Towards Understanding Sparse Filtering: A Theoretical Perspective

Fabio Massimo Zennaro ZENNAROF@CS.MANCHESTER.AC.UK
Ke Chen CHEN@CS.MANCHESTER.AC.UK
School of Computer Science
The University of Manchester
Manchester, M13 9PL, UK

Editor:

Abstract

In this paper we present our study on a recent and effective algorithm for unsupervised learning, that is, sparse filtering. The aim of this research is not to show *whether* or *how well* sparse filtering works, but to understand *why* and *when* sparse filtering does work. We provide a thorough study of this algorithm through a conceptual evaluation of feature distribution learning, a theoretical analysis of the properties of sparse filtering, and an experimental validation of our conclusions. We argue that sparse filtering works by explicitly maximizing the informativeness of the learned representation through the maximization of the proxy of sparsity, and by implicitly preserving information conveyed by the distribution of the original data through the constraint of structure preservation. In particular, we prove that sparse filtering preserves the *cosine neighborhoodness* of the data. We validate our statements on artificial and real data sets by applying our theoretical understanding to the explanation of the success of sparse filtering on real-world problems. Our work provides a strong theoretical framework for understanding sparse filtering, it highlights assumptions and conditions for success behind the algorithm, and it provides a fresh insight into developing new feature distribution learning algorithms.

Keywords: sparse filtering, feature distribution learning, soft clustering, information preservation, cosine metric

1. Introduction

Unsupervised learning can be conceptualized as the area of research within machine learning concerned with the *problem of modeling data*. This problem is stated as the problem of learning a transformation which maps data in a given representation onto a new representation. Contrasted with supervised learning, where we are provided labels and we learn a (logic, causal or stochastic) relationship between the data and the labels, unsupervised learning can not rely on any external semantics in the form of labels. Thus, in order to direct learning, unsupervised learning must rely on the specification of assumptions and constraints. These assumptions and constraints translate our very understanding of the problem of modeling data and our intuition on how useful or meaningful mappings may be learned (for example, if we judge that a useful representation of the data would be provided by grouping together the instances according to a specific metric, then we may employ a clustering algorithm to generate one-hot representations of the data).

Often, the tacit aim of unsupervised learning is to generate representations of the data that may next simplify the problem of learning meaningful relationships through supervised learning. Coates et al. (2011) clearly showed that very simple unsupervised learning algorithms (such as k-means clustering), when properly tuned, can generate representations of the data that allow even basic supervised classifiers, such as support vector machines, to achieve state-of-the-art performances.

One common assumption hard-wired in several unsupervised learning algorithms is *sparsity* (Bengio et al., 2013). Sparse representation learning aims at learning a mapping that produces new representations where few of the components of each new representation are active while all of the others are reduced to zero. The adoption of sparsity relies both on theoretical justifications and on biological analogies. From a formal perspective, sparse representations provide a good trade-off between robustness, maximized by one-hot representations, and representative power, maximized by dense representations (Bengio, 2009); they may be insensitive to irrelevant perturbations of the inputs (Bengio et al., 2013); they may deal with explaining-away phenomena (Bengio et al., 2013); they may improve pattern discrimination (Goh et al., 2014); they may help tackling the curse of high dimensionality (Ganguli and Sompolinsky, 2012); and, sparsity allows high levels of compression within the compressed sensing framework (Candes et al., 2006). From a physical and biological perspective, sparse representations are energy efficient; they are parsimonious and can enhance storage capacity (Rolls and Treves, 1990); several physical phenomena may be encoded with a sparse representation in a proper domain (Ganguli and Sompolinsky, 2012); the visual cortex (Olshausen and Field, 1997) and the brain cortex in general (Földiák and Young, 1995) seem to rely on sparse codes; and, sparse codes make it possible to control the variability and the overlap of input stimuli to the brain (Babadi and Sompolinsky, 2014). Beyond these rationales, the usefulness of sparsity has been confirmed through numerous practical implementations; see, for instance, Bengio et al. (2013), Coates and Ng (2011) and Ranzato et al. (2006). Several different algorithms have been developed or have been adapted to learn sparse representations; for a recent survey of these algorithms we refer the reader to Zhang et al. (2015).

1.1 Sparse Filtering and Related Work

In 2011, Ngiam et al. (2011) proposed an original unsupervised learning framework for generating sparse representation.

First, they put forward the idea and the intuitive definition of an alternative way to perform unsupervised learning. Most of the successful unsupervised learning algorithms may be described as *data distribution learning* algorithms, that is, algorithms that try to learn new representations which better model the underlying probability distribution that generated the data. In contrast, they proposed the possibility of developing *feature distribution learning* algorithms, that is, algorithms which try to learn a new representation having desirable properties, without taking into account the problem of modeling the distribution of the data.

Consistent with the feature distribution learning framework, they defined a concrete algorithm named *sparse filtering*. Sparse filtering ignores the problem of learning the data distribution and instead focuses only on optimizing the sparsity of the learned representations. Sparse filtering proved to be an excellent algorithm for unsupervised learning: it is extremely simple to tune, since it is "essentially hyperparameter-free", having only a single hyper-parameter to select; it scales very well with the dimension of the input; it is easy to implement; and, more importantly, it was shown to achieve state-of-the-art performance.

Because of its simplicity and its performance, the sparse filtering algorithm has been studied and adopted by other researchers. In the original paper, Ngiam et al. (2011) showed that sparse filtering was able to provide state-of-the-art performance on image recognition and phone classification. Thaler (see Goodfellow et al., 2013) and Romaszko (2013) used sparse filtering in their machine learning systems while taking part into the Kaggle Black Box Learning Challenge, achieving respectively the first and the sixth best positions. These results spurred interest in sparse filtering.

Indeed, more theoretical and experimental studies were published: Lederer and Guadarrama (2014) proposed an improved stopping criterion for sparse filtering when processing images relying on results from random matrix theory; Zhang et al. (2014) published an experimental comparison of six sparse coding algorithms, including sparse filtering; Romero et al. (2014) introduced a new algorithm inspired by sparse filtering with no meta-parameters; and Yang et al. (2014) modified the original sparse filtering algorithm by introducing a penalty on the weight matrix. These studies highlight a clear interest in the refinement and improvement of the original algorithm.

At the same time, on the practical side, the simplicity of the sparse filtering algorithm favored its adoption in many real-world applications: Raja et al. (2015) deployed it in a system for iris recognition on smartphones; Dong et al. (2014) and Dong et al. (2015) adopted it in a system for vehicle type recognition; Hahn et al. (2015) used sparse filtering for detecting human actions from videos; Yan et al. (2015) implemented it in their system for detecting driving posture; Lei et al. (2015) integrated it in a system for intelligent fault diagnosis from big data collected from mechanical apparatus; Gu et al. (2014) implemented sparse filtering in a system for the assessment of image quality; Han and Lee (2014) and Han and Lee (2016) used it in a system for detecting mistakes and overblowing in flute playing; Si et al. (2015) introduced it in their system for estimating high resolution images from low resolution images; Sun et al. (2016) implemented sparse filtering in their system for hyper-spectral anomaly detection; and, Ryman et al. (2016) integrated it in their system for modeling the olfactive temporal response in arrays of chemically diverse sensors. All of these applications bear witness to the usefulness of sparse filtering and to the simplicity of integrating it and using it in very different machine learning tasks.

Some studies have also provided sparse filtering with some biological support. Bruce et al. (2016) analyzed different biologically-grounded principles for representation learning of images, using sparse filtering as a starting point for the definition of new learning algorithms. More interestingly, Kozlov and Gentner (2016) used sparse filtering to model the receptive fields of high-level auditory neurons in a songbird; relying on the idea that sparseness and normalization are canonical neural processing operations (Carandini and Heeger, 2012), their results show that sparse filtering can reproduce the receptive fields of the European starling and provides further confirmation of the general hypothesis that sparsity and normalization are general principles of neural computation in the sensory system of animals. Sparse filtering may then be of interest not only for practical machine learning applications, but also for modeling and understanding biological systems.

1.2 Problem Statement

So far, the sparse filtering algorithm has been successfully applied to many scenarios, and its usefulness is beyond doubt. However, a clear theoretical explanation of the algorithm is still lacking. Ngiam et al. (2011) drew connections between sparse filtering, divisive normalization, independent component analysis, and sparse coding, while Lederer and Guadarrama (2014) provided a deeper analysis of the normalization steps inside the sparse filtering algorithm. Despite this, the reasons why sparse filtering, in particular, and feature distribution learning, in general, work are left unexplored. In this research, we do not aim to show *whether* or *how well* sparse filtering works; its state-of-the-art performance has already been reported and reproduced. Instead, we aim at understanding *why* and *when* sparse filtering works well.

We begin by arguing that any unsupervised learning algorithm, in order to work properly, has to deal with the problem of preserving information conveyed by the probability distribution of the data. Now, given that feature distribution learning ignores the problem of learning the data distribution, a natural question arises: *how is the information conveyed by the data distribution preserved in feature distribution learning and in sparse filtering?*

We first tackle this question about information preservation in the generic case of feature distribution learning. We believe that addressing this question in the abstract is important, as it will give insights on how we could develop new feature distribution learning algorithms. The first step in order to discuss information preservation in feature distribution learning algorithms is to have a clear understanding of what feature distribution algorithms are. We then state the problem: *how can we distinguish between data distribution learning algorithms and feature distribution learning algorithms?*

Once we have this understanding, we can tackle the actual problem of information preservation. We then consider: *what does it mean to ignore learning the data distribution and to what degree can we actually ignore it?*

Next, we turn to the question about information preservation in the concrete case of sparse filtering. The practical success of sparse filtering suggests that the algorithm is indeed able to preserve relevant information conveyed in the distribution of the data. However, no explanation for this behavior has been given. To understand how sparse filtering may preserve information, we focus on the transformations of the data defined by the algorithm. We then investigate the properties of these transformations in order to answer the following question: *is there any sort of data structure that is preserved by the processing in sparse filtering?*

If we can prove that sparse filtering does indeed preserve information, the subsequent questions concern whether our formal analysis can help us explain the success or the failure of sparse filtering in practical applications. In particular we consider: *can we explain the* *failure of alternative forms of sparse filtering on the grounds of information preservation? Can we explain the success of sparse filtering in terms of the type of structure preserved? Can we identify scenarios in which sparse filtering is likely to be helpful and other scenarios in which it is likely not to be useful?*

1.3 Contributions

We summarize the contributions made in this study as follows:

- We present a theoretical framework that contributes to a better understanding of feature distribution learning, provides a way to distinguish it from data distribution learning, and explains the role of information preservation.

- We provide a deep theoretical and practical understanding of how sparse filtering works, by describing its dynamics and its properties, by proving that it preserves information through data structure preservation, and by showing which sort of data structure it preserves.

- We apply our understanding to concrete scenarios, thus offering a justification of the real-world success of sparse filtering, providing an explanation for the failure of alternative forms of sparse filtering, and suggesting a set of conditions under which we can expect sparse filtering to be useful.

Ultimately, we show that sparse filtering, through its implicit constraints, encodes a precise assumption about the data that necessarily makes it more suitable for certain scenarios.

1.4 Organization

The rest of this paper is organized as follows. We first review the concepts and ideas forming the foundations of our work (Section 2). Next, we propose a more rigorous conceptualization of the idea of feature distribution learning (Section 3). This conceptual study drives the following theoretical analysis of the sparse filtering algorithm (Section 4). The theoretical results we achieved inform the ensuing experimental simulations (Section 5). We then discuss the results we collected, in light of our analysis of sparse filtering, in particular, and feature distribution learning, in general (Section 6). Finally, we draw conclusions by summarizing our contributions and highlighting future developments (Section 7).

The mathematical notation is introduced progressively through the text, but in Table 1 we offer a quick reference that summarizes the conventions we adopted.

2. Foundations

In this section we review basic concepts underlying our study. We provide a rigorous description of unsupervised learning, we relate it to distribution learning, we formalize the property of sparsity, and, finally, we bring all these concepts together in the definition of the sparse filtering algorithm.

Main identifiers used throughout the text:	
$\mathbf{X}$	Matrix of original representations with domain $\mathbb{R}^{O \times N}$.
$\mathbf{Z}$	Matrix of learned representations with domain $\mathbb{R}^{L \times N}$.
$\mathbf{F}/\tilde{\mathbf{F}}/\hat{\mathbf{F}}$	Matrix of intermediate representations with domain $\mathbb{R}^{L \times N}$.
$\mathbf{Y}$	Vector of labels associated with the data with domain $\mathbb{R}^{1 \times N}$.
$\mathbf{W}$	Matrix of weights with domain domain $\mathbb{R}^{L \times O}$.
N	Number of samples.
O	Original dimensionality of the samples.
L	Learned dimensionality of the samples.
Notation for the data (with reference to the original data):	
$\mathcal{X}$	Data set or collection of data.
$\mathbf{x}^{(i)}$	i-th sample from $\mathbf{X}$; vector of shape $(O \times 1)$ with domain $\mathbb{R}^{O}$, $1 \leq i \leq N$.
$\mathbf{x}_j$	j-th feature from $\mathbf{X}$; vector of shape $(1 \times N)$ with domain $\mathbb{R}^{N}$, $1 \leq j \leq O$.
$\mathbf{X}_j^{(i)}$	j-th feature of the i-th sample from $\mathbf{X}$; scalar with domain $\mathbb{R}$.
X	Multivariate random variable (random vector) modeling the original data $\mathbf{x}^{(i)}$.
$p(X)$	Probability density function of the original representations.
$p\left(\mathbf{x}^{(i)}\right)$	Probability of the outcome $\mathbf{x}^{(i)}$ when sampling from $p(X)$. $p\left(\mathbf{x}^{(i)}\right)$ is the shorthand for the more rigorous notation $p\left(X = \mathbf{x}^{(i)}\right)$.

