$ and $\phi_{m_{x,l},m_{y,l}}$ is the angle between the two vectors. A lower bound to $\tilde{\mu}$ can be found by computing the maximum angle between two realizations of the same molecule, i.e. for $l = k$. Then, $\mu_x \geq |max_{x,y} \cos\phi_{m_{x,l},m_{y,l}}|, \forall l$.

From the Figure A.3 we can see that, since all the molecule realizations live in a sphere of radius r around the prototype $m_{\pi,l}$, the angle between any two realizations $m_{x,l}, m_{y,l}$ has to be less than or equal to $2\phi_s$. For the bound to be different than zero, we need that $2\phi_s < \pi/2 \Leftrightarrow r < \sqrt{2}/2$. Then, from the Figure A.3, we have:

$$\cos\phi_s = \frac{||OC||}{||Om_2||} = \frac{\sqrt{1-r^2}}{1} = \sqrt{1-r^2}$$

since $||Om_2|| = ||m_{\pi,l}|| = 1$. Therefore:

$$\begin{aligned}
&\phi_{\phi_{m_{x,l},m_{y,l}}} \leq 2\phi_s \Leftrightarrow \\
&\cos\phi_{m_{x,l},m_{y,l}} \geq \cos 2\phi_s, \quad \phi_s \leq \frac{\pi}{4} \Leftrightarrow \\
&\cos\phi_{m_{x,l},m_{y,l}} \geq 2\cos^2\phi_s - 1, \quad \phi_s \leq \frac{\pi}{4} \Leftrightarrow \\
&\cos\phi_{m_{x,l},m_{y,l}} \geq 2(1-r^2) - 1, \quad r < \sqrt{2}/2 \Leftrightarrow \\
&\cos\phi_{m_{x,l},m_{y,l}} \geq 1 - 2r^2, \quad r < \sqrt{2}/2 \Leftrightarrow \\
&|\cos\phi_{m_{x,l},m_{y,l}}| \geq 1 - 2r^2, \quad r < \sqrt{2}/2 \Leftrightarrow \\
&\mu_x \geq 1 - 2r^2, \quad r < \sqrt{2}/2
\end{aligned} \tag{A.9}$$

□

Finally, we can use the same bound on the distance $||m_{x,l} - m_{\pi,l}||$ between a molecule prototype and its realizations to establish an upper bound on the coherence μ_u of any dictionary DC_u with maximum one realization per prototype. The following theorem formalizes this bound.

Theorem 3
Let the coherence of the molecule prototype dictionary DC be μ. Given the bound on the distance between any molecule prototype and its realizations $||m_{\pi,l} - m_{x,l}|| \leq r$ with $r < \frac{\sqrt{2}}{2}$, the coherence μ_u of any dictionary DC_u with at most one realization per molecule is

$$\mu_u \leq U_u = \mu(1-2r^2) + 2r\sqrt{(1-\mu^2)(1-r^2)} \tag{A.10}$$

Proof. We have:

$$\mu_u = max_{x,y,l,k,l\neq k}\frac{|< m_{x,l}, m_{y,k} >|}{||m_{y,k}|| * ||m_{x,l}||} = max_{x,l,y,k}|\cos\phi_{m_{x,l},m_{y,l}}| \tag{A.11}$$

where $m_{x,l}, m_{y,k}$ are realizations of the molecules $m_{\pi,l}$ and $m_{\pi,k}$ respectively and $\phi_{m_{x,l},m_{y,k}}$ is the angle between the two corresponding vectors. In the rest, we will restrict ourselves to the case where the angle $\phi_{m_{x,l},m_{y,l}}$ that maximizes the Eq. (A.11) is less or equal to $\frac{\pi}{2}$. In the opposite case, a similar analysis can be followed and the final bound on μ_u is the same. Under

this assumption, we have

$$\mu_u = max_{l,k,l\neq k} \cos\phi_{m_{x,l},m_{y,k}} \tag{A.12}$$

Moreover, we can assume that the indices l,k that maximize Eq. (A.12) are the same as the ones that maximize the equation $\mu = max_{l,k}| < m_{\pi,l}, m_{\pi,k} > | = max_{l,k} \cos\phi_{lk}$. In other words, we assume that the molecule prototypes that are the most coherent are also the ones that give rise to the most coherent realizations. Therefore, we will continue our analysis for the case where $\cos\phi_{lk} = \mu$. It is sufficient to restrict the rest of the analysis on the plane defined by the molecules prototypes $m_{\pi,l}, m_{\pi,k}$. This is true because the space occupied by each prototype's realizations is a sphere, and the minimum distance and angle points between spheres live on the plane defined by their centers.