Table 1: Notation.

2.1 Unsupervised Learning

Let $\mathcal{X} = \{\mathbf{x}^{(i)} \in \mathbb{R}^O\}_{i=1}^N$ be a set of N *samples* or *data points* represented as vectors in an O-dimensional space. We will refer to the given representation of a sample $\mathbf{x}^{(i)}$ in the space $\mathbb{R}^O$ as the *original representation* of the sample $\mathbf{x}^{(i)}$ and to $\mathbb{R}^O$ as the *original space*. From an algebraic point of view, we can represent the data set as a matrix $\mathbf{X}$ of dimensions $(O \times N)$; from a probabilistic point of view, we can model the data points $\mathbf{x}^{(i)}$ as i.i.d. samples from a multivariate random variable $X = (X_1, X_2, \dots, X_O)$ with pdf $p(X)$.

Unsupervised learning discovers a transformation $f : \mathbb{R}^O \to \mathbb{R}^L$ mapping the set $\mathcal{X} = \{\mathbf{x}^{(i)} \in \mathbb{R}^O\}_{i=1}^N$ from an O-dimensional space to the set $\mathcal{Z} = \{\mathbf{z}^{(i)} \in \mathbb{R}^L\}_{i=1}^N$ in an L-dimensional space. We will refer to the transformed representation $\mathbf{z}^{(i)}$ in the space $\mathbb{R}^L$ as the *learned representation* of the sample $\mathbf{x}^{(i)}$ and to $\mathbb{R}^L$ as the *learned space*. Again, from an algebraic point of view, we can represent the transformed data set as a matrix $\mathbf{Z}$ of dimensions $(L \times N)$; from a probabilistic point of view, we can model the data points $\mathbf{z}^{(i)}$ as i.i.d. samples from a multivariate random variable $Z = (Z_1, Z_2, \dots, Z_L)$ with pdf $p(Z)$.

Unsupervised learning is often used for learning better representations for ensuing supervised tasks. Suppose that we are given a set $\mathcal{Y} = \{y^{(i)} \in \mathbb{R}\}_{i=1}^N$ of N *labels*, such that the i^{th} label in $\mathcal{Y}$ is associated to the i^{th} sample in $\mathcal{X}$. From an algebraic point of view, we can represent the labels as a vector $\mathbf{Y}$ of dimensions $(1 \times N)$; from a probabilistic point of view, we can model the labels $y^{(i)}$ as i.i.d. samples from a random variable Y with pdf $p(Y)$. Let us now consider the new data set $(\mathcal{X}, \mathcal{Y}) = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathbb{R}^O \times \mathbb{R}\right\}_{i=1}^N$. In this scenario, the aim of unsupervised learning is to learn representations $\mathbf{z}^{(i)}$ such that modeling the relationship $f : \mathbf{z}^{(i)} \mapsto y^{(i)}$ is easier than modeling the relationship $f : \mathbf{x}^{(i)} \mapsto y^{(i)}$.

Clustering. A specific form of unsupervised learning is *clustering*. In general, clustering algorithms can be categorized as hard or soft. Hard clustering discovers a transformation $f : \mathbb{R}^O \to \mathbb{R}^L$ mapping the original samples $\mathbf{x}^{(i)}$ onto one-hot representations $\mathbf{z}^{(i)}$, where the single non-null component of $\mathbf{z}^{(i)}$ encodes the assignment of the original sample to a cluster. Soft clustering discovers a transformation $f : \mathbb{R}^O \to \mathbb{R}^L$ mapping the original samples $\mathbf{x}^{(i)}$ onto representations $\mathbf{z}^{(i)}$, where the value of each component of $\mathbf{z}^{(i)}$ encodes the degree of membership of the original sample to each cluster.

Soft clustering algorithms may be used for learning representations $\mathbf{z}^{(i)}$ that simplify the problem of modeling the relationship $f : \mathbf{z}^{(i)} \mapsto y^{(i)}$. A standard soft clustering algorithm is grounded on the following assumptions. (i) Data points are generated by a true stochastic process with pdf $p(X^*)$ and then corrupted by some form of random noise; the final samples $\mathbf{x}^{(i)}$ with pdf $p(X)$ are therefore corrupted versions of the true samples and they are expected to have a weaker correlation to the labels $y^{(i)}$ than the true samples. (ii) The true pdf $p(X^*)$ may be approximated through a mixture model. (iii) Relationships of neighborhoodness under a chosen metric in the original space $\mathbb{R}^O$ allows us to recover the original pdf $p(X^*)$; data points $\mathbf{x}^{(i)}$ that happen to be next to each other in the original space under the chosen metric are taken to be similar. Under these assumptions, soft clustering algorithms instantiate a set of C clusters (each one describing one component of the mixture model) and group into the clusters nearby data points. Two data points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ falling in the same clusters are represented by the same exemplar $\bar{\mathbf{x}}$, assuming that such an exemplar contains all the relevant information carried by $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, and that the information contained in the difference between $\mathbf{x}^{(1)}$ or $\mathbf{x}^{(2)}$ and the exemplar $\bar{\mathbf{x}}$ amounts to noise. If all the assumptions are correct, a soft clustering algorithm will learn new representations $\mathbf{z}^{(i)}$ whose pdf $p(Z)$ is closer to the true pdf $p\left(X^{*}\right)$ than the original pdf $p(X)$; and, therefore, it will be easier to learn $f : \mathbf{z}^{(i)} \mapsto y^{(i)}$ or $p(Y|Z)$ than learning $f : \mathbf{x}^{(i)} \mapsto y^{(i)}$ or $p(Y|X)$.

2.2 Distribution Learning

Distribution learning is a form of unsupervised learning focused either on modeling the true pdf $p\left(X^{*}\right)$ or on shaping a useful pdf $p(Z)$.

Data Distribution Learning. Intuitively, we define *data distribution learning* as any unsupervised learning algorithm that is concerned with learning a pdf $p\left(Z\right)$ that closely matches the true pdf $p\left(X^{*}\right)$. Data distribution learning algorithms aim at estimating $p\left(X^{*}\right)$ from the available data, and the learned representation $\mathbf{z}^{(i)}$ is meant to encode the factors that explain the original data $\mathbf{x}^{(i)}$. Examples of data distribution learning algorithms include denoising auto-encoders (DAE), restricted Boltzmann machines (RBM), and independent component analysis (ICA) (Ngiam et al., 2011).

In the context of representation learning for supervised tasks, learning a pdf $p(Z)$ that approximates the true pdf $p\left(X^{*}\right)$ is meaningful under the assumption that the labels we are given are strongly correlated with the true distribution of the data. If $p(Z)$ is a good estimation of the true pdf $p\left(X^{*}\right)$, we can reasonably expect that learning $p(Y|Z)$ or $p(Z,Y)$ will be simplified.

Feature Distribution Learning. In contrast, we intuitively define *feature distribution learning* as any unsupervised learning algorithm that is concerned with learning a pdf $p\left(Z\right)$ which has a set of desirable properties. Feature distribution learning algorithms overlook the problem of estimating the true distribution $p\left(X^{*}\right)$ and focus instead on shaping the learned pdf $p\left(Z\right)$ according to chosen criteria. The most representative algorithm of this family is sparse filtering (SF) (Ngiam et al., 2011).

In the context of representation learning for supervised tasks, learning a pdf $p(Z)$ with specific properties is meaningful if we know a priori that certain properties will be useful for supervised learning. Properties like sparsity or smoothness are well known to be useful properties when modeling $p(Y|Z)$ or $p(Z,Y)$ and, therefore, they can be imposed during feature distribution learning.

2.3 Sparsity

Definition (Sparsity). Given a generic vector $\mathbf{v}$ in an N-dimensional space, we say that $\mathbf{v}$ is sparse if a small number of components of the vector accounts for most of the energy[1] of the vector (Hurley and Rickard, 2009).

Practically, we say that the vector $\mathbf{v}$ is *sparse* if $n \ll N$ components of the vector $\mathbf{v}$ are active (that is, have a value different from zero) while the remaining $N-n$ components are inactive (that is, have the value zero). A vector $\mathbf{v}$ is *k-sparse* if exactly k components are active. By analogy, we may define sparsity for matrices (with reference to their components)

1. We refer here to the meaning of energy from signal processing, that is: $energy(\mathbf{v}) = \sum_{i=1}^{N} |\mathbf{v}_i|^2$.

and for random variables (with reference to their realizations).

Measures of sparsity. Several measures of sparsity have been proposed in the literature; Hurley and Rickard (2009) offer a review of different measures of sparsity and their properties. Here, we pay attention only to a specific measure from the family of ℓ_p-norm measures.

One of the most common measures of sparsity is the ℓ_1-norm. Given a generic vector $\mathbf{v}$ in an N-dimensional space, its ℓ_1-norm is defined as: $\ell_1(\mathbf{v}) = \sum_{i=1}^{N} |\mathbf{v}_i|$. The ℓ_1-norm is an approximation of the ℓ_0-norm that has a continuous derivative (except in the origin), and thus is easily optimized with standard derivative-based techniques. In the sparse filtering literature, the ℓ_1-norm is often referred to as *activation*. Given a representation $\mathbf{z}^{(i)}$, we will quantify its sparsity by computing $\ell_1\left(\mathbf{z}^{(i)}\right)$ or $activation\left(\mathbf{z}^{(i)}\right)$. Minimizing the activation of the learned representation $\mathbf{z}^{(i)}$ will maximize the sparsity of $\mathbf{z}^{(i)}$.

2.4 Sparse Filtering

Sparse filtering is the most representative example of feature distribution learning algorithms (Ngiam et al., 2011). Its aim is learning a pdf $p(Z)$ which maximizes the sparsity of the learned representations $\mathbf{z}^{(i)}$.

Enforcement of sparsity in sparse filtering. A sparse learned representation $\mathbf{z}^{(i)}$ is achieved by enforcing three constraints on the matrix of learned representations $\mathbf{Z}$:

- *Population sparsity*: each sample $\mathbf{z}^{(i)}$, $1 \leq i \leq N$, is required to be sparse, that is, described only by a few features. The sparsity of a sample $\mathbf{z}^{(i)}$ is computed as its activation: $\ell_1\left(\mathbf{z}^{(i)}\right) = \sum_{j=1}^{L} \left|\mathbf{z}_j^{(i)}\right|$.

- *Lifetime sparsity*: each feature $\mathbf{z}_j$, $1 \leq j \leq L$, is required to be sparse, that is, to describe only a few samples. Lifetime sparsity is often referred to as *selectivity* (Goh et al., 2012). The sparsity of a feature $\mathbf{z}_j$ is computed as its activation: $\ell_1(\mathbf{z}_j) = \sum_{i=1}^{N} \left|\mathbf{z}_j^{(i)}\right|$.

- *High dispersal*: all the features $\mathbf{z}_j$, $1 \leq j \leq L$, are required to have approximately the same activation. The dispersal of the features $\mathbf{z}_j$ is computed as the variance of their activation: $Var\left[activation(\mathbf{z}_j)\right]$. Lower variance corresponds to higher dispersal.

The enforcement of these three properties translates into learning non-degenerate sparse representation.

Implementation of sparse filtering. Sparse filtering is implemented as a simple algorithm in six steps (refer to Figure 1 for an illustration of this decomposition and to Figure 2 for an illustration of the transformations on a two-dimensional data set):

A0. Initialization of the weights: the weight matrix $\mathbf{W}$ with dimensions $(L \times O)$ is initialized by sampling each component from a normal distribution $\mathcal{N}(0, 1)$.

A1. Linear projection of the original data: $f_{A1}(\mathbf{X}) = \mathbf{WX}$. The weight matrix $\mathbf{W}$ can be interpreted as a *dictionary* (Denil and de Freitas, 2012) or as a *filter bank* (Dong et al., 2015), where each row is a codeword or a filter applied to every sample in the columns of $\mathbf{X}$. Refer to Figure 2(a) and 2(b) for an illustration of this transformation.

A2. Non-linear transformation: $\mathbf{F} = f_{A2}\left(\mathbf{WX}\right)$, where $f_{A2}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is an element-wise non-linear function. Although this non-linear function can, in principle, be arbitrarily chosen, all the implementations known to the authors used an element-wise absolute-value function $f(x) = |x|$. For practical reasons, this non-linearity is implemented as a soft absolute-value $f(x) = \sqrt{x^2 + \epsilon}$, where ϵ is a small negligible value (for instance, $\epsilon = 10^{-8}$). Refer to Figure 2(b) and 2(c) for an illustration of this transformation.

A3. ℓ_2-normalization along the features (or along the rows): $\tilde{\mathbf{F}} = f_{A3}\left(\mathbf{F}\right) = \frac{\mathbf{F}_j^{(i)}}{\sqrt{\sum_{i=1}^{N}\left(\mathbf{F}_j^{(i)}\right)^2}}$.

In this step, each feature is normalized so that its squared activation is one, that is, $\sum_{i=1}^{N}\left(\tilde{\mathbf{F}}_j^{(i)}\right)^2 = 1$. Refer to Figure 2(c) and 2(d) for an illustration of this transformation.

A4. ℓ_2-normalization along the samples (or along the columns): $\mathbf{Z} = \hat{\mathbf{F}} = f_{A4}\left(\tilde{\mathbf{F}}\right) = \frac{\tilde{\mathbf{F}}_j^{(i)}}{\sqrt{\sum_{j=1}^{L}\left(\tilde{\mathbf{F}}_j^{(i)}\right)^2}}$. In this step, each sample is normalized so that its squared activation is one, that is, $\sum_{j=1}^{L}\left(\hat{\mathbf{F}}_j^{(i)}\right)^2 = 1$. Refer to Figure 2(d) and 2(e) for an illustration of this transformation.

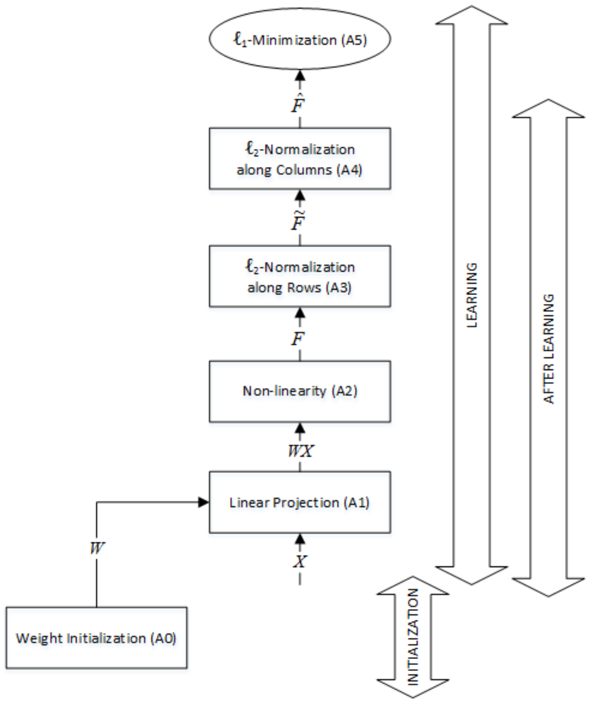
A5. ℓ_1-minimization: $\min \sum_{ij} \hat{\mathbf{F}}_j^{(i)}$. This minimization is the objective of sparse filtering; by minimizing the overall activation of the matrix $\hat{\mathbf{F}}$, we maximize the sparsity of the learned representations.

Notice that step A0 is executed only during the initialization of the algorithm, while step A5 is executed only during learning. After learning, new data $\mathbf{X}'$ is processed through step A1 to A4, that is, $\mathbf{Z}' = f_{A1:A4}\left(\mathbf{X}'\right) = f_{A4}\left(f_{A3}\left(f_{A2}\left(\mathbf{WX}'\right)\right)\right)$

As explained by Ngiam et al. (2011), the combination of the ℓ_1-minimization with the two ℓ_2-normalizations guarantees the learning of a representation with the properties of population sparsity, lifetime sparsity, and high dispersal.

3. Conceptual Analysis of Feature Distribution Learning

We believe that the intuitive definition of feature distribution learning reviewed in Section 2.2 is unsatisfactory. On one side, it can not really be used to categorize algorithms that both try to learn the true distribution of the data $p\left(X^*\right)$ and to model a useful distribution of the features $p(Z)$ (for example, sparse RBM, Ranzato et al., 2007). On the other hand, it is not clear what is implied by the fact that feature distribution learning ignores learning the true data distribution $p\left(X^*\right)$; "ignoring" may mean anything ranging from the extreme option of "completely disregarding the problem of learning the data distribution" to the more moderate option of "not caring about optimizing the learning of the data distribution."



ℓ_1-Minimization (A5)
$\hat{F}$
ℓ_2-Normalization along Columns (A4)
$\tilde{F}$
ℓ_2-Normalization along Rows (A3)
F
Non-linearity (A2)
WX
Linear Projection (A1)
X
W
Weight Initialization (A0)
LEARNING
AFTER LEARNING
INITIALIZATION

Figure 1: Decomposition of the sparse filtering algorithm in its constituent steps.

A better understanding of feature distribution learning is needed to properly study and analyze concrete instances of feature distribution learning, such as sparse filtering. Here we suggest a more comprehensive definition of distribution learning which is grounded on information-theoretic concepts and which relies on ideas already discussed by the machine learning community.

3.1 Infomax Principle and Informativeness Principle

Vincent et al. (2010) argued that an unsupervised learning algorithm can generate good representations by satisfying two requirements: (i) retaining information about the input, and (ii) applying constraints that lead to the extraction of useful information from noise.

In more general terms, we may state that a good unsupervised representation may be obtained by satisfying the two following information-theoretic requirements: (i) maximizing the mutual information between input and output, and (ii) maximizing a measure of information of the output (against noise). The first requirement is the same as the one stated by Vincent et al. (2010), and it corresponds to the *infomax principle* (Linsker, 1989). The second requirement, for a lack of a better term, will be referred to as *informativeness principle*.

According to this understanding, the aim of feature distribution learning may be expressed as a pure optimization of the informativeness principle. That is, we try to learn a transformation f such that the pdf $p(Z)$ of the new representations $\mathbf{z}^{(i)}$ has maximal informativeness. Maximizing the informativeness may be simply expressed as the minimization

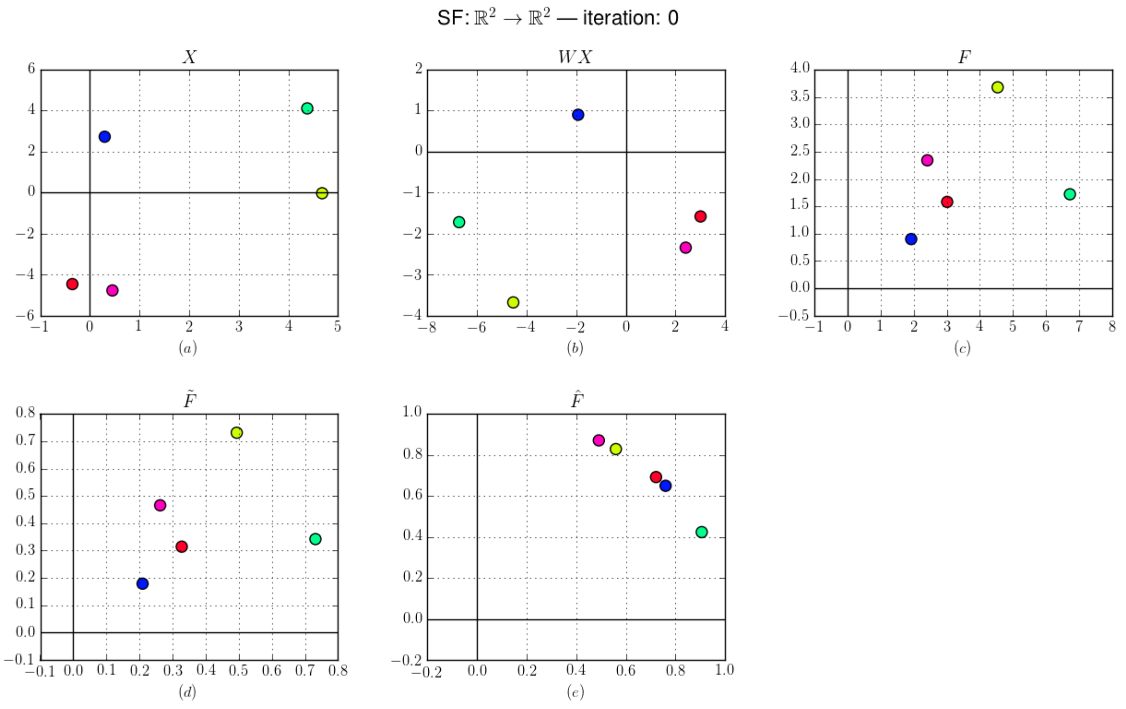


Figure 2: Illustration of sparse filtering.
Sparse filtering is applied to a random set of data $\mathbf{X}$ of five samples ($N = 5$) in two dimensions ($O = 2$). Each point is generated by sampling its coordinates from a uniform distribution $Unif\,(-5, 5)$. Sparse filtering is used to learn a new representation of the data in two dimensions ($L = 2$). This figure shows the transformations determined by the sparse filtering algorithm at iteration 0, after the weight matrix $\mathbf{W}$ has been randomly initialized and before any training.
(a) Original representation of the data $\mathbf{X}$ in $\mathbb{R}^2$. (b) Linear projection of the data onto the intermediate representation $\mathbf{WX}$. (c) Non-linear projection of $\mathbf{WX}$ using an absolute-value function onto the intermediate representation $\mathbf{F}$. (d) ℓ_2-normalization of the data $\mathbf{F}$ along the features, yielding the intermediate representation $\tilde{\mathbf{F}}$. (e) ℓ_2-normalization of the data $\tilde{\mathbf{F}}$ along the samples, yielding the final learned representation $\hat{\mathbf{F}} = \mathbf{Z}$.
Notice that the colors of the data points $\mathbf{x}^{(i)}$ do not have any meanings. A random color has been assigned to each point in order to allow the tracking of the location of the points through the different transformations applied by sparse filtering.

of the entropy $H[Z]$ or the maximization of the relative entropy between the learned pdf $p(Z)$ and the entropy-maximizing pdf q:

$$\max D\left[p(Z) \parallel q\right],$$

where $D[\cdot]$ is a measure of distance or divergence between the pdfs, such as the Kullback-Leibler divergence (MacKay, 2003).

However, without other requirements, the objective of maximizing the information is not sufficient to lead to any useful or meaningful learning. The optimal solution of the problem of maximizing $D[p(Z) \parallel q]$ is trivially learning a pdf with the shape of a Dirac delta function. If we map all the original samples $\mathbf{x}^{(i)}$ onto an arbitrary representation $\bar{\mathbf{z}}$, then the pdf $p(Z)$ will have the shape of a Dirac delta function centered on $\bar{\mathbf{z}}$. This pdf has minimal entropy and, therefore, maximal informativeness. However, it is clear that arbitrarily mapping all the samples $\mathbf{x}^{(i)}$ to a constant representation $\bar{\mathbf{z}}$ has no meaning. We maximize the information in $p(Z)$ but we lose all the information carried by $p(X)$ about $p(X^*)$.

It is necessary, therefore, to take into consideration the infomax principle as well. This translates into the maximization of the mutual information $I[X;Z]$, or, equivalently, into the maximization of the relative entropy between $p(X,Z)$ and $p(X)p(Z)$:

$$\max D\left[p(X,Z) \parallel p(X)p(Z)\right],$$

where, again, $D[\cdot]$ is a measure of distance or divergence between the pdfs.

Like any unsupervised learning algorithm, feature distribution learning has to somehow negotiate the trade-off between the infomax principle and the informativeness principle:

$$\underbrace{\max D\left[p(X,Z) \parallel p(X)p(Z)\right]}_{\substack{infomax\\ principle}} + \underbrace{\max D\left[p(Z) \parallel q\right]}_{\substack{informativeness\\ principle}}. \tag{1}$$

Even if the definition of feature distribution learning makes no reference to the infomax principle, we argue that it must taken into account in some way.

The learning objective defined in Equation 1 remains mainly theoretical. Information-theoretic quantities, like relative entropy, are extremely hard to estimate and therefore we have to rely on approximations or heuristics. Moreover, the optimization of a dual-objective function is often very challenging; operational research suggests that a simpler approach would be to translate one objective into a constraint and explicitly optimize the remaining one. Indeed, this approach seems to provide a better way to understand the difference between data distribution learning and feature distribution learning.

Data distribution learning. In relation to Equation 1, we can define data distribution learning as any unsupervised learning in which the main objective is maximizing the infomax principle (first term in Equation 1), while the maximization of the informativeness principle (second term in Equation 1) is accounted for through priors or constraints.

For instance, DAEs approximate the maximization of the mutual information $I[X;Z]$ through the minimization of the reconstruction error; Vincent et al. (2010) showed that minimizing the reconstruction error in a DAE is indeed equivalent to maximizing a lower bound on the mutual information. Similarly, RBMs approximate the maximization of the mutual information $I[X;Z]$ through the minimization of the divergence between the distribution of the data and the learned distribution via the maximization of the log probability of the data (Hinton et al., 2006). These algorithms do not tackle the problem of maximizing the relative entropy $D[p(Z) \parallel q]$ directly, but they often address it by using constraints, such as bottleneck architectures (Tishby et al., 2000), or by imposing priors, such as adding a sparsity penalty to the learning objective (Vincent et al., 2010).

Feature distribution learning. Again, in relation to Equation 1, we can define feature distribution learning as any unsupervised learning in which the main objective is maximizing the informativeness principle (second term in Equation 1), while the maximization of the infomax principle (first term in Equation 1) is accounted through priors or constraints.