The geometry on this plane is shown in Figure A.3. From the Figure we have that:

$$\phi_{m_{x,l},m_{y,k}} \geq \phi \text{ and } \phi = \phi_{lk} - 2\phi_s$$

Therefore:

$$\phi_{m_{x,l},m_{y,k}} \geq \phi_{lk} - 2\phi_S \Leftrightarrow \cos(\phi_{lk} - 2\phi_S) \geq \cos\phi_{m_{x,l},m_{y,k}} \tag{A.13}$$

Therefore, using Eq. (A.12), we have:

$$\mu_u \leq \cos(\phi_{lk} - 2\phi_S) \tag{A.14}$$

However, from trigonometry we have :

$$\cos(\phi_{lk} - 2\phi_S) = \cos\phi_{lk}\cos 2\phi_S + \sin\phi_{lk}\sin 2\phi_S \tag{A.15}$$

Since $\cos\phi_{lk} = \mu$, we also have $\sin\phi_{lk} = \sqrt{1-\cos^2\phi_{lk}} = \sqrt{1-\mu^2}$. Moreover from the triangle $OCm_{\pi,l}$ we have $\cos 2\phi_S = 1-2r^2$ and $\sin(2\phi_S) = \sqrt{1-\cos^2(2\phi_S)} = \sqrt{1-(1-2r^2)^2} = 2r\sqrt{1-r^2}$. Substituting the above in Eq. (A.15) we get:

$$\cos(\phi_{lk} - 2\phi_S) = \mu(1-2r^2) + 2r\sqrt{(1-\mu^2)(1-r^2)}$$

Substituting this expression in Eq. (A.14), we get :

$$\mu_u \leq \mu(1-2r^2) + 2r\sqrt{(1-\mu^2)(1-r^2)}$$

□

A.3 Proof of convexity of C

We assume closed convex set Ω. We want to prove that the set of points $C = \{z : z = v\, b\}$ with $b \geq 0$, $v \in \Omega$ is a closed convex cone. According to the definition from [18], the set C is a convex cone if for any $z_1, z_2 \in C$ and $a_1, a_2 \geq 0$, we have $a_1\, z_1 + a_2\, z_2 \in C$. Since $z_1, z_2 \in C$, there exists $b_1, b_2 \geq 0$ and $v_1, v_2 \in \Omega$ such that $z_1 = v_1\, b_1$ and $z_2 = v_2\, b_2$. Then, we have

$$\begin{aligned} z_{12} &= a_1\, z_1 + a_2\, z_2 \\ &= a_1\, b_1\, v_1 + a_2\, b_2\, v_2 \\ &= (a_1\, b_1 + a_2\, b_2)\left(\left(1 - \frac{a_1\, b_1}{a_1\, b_1 + a_2\, b_2}\right) v_2 + \frac{a_1\, b_1}{a_1\, b_1 + a_2\, b_2}\, v_1\right) \end{aligned}$$

Since $v_1, v_2 \in \Omega$ and Ω is convex, we have that $v_{12} = \left(1 - \frac{a_1\, b_1}{a_1\, b_1 + a_2\, b_2}\right) v_2 + \frac{a_1\, b_1}{a_1\, b_1 + a_2\, b_2}\, v_1 \in H$. Therefore,

$$z_{12} \in C \text{ as } z_{12} = (a_1\, b_1 + a_2\, b_2)\, v_{12} \text{ with } v_{12} \in H \text{ and } a_1\, b_1 + a_2\, b_2 \geq 0 \tag{A.16}$$

So C is a convex cone. Finally, since the set Ω is closed, so is the cone C.