4. Theoretical Analysis of Sparse Filtering

The previous conceptual analysis of feature distribution learning provides the bedrock for studying and understanding sparse filtering. Relying on the fact that any unsupervised algorithm (including feature distribution learning algorithms) must somehow satisfy the infomax principle and the informativeness principle, we will deploy a set of conceptual tools, definitions, proofs, and lemmas to demonstrate the following thesis:

> *Sparse filtering does satisfy the informativeness principle through the maximization of the proxy of sparsity and it satisfies the infomax principle through the constraint of preservation of the structure of cosine neighborhoodness of the data.*

4.1 Informativeness Principle

Showing that sparse filtering satisfies the informativeness principle is straightforward. Since the explicit maximization of the relative entropy $D[p(Z) \parallel q]$ is computationally hard, the sparse filtering algorithm adopts the standard proxy of sparsity. Increasing the sparsity of the representations $\mathbf{z}^{(i)}$ increases the entropy of the learned distribution $p(Z)$ by concentrating the mass of the pdf $p(Z)$ around zero. Thus, minimizing the ℓ_1-norm of the learned representations $\mathbf{z}^{(i)}$ in the objective function accounts for the maximization of the informativeness of the learned representation $\mathbf{z}^{(i)}$.

4.2 Infomax Principle

Showing that sparse filtering satisfies the informativeness principle is more complex. By definition, as a feature distribution learning algorithm, sparse filtering does not address the problem of modeling the data distribution. However, by virtue of the fact that sparse filtering works and its learned representations $\mathbf{z}^{(i)}$ allow the achievement of state-of-the-art performance when learning $p(Y|Z)$, it *must* be that the algorithm preserves information contained in the original representations $\mathbf{x}^{(i)}$. *If it were not so*, sparse filtering could simply solve its optimization problem by mapping the original data matrix $\mathbf{X}$ onto a pre-computed sparse representation matrix $\bar{\mathbf{Z}}$ with a minimal computational complexity of $O(1)$; however, these representations $\mathbf{z}^{(i)}$ would clearly be useless because of the independence between representations and labels, from which follows that $p(Y|Z) = p(Y)$.

Since sparse filtering does not try to explicitly model $p\left(X^{*}\right)$, we hypothesize that it must implicitly preserve information about $p\left(X^{*}\right)$ conveyed by the pdf $p(X)$. In particular, we hypothesize that sparse filtering preserves the information conveyed by the pdf $p(X)$ through the proxy of the preservation of data structure. The geometric structure of the data in the original space $\mathbb{R}^{O}$ constitutes a set of realizations of the random variable X through which we can estimate the pdf $p(X)$. Preserving relationships of neighborhoodness (under a given metric) allows us to preserve information conveyed by the pdf $p(X)$: regions of high density and low density in the domain of $p(X)$ can be maintained by preserving relationships of neighborhoodness. Thus, preservation of the geometric structure under a chosen metric may act as a proxy for the maximization of mutual information $I[X; Z]$.

4.3 Non-preservation of Euclidean Distances

The preservation of absolute or relative distances under the Euclidean metric is the most common way to preserve the structure of the data. However, it can be easily ruled out that sparse filtering preserves this type of structure.

Proposition. Let $\left\{\mathbf{x}^{(i)} \in \mathbb{R}^{O}\right\}_{i=1}^{N}$ be a set of points in the original space $\mathbb{R}^{O}$. Then, the transformations from A1 to A4 do not preserve the structure of the data described by the Euclidean metric.

Proof. Considering transformation A1, a generic linear transformation does not provide specific guarantees on the preservation of Euclidean distances from the original space $\mathbb{R}^{O}$ to the learned space $\mathbb{R}^{L}$. Specific instances of linear transformation may preserve Euclidean absolute distances (rotations and reflections) or relative distances (scaling). The Jonhson-Lindestrauss lemma (Dasgupta and Gupta, 2003; Johnson and Lindenstrauss, 1984) sets bounds on the preservation of the distances by random projections; however, we have no clear guarantees that the randomly initialized weight matrix $\mathbf{W}$ would implement such a transformation or that, during training, sparse filtering would learn a transformation that preserves Euclidean distances.

Concerning transformation A2, the absolute-value function does not preserve, in general, absolute or relative Euclidean distances. Absolute distances are preserved only in the particular case in which the input points belong to the same orthant in the domain $\mathbb{R}^{L}$. This effect is particularly obvious when we interpret the application of absolute-value as a folding of the space $\mathbb{R}^{L}$ onto the positive orthant (Montufar et al., 2014).

Concerning transformation A3, the normalization along the features is a transformation that preserves relative Euclidean distances from the domain to the codomain space. This can be intuitively understood by observing that the normalization along the features merely corresponds to a rescaling of the axes in the feature space. A formal proof may be given showing that the transformation A3 can be encoded as a linear transformation with an associated positive diagonal matrix. By definition, a positive diagonal matrix preserves relative distances among the points.

Concerning transformation A4, the normalization along the samples is a transformation with no guarantees of preserving absolute or relative distances from the domain to the codomain space. As all the samples are projected from the positive orthant $\mathbb{R}^L_{\geq 0}$ to the surface of the unit hyper-sphere, relationships of relative distance are not preserved.

Since more than one of the transformations from A1 to A4 do not preserve Euclidean distances, we can conclude that the overall transformation $f_{A1:A4}(\cdot)$ does not preserve Euclidean distances. ■

4.4 Preservation of Collinearity

Having ascertained that sparse filtering can not preserve the data structure defined by the Euclidean metric, we investigate other properties of the algorithm that may lead us to discover the preservation of alternative data structures. A first relevant observation is that sparse filtering preserves collinearity.

Proposition. Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathbb{R}^O$ be collinear points in the original space $\mathbb{R}^O$. Then, the outputs of transformations from A1 to A4, that is $f_{A1:A4}\left(\mathbf{x}^{(1)}\right), f_{A1:A4}\left(\mathbf{x}^{(2)}\right) \in \mathbb{R}^L$, are collinear.

Proof. Concerning transformation A1, linear transformations preserve collinearity as proved in appendix A.1.

Concerning transformation A2, the absolute-value function preserves collinearity by rigidly folding all the orthants on the first one. A formal proof is given in appendix A.2.

Concerning transformation A3, the normalization along the features preserves collinearity as well, by acting simply as a rescaling of the axes. A formal proof is given in appendix A.3.

Concerning transformation A4, the normalization along the samples also preserves collinearity. A formal proof is given in appendix A.4.

Since all the transformations from A1 to A4 preserve collinearity, we can conclude that the overall transformation $f_{A1:A4}(\cdot)$ preserves collinearity. ■

4.5 Homo-representation of Collinear Points

An immediate consequence of the previous proposition is the following theorem which states that all the collinear points in the original representation space are mapped to an identical representation. This result is significant as it gives us a first understanding of the principle and the type of metric that sparse filtering uses to map original samples $\mathbf{x}^{(i)}$ onto their representations $\mathbf{z}^{(i)}$.

Theorem. Let $\mathbf{x} \in \mathbb{R}^O$ be a point in the the original space $\mathbb{R}^O$. Then there is a set of infinite points $\mathbf{x}^{(i)} \in \mathbb{R}^O$ such that $f_{A1:A4}(\mathbf{x}) = f_{A1:A4}\left(\mathbf{x}^{(i)}\right)$. This set is the set of the points collinear with $\mathbf{x}$.

Proof. This theorem follows from the proposition of preservation of collinearity. ■

4.6 Homo-representation of Points with Same Moduli

A further analysis of sparse filtering reveals that not only collinear points are mapped to the same representation, but also points in the learned representation space having the same moduli (that is, the same absolute value for their components) are mapped to identical representations. Again, this result is relevant since it sheds light on the type of structure preserved by sparse filtering.

Theorem. Let $\mathbf{z} \in \mathbb{R}^L$ be a point in the codomain of the function $\mathbf{WX}$. It holds that for a $\mathbf{z}$ in the first orthant, there are at least 2^L points $\mathbf{z}^{(i)} \in \mathbb{R}^L$ such that $f_{A2:A4}(\mathbf{z}) = f_{A2:A4}\left(\mathbf{z}^{(i)}\right)$.

Proof. By definition, $\mathbf{z} = \left[\begin{array}{cccc} z^{(1)} & z^{(2)} & \ldots & z^{(L)} \end{array}\right]$ is an L-dimensional vector whose components belong to the first orthant and are therefore positive.
Now, the application of the absolute-value $f_{A2}(\cdot)$ maps $\mathbf{z}$ to itself:

$$f_{A2}(\mathbf{z}) \;=\; \left[\begin{array}{cccc} z^{(1)} & z^{(2)} & \ldots & z^{(L)} \end{array}\right] = \mathbf{z}.$$

However, all the vectors $\mathbf{z}^{(i)}$ in the codomain of the function $\mathbf{WX}$ with the following form are mapped to $\mathbf{z}$ by the absolute-value $f_{A2}(\cdot)$:

$$\mathbf{z}^{(i)} = \left[\begin{array}{cccc} \pm z^{(1)} & \pm z^{(2)} & \ldots & \pm z^{(L)} \end{array}\right].$$

By combinatorial analysis, we know that there are 2^L possible ways of picking the values of $\mathbf{z}^{(i)}$, thus defining 2^L points in $\mathbb{R}^L$ that are mapped to the same value $\mathbf{z}$ by the the absolute-value $f_{A2}(\cdot)$. Since all the points $\mathbf{z}^{(i)}$ are mapped to the same point $\mathbf{z}$ at the end of step A2, the application of the remaining deterministic functions in the following steps of sparse filtering will map them to the same representation, so that $f_{A2:A4}(\mathbf{z}) = f_{A2:A4}\left(\mathbf{z}^{(i)}\right)$. ■

4.7 Poles and Pole Pursuit

To facilitate our formal description, we now introduce two concepts that will be useful in our analysis.

Definition (Poles). In an L-dimensional space $\mathbb{R}^L$, a *pole* is a point identified by a vector $\mathbf{p}$ such that $\exists!\, i,\ 1 \leq i \leq L$ for which $\mathbf{p}_i = 1$, and $\forall j,\ j \neq i,\ 1 \leq j \leq L$ then $\mathbf{p}_j = 0$.

Poles are then binary 1-sparse vectors having a single component set to one. The following properties derive easily from the definition: (i) in an L-dimensional space there are exactly L poles; and, (ii) mapping a set of original representations $\mathbf{x}^{(i)} \in \mathbb{R}^O$ to the poles of $\mathbb{R}^L$ produces an optimal solution for the sparse filtering algorithm, as the poles have a minimal ℓ_1-norm in $\mathbb{R}^L$ under the constraint of sparse filtering.

Definition (Pole Pursuit). Given a set of original representations $\mathbf{x}^{(i)} \in \mathbb{R}^O$, we define the search for an optimal solution as *pole pursuit*. Through gradient descent, the representations $\mathbf{z}^{(i)} \in \mathbb{R}^L$ are progressively pushed towards the poles of $\mathbb{R}^L$.

However, in general, notice that sparse filtering is not guaranteed to find a solution in which all the original representations $\mathbf{x}^{(i)}$ are mapped onto poles. The achievement of such an optimal solution depends on the original data set $\mathcal{X}$, on the dimensionality of the learned space L, and on the random initialization of the weight matrix $\mathbf{W}$. Gradient descent may lead sparse filtering to settle into a sub-optimal solution where the original representations $\mathbf{x}^{(i)}$ are not mapped onto poles but onto k-sparse ($k > 1$) representations in $\mathbb{R}^L$.

4.8 Representation Cones

The propositions we proved above and the concept of pole pursuit allow us to introduce a last conceptual tool that gives us a better insight into the properties and the dynamics of sparse filtering.

From the theorem of homo-representation of collinear points we learned that sparse filtering identifies sets of collinear points in the original space that are mapped onto poles; from the theorem of homo-representation of points with the same moduli we can deduce that there must a symmetric structure around lines of collinear points. Putting together these two intuitions, we can conclude that sparse filtering defines precise maps in the original representation space $\mathbb{R}^O$. More precisely, we state that sparse filtering defines *representation cones*[2] in the original representation space $\mathbb{R}^O$.

Definition (Representation Cone). A *representation cone* $R^{\mathbf{p}^{(j)}}$ is a function $R^{\mathbf{p}^{(j)}}: \mathbb{R}^O \rightarrow \mathbb{R}_{\geq 0}$ mapping points in the original representation space $\mathbb{R}^O$ to their (Euclidean) distance from the pole $\mathbf{p}^{(j)}$.

Plotting a representation cone $R^{\mathbf{p}^{(j)}}$ in the original representation space $\mathbb{R}^O$ defines a region of space having a conical (or hyper-conical in higher dimensions) shape, such that all the points on the line of its height are mapped onto the pole $R^{\mathbf{p}^{(j)}}$, and all the points in its volume are mapped into the neighborhood of the pole $R^{\mathbf{p}^{(j)}}$. The volume of a representation cone is loosely defined and it changes according to the chosen size for the pole neighborhood.