A.4 Maximum angle

Here, we prove the formula for the maximum angle to center for vectors in a hypersphere. We assume hypersphere H centered at $c \in \mathbb{R}^N$ and with radius T. Then, a vector $v \in \mathbb{R}^N$ belongs to H if and only if

$$||c - v|| \leq T \Leftrightarrow ||c - v||^2 \leq T^2 \tag{A.17}$$

Furthermore, we have $||c - v||^2 = ||c||^2 + ||v||^2 - 2\langle c, v\rangle$. Substituting the inner product formula $\langle c, v\rangle = ||c||\,||v|| \cos\phi$ we get:

$$||c - v||^2 = ||c||^2 + ||v||^2 - 2||c||\,||v|| \cos\phi \tag{A.18}$$

where ϕ is the angle between the vectors c and v. Combining Eq. (A.17) and Eq. (A.18) we get:

$$||v||^2 - 2||c||\,||v|| \cos\phi + ||c||^2 - T^2 = 0 \tag{A.19}$$

The Eq. (A.19) is a quadratic equation for $||v||$. In order for the equation to be feasible, the discriminant needs to be non-negative, i.e.,

$$\begin{aligned} \Delta \geq 0 \Leftrightarrow \\ 4||c||^2 \cos^2\phi - 4(||c||^2 - T^2) \geq 0 \Leftrightarrow \\ \cos^2\phi \geq \frac{||c||^2 - T^2}{||c||^2} \end{aligned} \tag{A.20}$$

From Eq. (A.20) we can observe that if $T > ||c||$ then $\phi \in [0, \pi]$. However, in our problem we assume $T << ||c||$. Therefore, in order for $\Delta \geq 0$ we need

$$\phi \leq \cos^{-1} \frac{||c||^2 - T^2}{||c||^2} \text{ or} \tag{A.21}$$

$$\phi \geq \pi - \cos^{-1}(\frac{||c||^2 - T^2}{||c||^2}) \tag{A.22}$$

However, the solution in Eq. (A.22) is not valid as it produces negative values for $||v||$. Therefore, we are left with the constraint in Eq. (A.21). As a result the maximum angle ϕ between a vector v in the hypersphere H and the hyperspheres centre c is given by the formula

$$\phi = \cos^{-1} \frac{||c||^2 - T^2}{||c||^2} \tag{A.23}$$

Bibliography

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

[2] Ferran Diego Andilla and Fred A Hamprecht. Learning multi-level sparse representations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 818–826, 2013.

[3] V. Batagelj. Agglomerative methods in clustering with constraints. Preprint Series Dept. Math. Univ. Ljubljana, 1984.

[4] Vladimir Batagelj. Constrained clustering problems, IFCS, 1998.

[5] Vladimir Batagelj, Simona Korenjak-Cerne, and Sandi Klavzar. Dynamic programming and convex clustering. *Algorithmica*, 11(2):93–103, 1994.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conf. on Computer Vision*, pages 404–417. Springer, 2006.

[7] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[9] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[10] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[11] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[12] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-François Paiement, Pascal Vincent, and Marie Ouimet. *Spectral dimensionality reduction*. Springer, 2006.

[13] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[14] James C Bezdek and Richard J Hathaway. Some notes on alternating optimization. In *Advances in Soft Computing (AFSS)*, pages 288–300. Springer, 2002.

[15] Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[16] Christopher M. Bishop and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[17] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[18] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[19] Matthew Brand and Matthew Brand. Charting a manifold. In *Advances in Neural Information Processing Systems (NIPS)*, pages 961–968, 2002.

[20] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

[21] Christopher JC Burges. Dimension reduction: A guided tour. *Machine Learning*, 2(4):275–365, 2009.

[22] T T Cai, Lie Wang, and Guangwu Xu. Stable recovery of sparse signals and an oracle inequality. *IEEE Trans. on Information Theory*, 56(7):3516–3522, 2010.

[23] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[24] Emmanuel J. Candes, Michael B. Wakin, and Stephen P. Boyd. Enhacing sparsity by reweighted $l1$ minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[25] R. Cappelli, D. Maio, and D. Maltoni. Multispace KL for pattern representation and classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):977–996, 2001.

[26] Volkan Cevher, Marco F. Duarte, Chinmay Hegde, and Richard G. Baraniuk. Sparse signal recovery using markov random fields. In *Advances in Neural Information Processing Systems (NIPS)*, pages 257–264, 2008.

[27] Jen-Mei Chang. *Classification on the grassmannians: theory and applications*. PhD thesis, 2008.

[28] G. Chen and M. Maggioni. Multiscale geometric and spectral analysis of plane arrangements. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2825–2832. IEEE, 2011.

[29] Guangliang Chen and Gilad Lerman. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[30] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[31] Laurent Daudet. Sparse and structured decompositions of signals with the molecular matching pursuit. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(5):1808–1816, 2006.

[32] Arnaud Delorme, Terrence Sejnowski, and Scott Makeig. Enhanced detection of artifacts in eeg data using higher-order statistics and independent component analysis. *Neuroimage*, 34(4):1443–1449, 2007.

[33] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE Trans. on Image Processing*, 22(4):1620–1630, 2013.

[34] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[35] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[36] Alan Edelman, Tom çs, A. Arias, Steven, and T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[37] Ehsan Elhamifar and Rene Vidal. Sparse manifold clustering and embedding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 55–63, 2011.

[38] Kjersti Engan, Sven Ole Aase, and John Håkon Husøy. Frame based signal compression using method of optimal directions (mod). In *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, volume 4, pages 1–4, 1999.

[39] René Escalante and Marcos Raydan. *Alternating projection methods*, volume 8. SIAM, 2011.

[40] Wei Fan and Dit-Yan Yeung. Locally linear models on face appearance manifolds with application to dual-subspace based classification. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1384–1390, 2006.

[41] Alhussein Fawzi and Pascal Frossard. Image registration with sparse approximations in parametric dictionaries. *SIAM Journal on Imaging Sciences*, 6(4):2370–2403, 2013.

[42] Pierre J. Garrigues and Bruno A. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 505–512, 2008.

[43] Samuel Gerber, Tolga Tasdizen, and Ross Whitaker. Dimensionality reduction and principal surfaces via kernel map manifolds. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 529–536. IEEE, 2009.

[44] Jan-Mark Geusebroek, Gertjan J Burghouts, and Arnold WM Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[45] A. Goh and R. Vidal. Clustering and dimensionality reduction on riemannian manifolds. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2008.

[46] J Hamm and D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Int. Conf. on Machine Learning (ICML)*, pages 376–383. ACM, 2008.

[47] Shih-Ping Han. A successive projection method. *Mathematical Programming*, 40(1-3):1–14, 1988.

[48] Mehrtash T Harandi, Conrad Sanderson, Sareh Shirazi, and Brian C Lovell. Kernel analysis on grassmann manifolds for action recognition. *Pattern Recognition Letters*, 2013.

[49] Harry H Harman. Modern factor analysis. 1960.

[50] Trevor Hastie. *Principal curves and surfaces*. PhD thesis, 1984.

[51] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[52] J. Ho, Ming-Husang Yang, Jongwoo Lim, Kuang-Chih Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–11, 2003.

[53] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[54] Junzhou Huang, Tong Zhang, et al. The benefit of group sparsity. *Annals of Statistics*, 38(4):1978–2004, 2010.

[55] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *The Journal of Machine Learning Research*, 12:3371–3412, 2011.

[56] Ke Huang and Selin Aviyente. Sparse representation for signal classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 609–616, 2006.

[57] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.

[58] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Int. Conf. on Machine Learning (ICML)*, pages 433–440, 2009.

[59] Kevin Jarrett, Koray Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *IEEE Int. Conf. on Computer Vision*, pages 2146–2153, 2009.

[60] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

[61] Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Int. Conf. on Machine Learning (ICML)*, pages 487–494, 2010.

[62] Robert E. Jensen. A dynamic programming algorithm for cluster analysis. *Operations Research*, pages 1034–1057, 1969.

[63] Jongeun Jun, Seokkyung Chung, and Dennis McLeod. Subspace clustering of microarray data based on domain transformation. In *Data Mining and Bioinformatics*, pages 14–28. Springer, 2006.

[64] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977.

[65] Sofia Karygianni and Pascal Frossard. Structured sparse coding for image denoising or pattern detection. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3533–3537, 2014.

[66] Koray Kavukcuoglu, M Ranzato, Rob Fergus, and Yann LeCun. Learning invariant features through topographic filter maps. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1605–1612, 2009.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[68] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[69] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[70] Quoc V Le. Building high-level features using large scale unsupervised learning. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8595–8598, 2013.

[71] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[72] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits.

[73] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808, 2006.

[74] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Int. Conf. on Machine Learning (ICML)*, pages 609–616, 2009.

[75] Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

[76] Hwasup Lim, Octavia Camps, Mario Sznaier, Vlad Morariu, et al. Dynamic appearance modeling for human tracking. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 751–757. IEEE, 2006.