Moreover, given a point $\mathbf{x}^{(i)} \in \mathbb{R}^O$, we say that the representation cone $R^{\mathbf{p}^{(j)}}_{\mathbf{x}^{(i)}}$ is centered on $\mathbf{x}^{(i)}$ if the point $\mathbf{x}^{(i)}$ lies on the line of the height of the representation cone $R^{\mathbf{p}^{(j)}}_{\mathbf{x}^{(i)}}$. Therefore, the representation cone $R^{\mathbf{p}^{(j)}}_{\mathbf{x}^{(i)}}$ maps the point $\mathbf{x}^{(i)}$ and all the points collinear with $\mathbf{x}^{(i)}$ to the pole $\mathbf{p}^{(j)}$.

Properties. Several interesting and useful properties immediately follow from the definition of representation cones:

Association with a pole Each representation cone is associated with a single pole in the learned representation space $\mathbb{R}^L$.

Existence of L representation cones Given the learned representation space $\mathbb{R}^L$, the sparse filtering algorithm defines exactly L representation cones. This statement easily follows from the fact that in an L-dimensional space $\mathbb{R}^L$ we have exactly L poles.

O-dimensionality of the representation cones Given the original representation space $\mathbb{R}^O$, the sparse filtering algorithm defines O-dimensional representation cones in the

2. *Cones* suggest the idea of two-dimensional shapes; this is not strictly true as *representation cones* may live in higher dimensions; the nomenclature *cones* remains, though, as we first developed this idea while working in two-dimensions.

original representation space. Thus, if the original space is two-dimensional, the representation cone is an actual cone; in higher dimensions the representation cone is a hyper-cone.

Bounds of representation cones Given a point $\mathbf{x}^{(i)}$ in the original representation space $\mathbb{R}^O$, then $0 \leq R^{\mathbf{p}^{(j)}}\left(\mathbf{x}^{(i)}\right) \leq \sqrt{2}\ \ \forall j,\ 0 \leq j \leq L$. Since each point $\mathbf{x}^{(i)}$ is mapped to a point $\mathbf{z}^{(i)}$ on the surface of the unit hyper-sphere in the positive orthant, the distance from any pole of $\mathbf{z}^{(i)}$ is bounded between 0 and $\sqrt{2}$.

Closeness to the poles The representation cones comply with a simple rule of inverse proportionality: the closer a point $\mathbf{x}^{(i)}$ approaches a pole $\mathbf{p}^{(j)}$, the further it moves away from all other poles $\mathbf{p}^{(k)}$.

Complementarity of the representation poles Inspecting a plot of the representation cones gives us a rapid intuitive idea of the quality of the solution: points on the line of height of a representation cone $R^{\mathbf{p}^{(j)}}$ are mapped onto a perfect 1-sparse representation (pole); points within the volume of a representation cone $R^{\mathbf{p}^{(j)}}$ are mapped in the neighborhood of a pole; points far from any representation cone are mapped onto sub-optimal k-sparse representations, with $1 < k \leq L$.

Learning. After initialization, the representation cones are randomly placed in the original representation space $\mathbb{R}^O$. This leads to an unsatisfactory solution, as random points far from our samples $\mathbf{x}^{(i)}$ may be mapped to poles, while the samples $\mathbf{x}^{(i)}$ may be left far from the poles. Pole pursuit consists in moving around the representation cones so that at the end they are centered on the samples $\mathbf{x}^{(i)}$.

The optimization process of sparse filtering can be interpreted as the *search for an optimal location of the representation cones*: representation cones are rotated in a continuous way in the original representation space, until their placement provides an optimal solution in terms of sparsity of the learned representations. Indeed, the optimal solution to the problem of minimizing the ℓ_1-norm of the learned representations $\mathbf{z}^{(i)}$ is equivalent to the optimal solution to the problem of minimizing the $R^{\mathbf{p}^{(j)}}$ distances of the original representations $\mathbf{x}^{(i)}$. In other words, instead of minimizing the sparsity of the learned representations we can minimize the distances computed by the representation cones.

Lemma (Learning). The optimal solution for the minimization of $\sum_{i=1}^{N} \ell\left(\mathbf{z}^{(i)}\right)$, under the constraint $\ell_2\left(\mathbf{z}^{(i)}\right) = 1$, is the same as the optimal solution for $\sum_{i=1}^{N}\sum_{j=1}^{L} R^{\mathbf{p}^{(j)}}\left(\mathbf{x}^{(i)}\right)$, under the same constraint.

Proof. A proof of this lemma is given in appendix A.5. ■

4.9 Preservation of Cosine Neighborhoodness

Through representation cones we have shown that sparse filtering maps points having the same angles to the same representation and points having similar angles to similar representation. The structure that sparse filtering tries to preserve is therefore the structure defined by the cosine metric in the original space $\mathbb{R}^O$:

$$D_C\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = 1 - \cos\theta,$$

where $D_C\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$ is the cosine distance between the vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, and θ is the angle between the vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. Sparse filtering preserves cosine distances only in an approximate way: collinear or close-by points will be mapped next to each other, but points far apart are not necessarily kept separated. This is a consequence of the fact that the linear transformation in step A1 preserves collinearity but does not preserve the cosine metric, and that the non-linearity in step A2 may collapse together points radially far apart. In general, points with small cosine distances $D_C\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$ in the original space $\mathbb{R}^O$ have small Euclidean distances $D_E\left(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}\right)$ in the learned space $\mathbb{R}^L$.

Recall that, given a sample $\mathbf{x}^{(i)}$ in the original space $\mathbb{R}^O$, this sample can be described by O Cartesian coordinates, or by one radial coordinate ρ and $O-1$ angular coordinates θ_i. Thus, sparse filtering tries to preserve the information about the $O-1$ angular coordinates θ_i, discarding the information about the radial coordinate ρ.

4.10 Non-preservation of Cosine Neighborhoodness in Alternative Implementations of Sparse Filtering

The choice of the non-linearity applied in step A2 is crucial for guaranteeing the preservation of cosine neighborhoodness. Indeed, the absolute-value non-linearity is a suitable non-linear function for sparse filtering precisely because of its property of preservation of collinearity.

Ngiam et al. (2011) suggest that the original absolute-value non-linearity may be substituted by other non-linear functions; for instance, standard non-linear functions from the neural networks literature, such as the sigmoid non-linearity or the rectified linear unit (ReLU), may be used. Despite this possibility, all the implementations of sparse filtering so far have relied on the absolute-value non-linearity. An unpublished technical report by Thaler[3], after the Kaggle Black Box Learning Challenge, states that sparse filtering with alternative non-linearities (ReLU and quadratic non-linearity) did not perform as well as the absolute-value non-linearity, but does not clarify the reasons of this lack of success. Therefore, for experimental reasons, the absolute-value has always been recommended as the best non-linearity for sparse filtering.

Here, we argue that one theoretical reason for the limited success of alternative implementations of sparse filtering is due to the fact that they can not provide strong guarantees of preservation of data structure, as the standard implementation of sparse filtering does. If we replace the absolute-value non-linearity with another non-linearity, such as sigmoid or ReLU, we lose the property of preservation of collinearity. Indeed, non-linearities such as sigmoid or ReLU do not induce in the original space representation cones with a regular conical shape, but they define arbitrary regions of space to be mapped onto a pole. Some non-linearities may preserve other structures, such as sigmoid non-linearity which preserves relative Euclidean distances; however, this property is not very useful since other steps in sparse filtering (steps A1 and A4) do not preserve Euclidean distances. From a theoretical perspective, the absolute-value non-linearity is then an optimal choice for the sparse filtering algorithm, in that it preserves the property of collinearity which is also preserved by all the other steps of the algorithm, and it thus guarantees the preservation of cosine neighborhoodness.

3. `https://www.kaggle.com/c/challenges-in-representation-learning-the-black-box-learning-challenge/forums/t/4717/1st-place-entry`

4.11 Preservation of Euclidean Neighborhoodness in the Limit Case

Interestingly, beyond preserving cosine neighborhoodness, we can also prove that, in the limit of an infinite dimensionality ($O \to \infty$), representation cones defined by sparse filtering not only preserve cosine neighborhoodness, but are likely to preserve relations of Euclidean neighborhoodness.

Theorem. Let $\mathbf{x}^{(i)} \in \mathbb{R}^O$ be a point in the original space $\mathbb{R}^O$ and let $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ be a representation cone centered on $\mathbf{x}^{(i)}$ that maps $\mathbf{x}^{(i)}$ onto the pole $\mathbf{p}^{(k)}$. Let us now consider a point $\mathbf{x}^{(j)}$ in the same representation cone $\mathbf{x}^{(j)} \in R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$, that is, a point $\mathbf{x}^{(j)}$ which is mapped by $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ into the neighborhood of $\mathbf{p}^{(k)}$.

Let us assume that: (i) points $\mathbf{x}^{(m)}$ distribute on a limited region of space; and, (ii) points $\mathbf{x}^{(m)}$ distribute uniformly on this limited region of space.

Then, if we consider a point $\mathbf{x}^{(j)}$, for $O \to \infty$, $\frac{P\left(\mathbf{x}^{(j)} \in neighbourhood(\mathbf{x}^{(i)}) | \mathbf{x}^{(j)} \in R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}\right)}{P\left(\mathbf{x}^{(j)} \notin neighbourhood(\mathbf{x}^{(i)}) | \mathbf{x}^{(j)} \in R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}\right)} = 1.$

Proof. Let us consider $\mathbf{x}^{(i)} \in \mathbb{R}^O$ and let us define its neighborhood as the set of points whose distance from $\mathbf{x}^{(i)}$ is at most r. The neighborhood of $\mathbf{x}^{(i)}$ is therefore a hyper-sphere of radius r.

Let us consider now the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ centered on $\mathbf{x}^{(i)}$ and define its area. As this representation cone is inscribing the neighborhood of $\mathbf{x}^{(i)}$, we will take its width around $\mathbf{x}^{(i)}$ to be $2r$. We also define the length of the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ as l. We do not care about the absolute or relative value of l; we only care that it has a finite value, which is guaranteed by the first assumption. In conclusion, the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ is a hyper-cone with a hyper-spherical base of radius $2r$ and height l. For illustration, refer to the schema in Figure 3, where we represented this setup in the case $O = 2$. In two dimensions, the neighborhood of $\mathbf{x}^{(i)}$ is a circle of radius r, while the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ is a triangle with base $2r$ and height l.

Given this setup, we can now compute the volume contained in the neighborhood of $\mathbf{x}^{(i)}$ and the volume contained by the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$. Without loss of generality, we will restrict our attention to the representation cone $R_{\mathbf{x}^{(i)}}^{\mathbf{p}^{(k)}}$ of height l. We could consider a bigger representation cone of height l' and bigger radius $2r'$; however, it will be shown that the ratio between the volumes depends on the dimensionality O, and choosing a different value for the height or the radius does not affect the result.

Let us consider the neighborhood of $\mathbf{x}^{(i)}$, defined as a hyper-sphere of radius r in O dimensions. Its volume can be computed as:

$$V_{sphere}(O, r) = \mathcal{V}_O r^O,$$

where $\mathcal{V}_O$ is the following function:

$$\mathcal{V}_n = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2} + 1\right)},$$

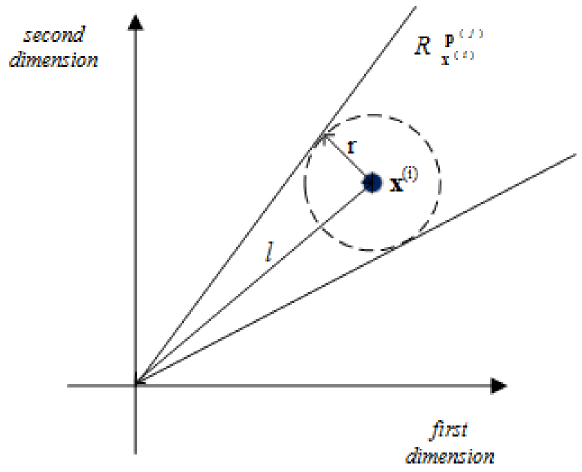


Figure 3: Schema of the the data point $\mathbf{x}^{(i)}$, the neighborhood of $\mathbf{x}^{(i)}$, and the representation cone $R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}$ in two-dimensional space.

and $\Gamma(n)$ is the gamma function.

Let us now consider the representation cone $R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}$, defined as a hyper-cone with base radius r and height l in O dimensions. The volume of the hyper-cone depends on the volume of the lower-dimensional hyper-sphere in the base (Ball, 1997) and it can be computed as:

$$V_{cone}(O, r, l) \;=\; \frac{1}{O} \cdot l \cdot V_{sphere}(O-1, r) = \frac{1}{O} \cdot l \cdot \mathcal{V}_{O-1} \cdot r^{O-1},$$

where $\mathcal{V}_O$ is the function defined above.

Let us now consider the ratio of the volume of the hyper-sphere and the volume of the hyper-cone as a function of the dimensions O:

$$\frac{V_{sphere}(O, r)}{V_{cone}(O, r, l)} \;=\; \frac{\mathcal{V}_O}{\mathcal{V}_{O-1}} \cdot \frac{Or}{l}.$$

The behavior of this quantity as a function of the number of dimensions O may be studied by taking the limit:

$$\lim_{O\to\infty} \frac{V_{sphere}(O, r)}{V_{cone}(O, r, l)} = \lim_{O\to\infty} O \cdot \frac{\mathcal{V}_O}{\mathcal{V}_{O-1}} = \lim_{O\to\infty} O \cdot \frac{\Gamma\left(\frac{O+1}{2}\right)}{\Gamma\left(\frac{O+2}{2}\right)}.$$

Let us rewrite the argument of the gamma function as follows:

$$\lim_{O\to\infty} O \cdot \frac{\Gamma\left(\frac{O}{2}+\frac{1}{2}\right)}{\Gamma\left(\frac{O}{2}+\frac{1}{2}+\frac{1}{2}\right)},$$

and let us substitute $x = \frac{O}{2} + \frac{1}{2}$ and, consequently, $O = 2x - 1$:

$$\lim_{x\to\infty} x \cdot \frac{\Gamma(x)}{\Gamma\left(x+\frac{1}{2}\right)}.$$

Now, by lemma A.6, we can solve the ratio of the gamma functions:

$$\lim_{x\to\infty} x \cdot \frac{1}{\sqrt{x}} = \lim_{x\to\infty} \sqrt{x}.$$

Re-substituting, to make the dependence from the dimensionality O explicit, we get:

$$\lim_{O\to\infty} \sqrt{\frac{O}{2} + \frac{1}{2}}$$

$$\lim_{O\to\infty} \sqrt{O} = \infty.$$

Therefore, as the dimensionality O of the original space increases, the ratio between the volume of the hyper-sphere and the volume of the hyper-cone tends sub-linearly to infinity.

As the dimensionality O increases, the proportion of space accounted for by the volume in the neighborhood of $\mathbf{x}^{(i)}$ increases with respect to the total amount of space accounted for by the whole representation cone $R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}$. Therefore, if we consider a random point $\mathbf{x}^{(j)} \in R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}$ sampled inside the representation cone $R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}$, then the probability of this point falling in the neighborhood of $\mathbf{x}^{(i)}$ grows with the dimensionality O. Then, in the limit, for $O \to \infty$, $\frac{P\left(\mathbf{x}^{(j)} \in neighbourhood(\mathbf{x}^{(i)}) | \mathbf{x}^{(j)} \in R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}\right)}{P\left(\mathbf{x}^{(j)} \notin neighbourhood(\mathbf{x}^{(i)}) | \mathbf{x}^{(j)} \in R^{\mathbf{p}^{(k)}}_{\mathbf{x}^{(i)}}\right)} = 1.$ ■

Notice that this proof is based on two simplified assumptions. First, the region of the original space in which a point $\mathbf{x}^{(j)}$ can fall is limited; this assumption is reasonable because, practically, the range of any feature is always bounded, and, technically, we often rescale or normalize features within bounded intervals. Second, a point $\mathbf{x}^{(j)}$ has a uniform probability of falling anywhere within the area defined by the representation cone $R_{\mathbf{x}^{(i)}}$; this is clearly a simplified assumption because the pdf of the data $p(X)$ may have a very irregular distribution within the area defined by the representation cone $R_{\mathbf{x}^{(i)}}$; however, since such a pdf varies from case to case, assuming a uniform distribution, which is a distribution that maximizes our uncertainty, seems a reasonable choice.

If we accept these assumptions, we can then conclude that, in the case of a high-dimensional space, sparse filtering provides a strong guarantee for the preservation of the cosine neighborhood structure of the data and a weaker asymptotic guarantee for the preservation of the Euclidean neighborhood structure.

4.12 Sparse Filtering for Representation Learning

Given the above results, we may now interpret sparse filtering as a soft clustering algorithm for representation learning.

Indeed, we may state that sparse filtering implicitly makes all the assumptions made by traditional soft clustering algorithms: (i) it aims at discovering less noisy representations whose pdf may automatically be closer to a true stochastic generating process with pdf $p(X^*)$; (ii) it models the true pdf $p(X^*)$ with a mixture model whose components are related to the poles $\mathbf{p}^{(j)}$; and, (iii) it relies on the cosine metric to evaluate relationships of neighborhoodness in the original space $\mathbb{R}^O$. From this perspective, we can interpret the dimensionality of the learned space as the number of clusters for soft clustering, the poles as the cluster centroids in a space described by the cosine metric, the pole pursuit process as the sequential process of update of the location of the centroids, and the learned representation $\mathbf{z}^{(i)}$ as the (stochastic) degree of membership of the original data samples $\mathbf{x}^{(i)}$ to each cluster.

Given this interpretation, we can align and meaningfully compare sparse filtering with other soft clustering algorithms for representation learning that use different metrics. The choice of an appropriate metric is critical for a distance-based clustering algorithm (Xing et al., 2003), and it expresses our understanding on which spatial directions encode relevant changes (Simard et al., 1998). It is natural then to compare sparse filtering with other standard algorithms which adopt the Euclidean metric to explain the data. Preserving the relationships of neighborhoodness under the Euclidean metric means preserving the information conveyed by the pdf $p(X)$ in the representation space defined by the Cartesian product of the random variables $X_1, X_2, \ldots, X_O$. Preserving this information for representation learning makes sense if we expect that the structure of the data (with respect to a set of labels) is better explained by an Euclidean structure. In contrast, preserving the relationships of neighborhoodness under the cosine metric means preserving information conveyed by the pdf $p(X)$ in the representation space defined by the projection into polar (or hyper-spherical) coordinates of the random variables $X_1, X_2, \ldots, X_O$. Preserving such information for representation learning makes sense if we expect that the structure of the data (with respect to a set of labels) is better explained by a radial structure.

5. Empirical Validation

Based on the theoretical analysis provided in the previous section, we conduct a set of simulations aimed at verifying our theoretical results empirically. As a general rule, in order to make our results visualizable and easily understandable, we favor simple simulations in low dimensions; experiments in higher dimensions generalize our results but they do not add anything conceptually new to our conclusions.

5.1 Properties of Sparse Filtering

First, we run simulations on elaborately designed toy data sets in order to validate our basic understanding of sparse filtering. These simulations aim at verifying: (i) the property of homo-representation of collinear points (see Section 4.5); (ii) the property of homo-representation of points with the same moduli (see Section 4.6); (iii) the usefulness of representation cones (see Section 4.8); and, (iv) the dynamics of pole pursuit (see Section 4.8).

We generate a random set of data $\mathbf{X}$ of three samples ($N = 3$) in two-dimensional space ($O = 2$). Each point is generated using spherical coordinates: the radial distance ρ is sampled from a uniform distribution $Unif(-5, 5)$; the angular coordinate θ is set to $\frac{\pi}{3}$ for the first two points and sampled from a uniform distribution $Unif(0, \pi)$ for the third point. A sparse filtering module is trained on this data set in order to learn a new representation of the data in two dimensions ($L = 2$). After training, we create a dense mesh of points $\mathbf{x}$

in the original representation space $\mathbb{R}^O$; we project each point $\mathbf{x}$ to its representation $\mathbf{z}$ in the learned representation space $\mathbb{R}^L$, and we compute the distance from each pole $\mathbf{p}^{(j)}$ in $\mathbb{R}^L$. The plot of each representation cone is then shown as a two-dimensional contour plot in the original space $\mathbb{R}^O$.

Figure 4 shows the state of sparse filtering before training. From the plots 4(b) and 4(d) we can immediately verify the property of homo-representation of collinear points; indeed, in the learned space $\mathbb{R}^L$ the collinear points occupy the same location and their matrix representation is the same. From the plots 4(e) and 4(f) we can verify the existence of representation cones in the original space $\mathbb{R}^O$ and appreciate several of the properties discussed above (existence of L representation cones; O-dimensionality of each representation cone; bounds of representation cones; and, complementarity of the representation cones). At the same time, the symmetric structure in the same plots 4(e) and 4(f) validate the properties of homo-representation of points with the same moduli. Notice that, at this point, after the random initialization of the weight matrix $\mathbf{W}$, the quality of the representations generated by the untrained sparse filtering module is far from satisfactory.

Figure 5 shows the state of sparse filtering at the end of training. From the plots 4(b) and 4(d) we can see that the trained sparse filtering module has found an optimal solution that maps all the points to poles; as expected, the collinear points are mapped to the same pole, while the third point is mapped to the remaining pole. From the plots 4(e) and 4(f) we can validate our intuition about pole pursuit; indeed, training corresponded to a rotation of the representation poles in order to center them on the available samples. Moreover, the same plots 4(e) and 4(f) also confirm the last properties of representation cones which we could not evaluate at the beginning of our simulation (association to the pole; closeness to a pole).

5.2 Preservation of Cosine Neighborhoodness

Next, we run more simulations on other toy data sets in order to validate the properties of data structure preservation in sparse filtering. These simulations aim at verifying: (i) that sparse filtering preserves a structure defined by cosine neighborhoodness; and, (ii) that the absolute-value non-linearity is crucial in preserving structure and substituting it with other non-linearities (such as, sigmoid or ReLU) negates this property.

We generated a random set of data $\mathbf{X}$ of nine samples ($N = 9$) in two-dimensional space ($O = 2$). Each point is generated using spherical coordinates. The first three points have a radial distance ρ sampled from a uniform distribution $Unif\left(-2, 0\right)$ and an angular coordinate θ sampled from a uniform distribution $Unif\left(\frac{\pi}{9} - \eta, \frac{\pi}{9} + \eta\right)$; the second three points have a radial distance ρ sampled from a uniform distribution $Unif\left(0, 3\right)$ and an angular coordinate θ sampled from a uniform distribution $Unif\left(\frac{2\pi}{9} - \eta, \frac{2\pi}{9} + \eta\right)$; the last three points have a radial distance ρ sampled from a uniform distribution $Unif\left(2, 4\right)$ and an angular coordinate θ sampled from a uniform distribution $Unif\left(\frac{4\pi}{9} - \eta, \frac{4\pi}{9} + \eta\right)$. The parameter η is meant to represent a form of noise and its value is set to $\eta = \frac{\pi}{45}$. In this way, we generate three clusters of points, such that the cosine distances among the points belonging to the same cluster are small, while the distances among points belonging to different clusters are large. Three implementations of sparse filtering with different non-

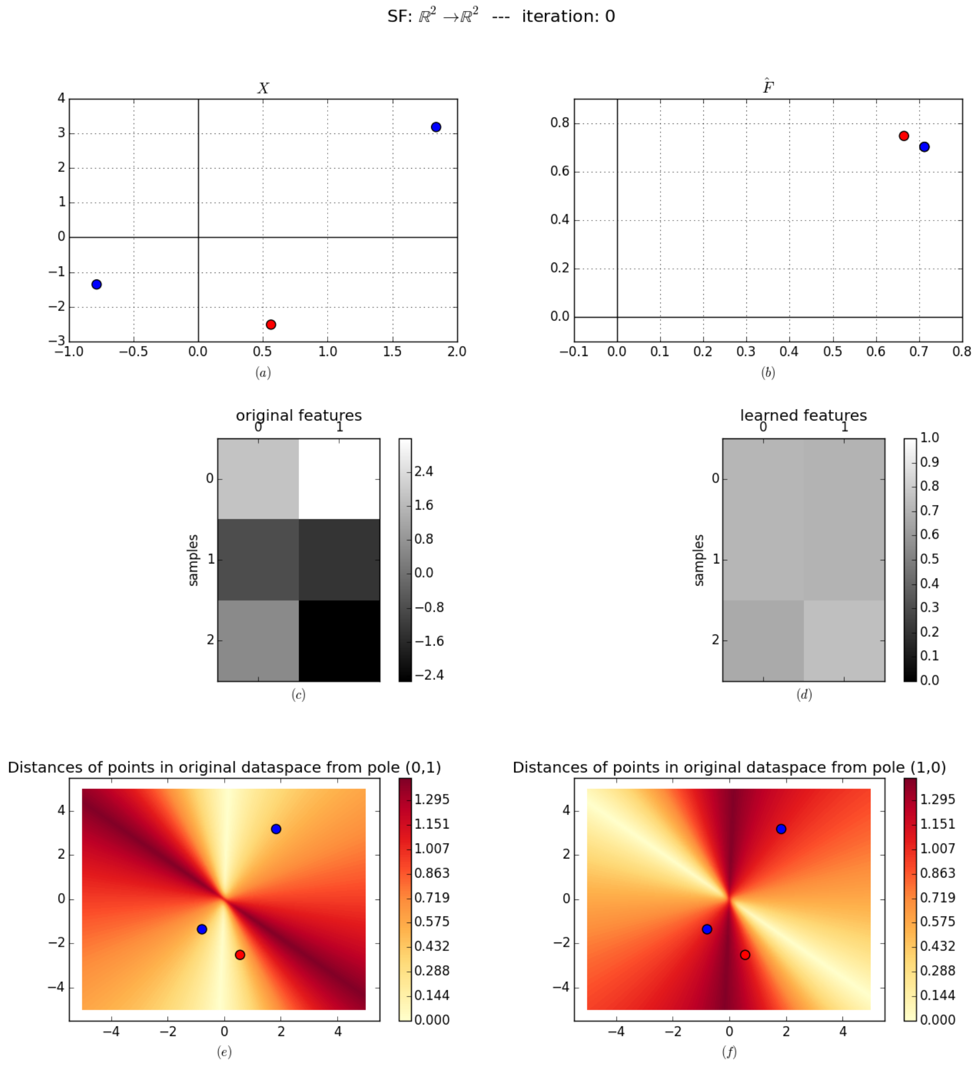


Figure 4: Experimental validation of the properties of sparse filtering (homo-representation of collinear points, homo-representation of points with the same moduli, representation cones).
Data is generated as explained in the text (the blue dots represent collinear points). (a) Data in the original representation space $\mathbb{R}^O$; (b) data in the learned representation space $\mathbb{R}^L$; (c) matrix plot of the original data $\mathbf{X}$; (d) matrix plot of the learned representations $\mathbf{Z}$; (e) plot of first representation cone showing distances from the pole $\mathbf{p}^{(1)} = [0,1]^T$; (f) plot of the second representation cone showing distances from the pole $\mathbf{p}^{(2)} = [1,0]^T$.