[77] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156, 1998.

[78] Han Liu, Mark Palatucci, and Jian Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Int. Conf. on Machine Learning (ICML)*, pages 649–656. ACM, 2009.

[79] David G Lowe. Object recognition from local scale-invariant features. In *IEEE Int. Conf. on Computer Vision (ICCV)*, volume 2, pages 1150–1157. Ieee, 1999.

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

[81] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2272–2279. IEEE, 2009.

[82] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Trans. on Image Processing*, 17(1):53–69, 2008.

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

[84] Stephane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, 41(12):3397–3415, 1993.

[85] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE Int. Conf. on Computer Vision (ICCV)*, volume 2, pages 416–423, 2001.

[86] Peter Meinicke, Stefan Klanke, Roland Memisevic, and Helge Ritter. Principal surfaces from unsupervised kernel regression. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(9):1379–1391, 2005.

[87] Mark S Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision*. Academic Press, 2012.

[88] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1. *VisR*, 37(23):3311–3325, 1997.

[89] Tomer Peleg, Yonina C. Eldar, and Michael Elad. Exploiting statistical dependencies in sparse representations for signal recovery. *IEEE Trans. on Signal Processing*, 60(5):2286–2303, 2012.

[90] Nikolaos Pitelis, Craig Russell, and Leobelle Agapito. Learning a manifold as an atlas. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1642–1649. IEEE, 2013.

[91] Robert Pless and Richard Souvenir. A survey of manifold learning for images. *IPSJ Trans. on Computer Vision and Applications*, 1(0):83–94, 2009.

[92] Matan Protter and Michael Elad. Image sequence denoising via sparse and redundant representations. *IEEE Trans. on Image Processing*, 18(1):27–35, 2009.

[93] Karthikeyan Natesan Ramamurthy, Jayaraman J. Thiagarajan, and Andreas Spanias. Improved sparse coding using manifold projections. In *IEEE Int. Conf. on Image Processing*, pages 1237–1240, 2011.

[94] Rajesh PN Rao and Daniel L Ruderman. Learning lie groups for invariant visual perception. *Advances in Neural Information Processing Systems (NIPS)*, pages 810–816, 1999.

[95] Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2294–2302, 2011.

[96] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Int. Conf. on Machine Learning (ICML)*, pages 833–840, 2011.

[97] Fernando Rodriguez and Guillermo Sapiro. Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries. Technical report, DTIC Document, 2008.

[98] Jason Tyler Rolfe and Yann LeCun. Discriminative recurrent sparse auto-encoders. *arXiv preprint arXiv:1301.3775*, 2013.

[99] Sam Roweis, Lawrence K. Saul, and Geoffrey E. Hinton. Global coordination of local linear models. *Advances in Neural Information Processing Systems (NIPS)*, 2:889–896, 2002.

[100] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[101] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE Trans. on Signal Processing*, 58(3):1553–1564, 2010.

[102] Conrad Sanderson. *Biometric person recognition: Face, speech and fusion*. VDM Publishing, 2008.

[103] Mahdi Soltanolkotabi, Ehsan Elhamifar, Emmanuel J Candes, et al. Robust subspace clustering. *Annals of Mathematical Statistics*, 42(2):669–699, 2014.

[104] Richard Souvenir, Qi Zhang, and Robert Pless. Image manifold interpolation using free-form deformations. In *IEEE Int. Conf. on Image Processing*, pages 1437–1440. IEEE, 2006.

[105] Michael Spivak. *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*. Addison-Wesley, 1965.

[106] Clifford Stein, T Cormen, R Rivest, and C Leiserson. *Introduction to algorithms*, volume 3. MIT Press Cambridge, MA, 2001.

[107] Jacques JA Tacq and Jacques Tacq. *Multivariate analysis techniques in social science research: From problem to analysis*. Sage, 1997.

[108] Joshua B. Tenenbaum, Vin Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[109] Jayaraman J Thiagarajan, Karthikeyan Natesan Ramamurthy, and Andreas Spanias. Learning stable multilevel dictionaries for sparse representation of images. *IEEE Trans. on Neural Networks and Learning Systems*, 9:1913–1916, 2015.

[110] Louis Leon Thurstone. Multiple factor analysis. 1947.

[111] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, pages 267–288, 1994.