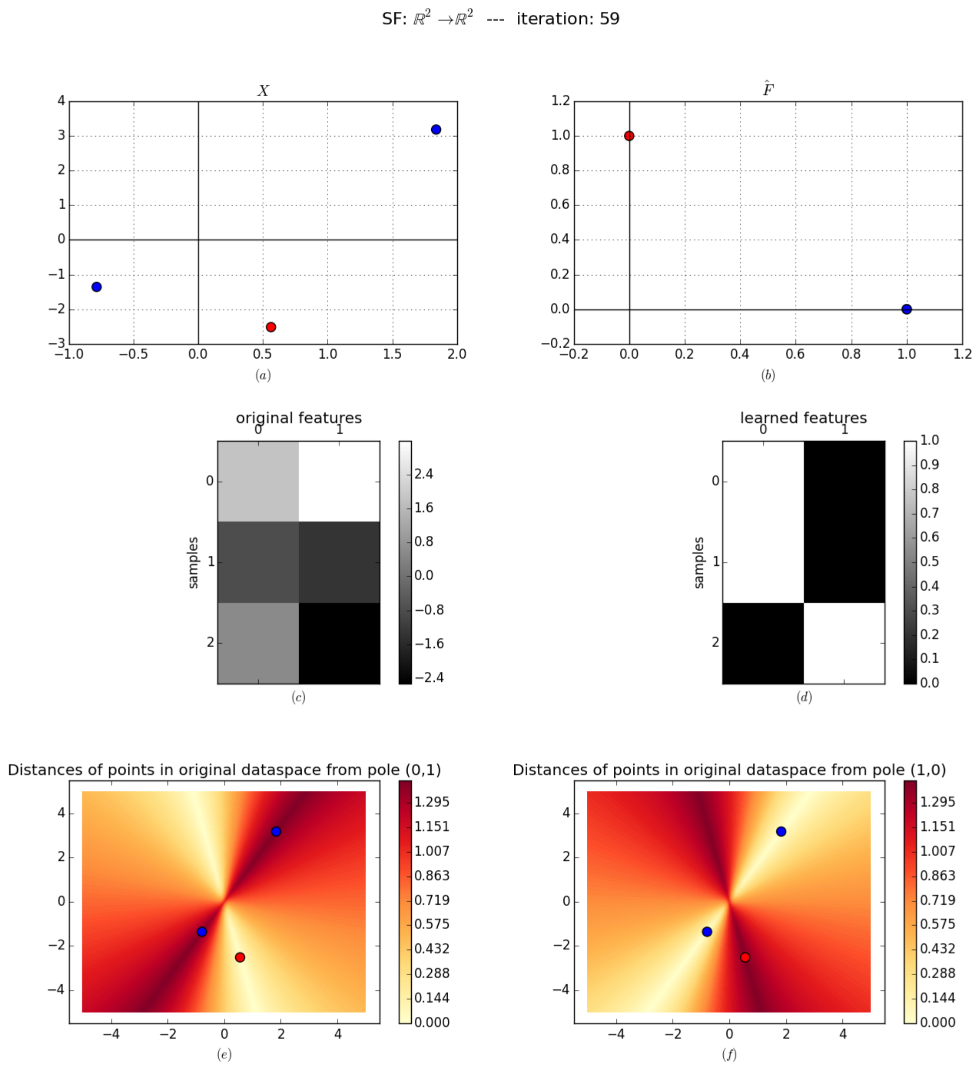


Figure 5: Experimental validation of the properties of sparse filtering (pole pursuit). Data is generated as explained in the text. The meaning of the subplots is the same as in Figure 4.

linearities (absolute-value, sigmoid, and ReLU[4]) are used to learn a new representation of the data in three dimensions ($L = 3$).

Figure 6 shows the state of the modules of the three implementations of sparse filtering at the end of the training. From the plots 4(a)-4(c) we can immediately verify that sparse filtering with an absolute-value non-linearity preserves cosine neighborhoodness. The representation-cone plots show that points with similar angular coordinates fall within the same representation cones. The matrix plot shows that points with similar angular coordinates are projected onto very similar representations; in other words, points that originally had a small cosine distance are projected onto almost identical representations. On the other hand, from plots 4(d)-4(i) we can easily see that sparse filtering with an alternative non-linearity does not preserve cosine neighborhoodness. The representation-cone plots show that the sigmoid and the ReLU non-linearity do not induce representation cones, but, instead define large regions of the original space to be mapped onto a pole. Since these regions are not rigidly bounded (as in the case of the absolute-value non-linearity) several points are indistinctly mapped onto a pole. The matrix plots show that the representations computed by these alternative sparse filtering modules are not related to the original cosine distances anymore; points originally belonging to the same cluster are mapped to opposite representations, and, vice versa, points originally belonging to different clusters are mapped to identical representations.

5.3 Sparse Filtering for Representation Learning

In the following set of simulations, we compare sparse filtering against another unsupervised algorithm, the soft k-means algorithm (MacKay, 2003), in order to show under which conditions sparse filtering is a good choice for processing data. These simulations aim at verifying the following intuitive implication: if the structure of the data (with respect to a specific set of labels) is better explained by the cosine metric, then sparse filtering is likely to be a good option for unsupervised learning.

In our comparison, we measure sparse filtering against the soft k-means algorithm. We choose this algorithm for the following reason: (i) like sparse filtering, the soft k-means algorithm is a soft clustering algorithm producing sparse representations; (ii) the algorithm is based on the Euclidean metric, thus providing a different interpretation of the data from sparse filtering; and, (iii) it is a well-known and easy-to-interpret algorithm (even if, analogous results may be obtained by other algorithms, such as Gaussian mixture models or sparse auto-encoders).

To validate our hypothesis, we generate two data sets, $\mathbf{X}_{Euclid}$ and $\mathbf{X}_{cosine}$. The data set $\mathbf{X}_{Euclid}$ contains data explained by the Euclidean metric. It is composed of nine samples ($N = 9$) in two dimensions ($O = 2$). The first three points are sampled from a multivariate normal distribution $\mathcal{N}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} .05 & 0 \\ 0 & .05 \end{bmatrix}\right)$; the second three points are sampled from a multivariate normal distribution $\mathcal{N}\left(\begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} .05 & 0 \\ 0 & .05 \end{bmatrix}\right)$; the last three points are

4. ReLU has been implemented in a soft version, like the absolute-value: $softReLU(x) = \begin{cases} x & if\, x > 0 \\ \epsilon & otherwise \end{cases}$ for $\epsilon = 10^{-8}$

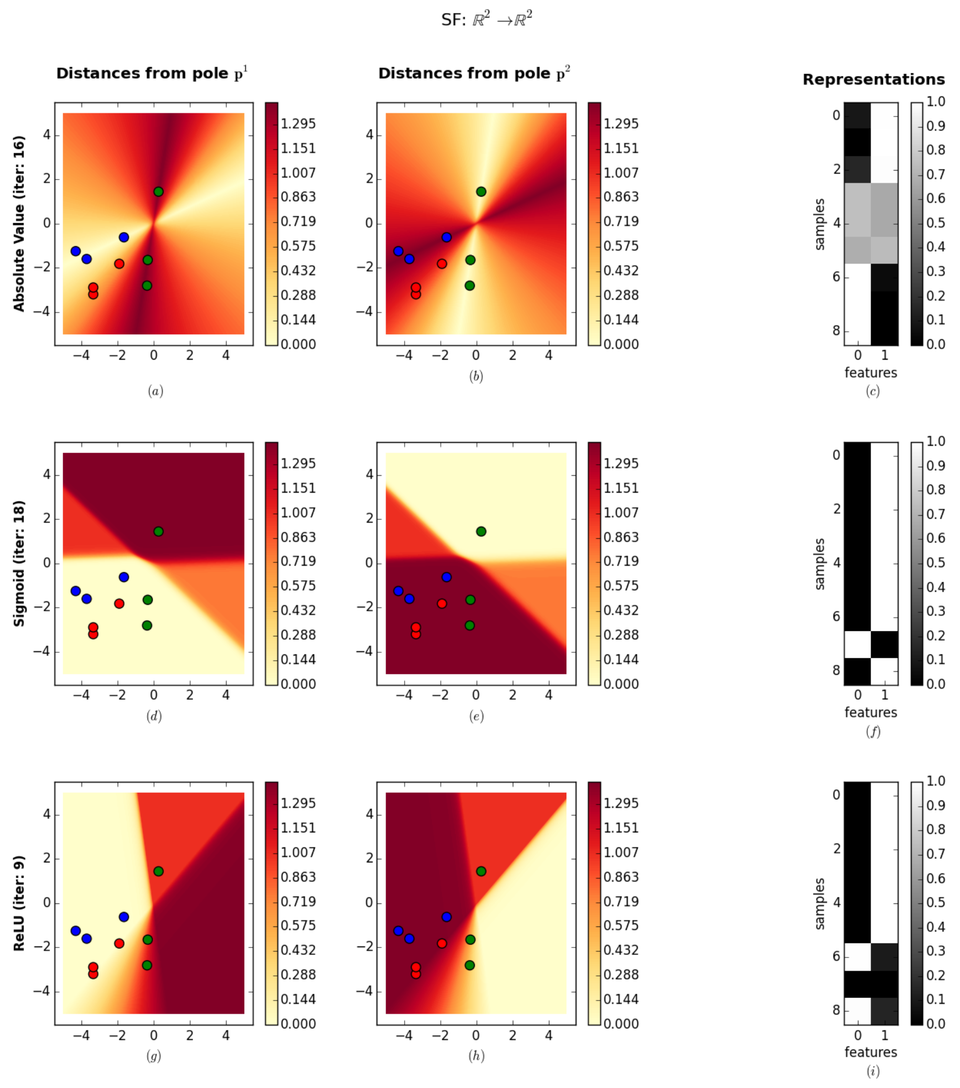


Figure 6: Experimental validation of the preservation of cosine neighborhoodness. Data is generated as explained in the text (first set of points in blue, second set of points in red, third set of points in green). (a, d, g) Plot of the first representation cone showing distances from the pole $\mathbf{p}^{(1)} = [0, 1]^T$, respectively for the sparse filtering with absolute-value, sigmoid, and ReLU non-linearity; (b, e, h) plot of the second representation cone showing distances from the pole $\mathbf{p}^{(2)} = [1, 0]^T$, respectively for the sparse filtering with absolute-value, sigmoid, and ReLU non-linearity; (c, f, i) matrix plot of the learned representations $\mathbf{Z}$, respectively for the sparse filtering with absolute-value, sigmoid, and ReLU non-linearity.

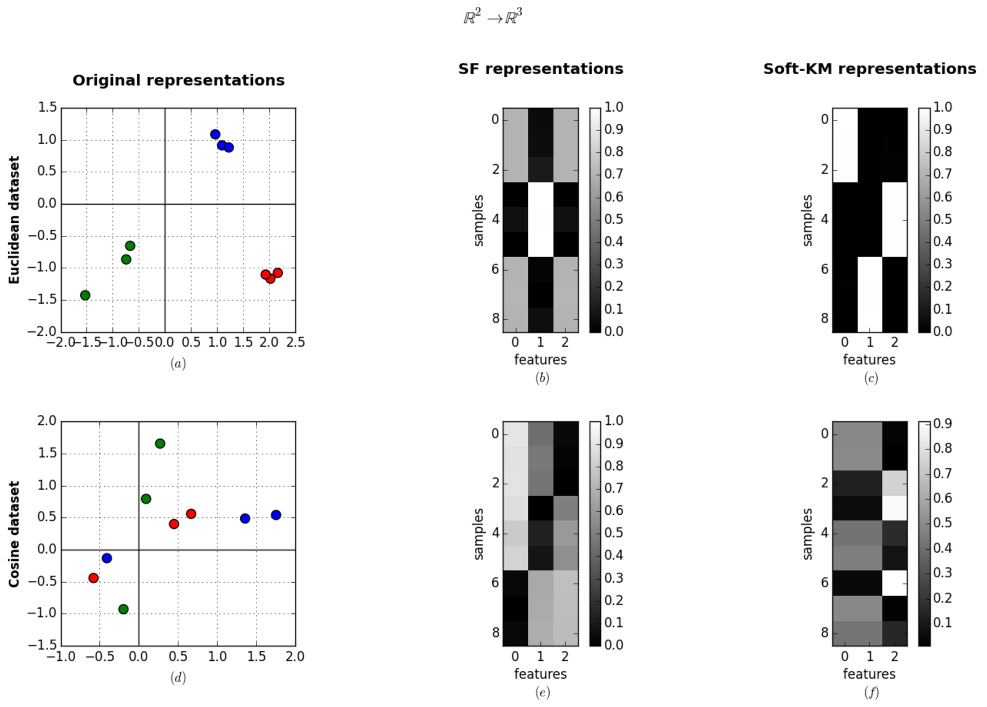


Figure 7: Representation data with Euclidean and cosine data structure.
Data is generated as explained in the text (first set of points in blue, second set of points in red, third set of points in green). (a, d) Samples in the original space; (b, e) matrix plot of the representations learned by sparse filtering; (c, f) matrix plot of the representations learned by soft k-means.

sampled from a multivariate normal distribution $\mathcal{N}\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} .05 & 0 \\ 0 & .05 \end{bmatrix}\right)$. The data set $\mathbf{X}_{cosine}$ contains data explained by the cosine metric. The data is generated following the same protocol used in the simulation in Section 5.2. Sparse filtering is used to learn a new representation of the data in three dimensions ($L = 3$).