[112] Joel A. Tropp and Anna C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. on Information Theory*, 53(12):4655–4666, 2007.

[113] Hemant Tyagi, Elıf Vural, and Pascal Frossard. Tangent space estimation for smooth embeddings of riemannian manifolds. *Information and Inference*, 2(1):69–114, 2013.

[114] L. J. P. Van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. 2007.

[115] René Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2010.

[116] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (GPCA). *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.

[117] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[118] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511. IEEE, 2001.

[119] Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[120] Ruiping Wang and Xilin Chen. Manifold discriminant analysis. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 429–436, 2009.

[121] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent laboratory systems*, 2(1):37–52, 1987.

[122] Y. C. Wong. Differential geometry of grassmann manifolds. *Proceedings of the National Academy of Sciences*, 57(3):589, 1967.

[123] He Xiaofei and Lin Binbin. Tangent space learning and generalization. *Frontiers of Electrical and Electronic Engineering in China*, 6(1):27–42, 2011.

[124] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conf. on Computer Vision*, pages 94–106, 2006.

[125] Ming Yuan, Ming Yuan, Yi Lin, and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, 68(1):49–67, 2006.

[126] L. Zappella, X. Lladó, E. Provenzi, and J. Salvi. Enhanced local subspace affinity for feature-based motion segmentation. *Pattern Recognition*, 44:454–470, 2011.

[127] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conf. on Computer Vision*, pages 818–833. Springer, 2014.

[128] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2018–2025, 2011.

[129] Teng Zhang, A. Szlam, and G. Lerman. Median k-flats for hybrid linear modeling with many outliers. In *IEEE Int. Conf. on Computer Vision (ICCV) Workshops*, pages 234–241, 2009.

[130] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, pages 3468–3497, 2009.

[131] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[132] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495, 2014.

Sofia Karygianni

Druey 14 B
1018 Lausanne
+41 76 201 9380
sofikari@gmail.com
lts4.epfl.ch/karygianni

Key Strengths

- Proven analytical skills on signal modeling with strong programming experience.
- In depth knowledge of signal processing, pattern recognition and machine learning.

Education

Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
Doctor of Philosophy (Ph.D.) in Electrical Engineering *2010 - Present*
Thesis: 'Signal Structure: from manifolds to molecules and structured sparsity'
Expected graduation: October 2015

University of Toronto (UoT), Canada
Master of Science in Computer Science *2007 - 2009*
Thesis: 'Face tracking in video sequences based on multiple local features and high-light free color information'
Connaught Tuition award, Wolfond Scholarship

National Technical University of Athens (NTUA), Greece
Diploma in Electrical and Computer Engineering *2002-2007*
Thesis: 'Study of the problems in the complexity class $\sharp P$ and analysis of their approximation schemes'
GPA: 9.62/10, university admission scholarship (ranked 1st)

Professional Experience

Signal Processing Laboratory (LTS4), EPFL
Research assistant *Jul 2010–Present*
I worked on efficient data manipulation by detecting, analyzing and modeling the underlying structure. Specifically:
- I created a **novel signal model** to account for complex patterns in images.
- I designed an algorithm for extracting the **data structure** from the signals' representations.

Extensive evaluation of the algorithmic scheme proved its superior performance to state-of-the-art techniques.

Signal Processing Laboratory (LTS4), EPFL
Research Intern *Jan - Jul 2010*
I developed an algorithm to facilitate the use of a complex mathematical model, the manifold, in practice. The results of my project were published in one of the major journals in our field.

NTUA, UoT, EPFL
Teaching Assistant *2005-2010*
Courses:
- Image Communication (EPFL)
- Introduction to Programming (UoT)
- Data Structures and Analysis (UoT)
- Programming Techniques (NTUA)

Skills

Computer:

- Matlab, C, C++, Python, SQL, Linux

Language:

- English: Fluent
- French: Intermediate (reading,writing), Basic (speaking)
- Greek: Mothertongue

Extracurricular activities

- Keeping the balance between the body and the mind is important to me. My favorite activity is pilates as it focuses on the core of the body with very precise and accurate movements. I also enjoy tennis and skiing.
- My friends are also a valuable part of my life. Together we cook, organize fun activities, discuss and laugh.

Personal Information

31, single, Greek Citizenship, Swiss C Permit

Publications

A full list of publications is available on my personal webpage: lts4.epfl.ch/karygianni