From Figure 7, we can see that our understanding of sparse filtering is correct: if the data we are processing are better explained by the cosine metric, then sparse filtering produces a good representation; otherwise, if the data are better explained by the Euclidean metric, then it is better to opt for a different unsupervised learning algorithm, such as soft k-means. In the case of the data set with Euclidean structure, plot 7(b) shows that sparse filtering is not able to preserve the identity of the generating clusters, and indeed it maps samples from the first and the third clusters onto the same representation (because of their collinearity); instead, plot 7(c) shows that soft k-means algorithm maps points from different clusters to different representations. In contrast, in the case of the data set with cosine structure, plot 7(e) shows that sparse filtering preserves the identity of the generating clusters, while plot

7(f) shows that the soft k-means algorithm is unable to map samples from different clusters onto consistent representations.

5.4 Sparse Filtering on Real Data Sets

In this last set of simulations we apply our discoveries about sparse filtering to real-world data sets to further verify our results. Once again, these experiments aim at validating the connection between the radial structure of the data and the success of sparse filtering. In the first simulation, we extend the result that we proved in Section 5.3 for toy data sets to real data sets; that is, we verify the direct implication: *if* the structure of the data (with respect to a specific set of labels) is better explained by the cosine metric, *then* sparse filtering is likely to be a good option for unsupervised learning. In the second simulation, we validate, instead, the reverse implication: *if* sparse filtering happens to be a good option for unsupervised learning, *then* the structure of the data (with respect to a specific set of labels) is likely to be better explained by the cosine metric.

Notice that when dealing with real data sets, it is very challenging to assess the structure of the data. In low dimensions, with few samples and with the simplified assumption that all the data belonging to a given class are generated by a single highly localized cluster (as in the previous simulations), a simple visualization of the data is enough to understand which metric is underlying the data. Thanks to these simplified assumptions, a straight computation of distances among samples belonging to the same class is sufficient to decide which metric best describes the data. However, when we consider real data sets, we have to deal with samples in high dimensions, with a large number of samples, and with the fact that samples belonging to the same class may be generated by different clusters spread throughout the space; in this case, we can not rely on visualization anymore. In order to explore high-dimensional data, we decided to rely on the k-nearest neighbors algorithm (KNN). We implemented two versions of KNN, one selecting k neighbors according to the Euclidean distance and one selecting k neighbors according to the cosine distance[5]. If the structure of the data (with respect to a set of labels) is better explained by the Euclidean distance, we expect KNN with the Euclidean metric to provide better results; alternatively, if the structure of the data (with respect to a set of labels) is better explained by the cosine distance, we expect KNN with the cosine metric to provide better results.

Berlin Emotional data set. The Berlin Emotional (EMODB) data set is a well-known audio data set in the emotion recognition community (Burkhardt et al., 2005); it contains recordings of ten German actors expressing seven different types of emotions. We opted for this data set to validate the direct implication between data structure and effectiveness of sparse filtering for the following reasons. (i) Samples in EMODB naturally lend themselves to alternative labellings; in particular, the same data may be used both for speaker recognition (using subject labels) and for emotion recognition (using emotional labels). (ii) The same set of Mel-frequency spectrum (Childers et al., 1977) coefficient (MFCC) features may

5. The KNN using cosine distance has been implemented relying on the "trick" that the cosine distance between vectors $\mathbf{v}, \mathbf{u}$ is the same as the Euclidean metric on the ℓ_2-normalized vectors. Therefore, we perform an ℓ_2-normalization of each data sample and then we run KNN with Euclidean distance, re-using off-the-shelf KNN code optimized for the Euclidean metric.

reasonably be used both for speaker recognition and for emotion recognition; indeed, MFCC features were primarily designed for speaker recognition, but they proved to be relevant for emotion recognition as well (Wu et al., 2010; Schuller et al., 2011). Using the same features we explore the data under different labeling.

We first explore structure of the data with respect to the two different labeling systems in order to evaluate whether the Euclidean distance or the cosine distance better explains the structure of the data. We run the KNN algorithm with different values of neighbors ($k = \{2, 3, 5, 7, 10, 15, 20, 25, 50, 75, 100\}$); for each configuration of KNN, fifty simulations are executed; in each simulation the data set is randomly partitioned into a training data set (900 samples) and a test data set (311 samples); KNN is then trained and tested using one of the two available metrics.

After this analysis, we use both an Euclidean-based unsupervised learning algorithm, Gaussian mixture model (Bishop, 2007), and a cosine-based unsupervised learning algorithm, sparse filtering, to project the data into an L-dimensional space. We opted for the Gaussian mixture of models (GMM) algorithm because it is based on the Euclidean metric and yields better performance than the soft k-means algorithm. After processing the data, we then run a simple linear SVM classifier on the processed data and we analyze how our observations on the structure of the data relate with the actual classification performance. We consider several values of dimensionality ($L = \{2, 3, ..., 40\}$); for each configuration, fifty simulations are executed; as before, in each simulation the data set is randomly partitioned into a training data set (900 samples) and in a test data set (311 samples).

Figure 8(a) shows that the structure of EMODB data with respect to emotional labels is better explained by the Euclidean distance. This result is further confirmed by the classification with the linear SVM module in Figure 8(b). Classification using the GMM-processed data with low learned dimensionality ($L \leq 15$) returns an accuracy that is significantly better than using sparse filtering-processed data (Wilcoxon signed-rank test, p-value $P = 5 \cdot 10^{-85}$); however, in higher dimensions the classification with sparse filtering-processed data approaches and overtakes the accuracy obtained using GMM-processed data. In general, in low dimensions, the Euclidean structure assumed by GMM explains the data better; in high dimensions, sparse filtering provides good results (most likely thanks to the property of sparsity) but the gap between the accuracy provided by the two representations remains limited. On the other hand, Figure 9(a) shows that the structure of EMODB data with respect to the speaker identity labels is better explained by the cosine distance. This result is further confirmed by the classification with the linear SVM module in Figure 9(b). Classification using the sparse filtering-processed data returns, for all learned dimensionality, an accuracy that is significantly better than GMM-processed data (Wilcoxon signed-rank test, p-value $P = 4 \cdot 10^{-307}$). The assumption of the cosine metric allows sparse filtering to explain the data much better, as is evident from the large gap between the accuracy provided by the two representations.

These results confirm a connection between the radial structure of the data (with respect to a set of labels) and the usefulness of sparse filtering.

Kaggle Black Box Learning Challenge data set. The Kaggle Black Box Learning Challenge (KBBLC) data set is a visual data set made up of obfuscated images of house numbers; the original images are taken from the well-known Street View House Numbers

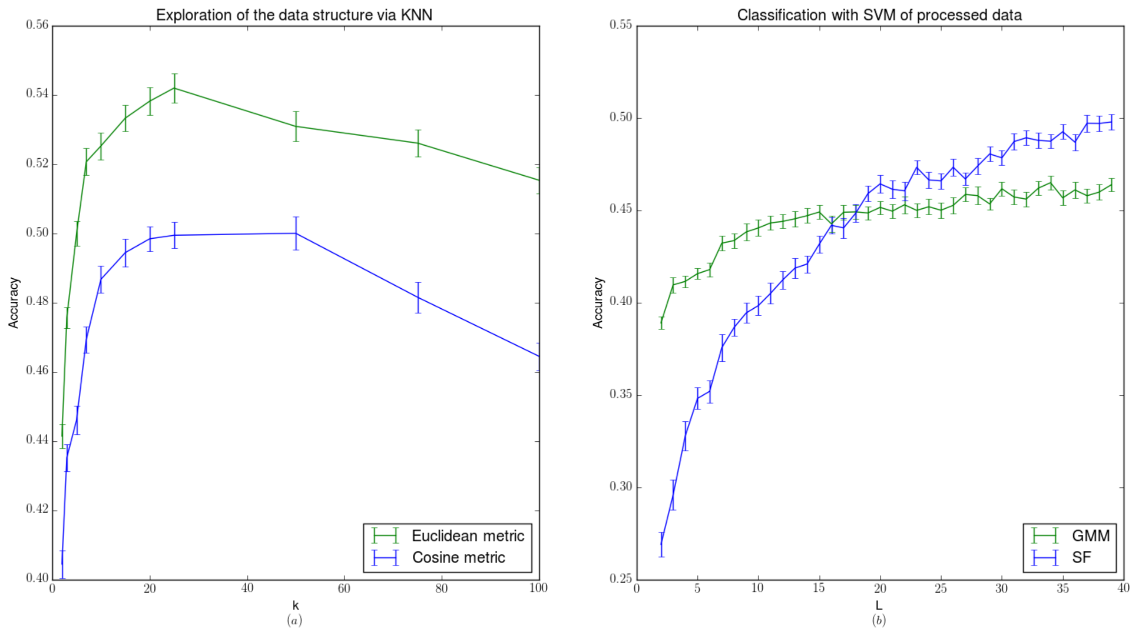


Figure 8: Analysis of the data structure and the classification of the EMODB data set with respect to emotion labels.
Classification is performed as explained in the text. (a) Exploration of the data via KNN with Euclidean metric (green line) and with cosine metric (blue line); (b) Classification using a linear SVM after processing with a GMM algorithm (green line) and with sparse filtering (blue line). The plot shows the average accuracy and the standard error of SVM (over fifty simulations).

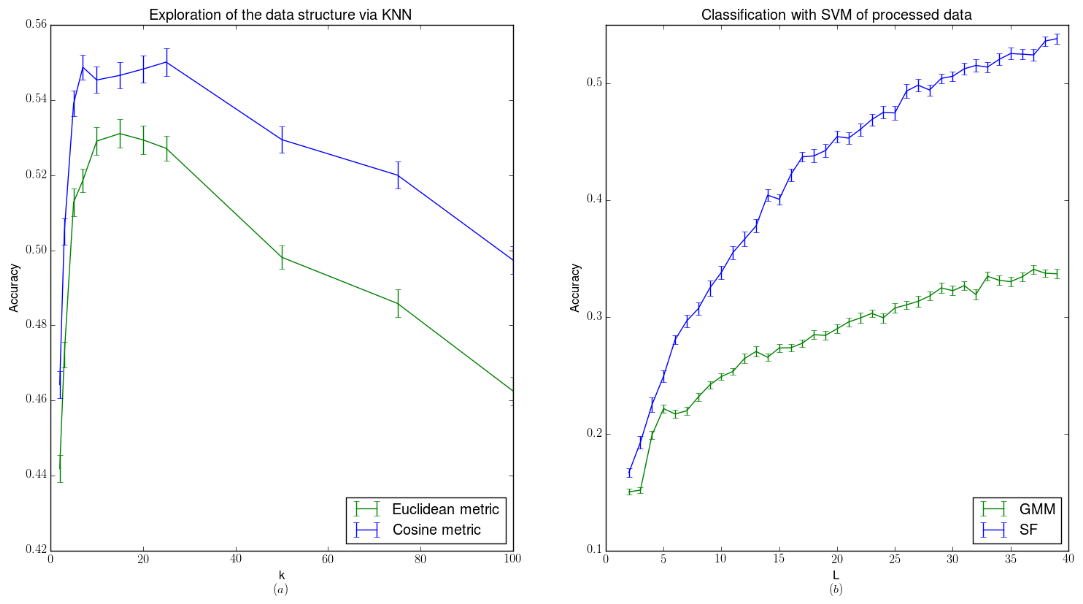


Figure 9: Analysis of the data structure and the classification of the EMODB data set with respect to subject labels.
The meaning of the subplots is the same as in Figure 8.

(SVHN) data set (Netzer et al., 2011). Each sample in the KBBLC data set contains a single obfuscated digit and it is accompanied by a label specifying the value of the digit. We opted to validate the reverse implication between data structure and effectiveness of sparse filtering on this data set for the following reasons. (i) Sparse filtering provided state-of-the-art performance in the competitive KBBLC contest, thus showing that sparse filtering was a particularly suitable choice for this data set. (ii) The KBBLC data set is available with labels. During the challenge the authors provided obfuscated data without labels; however, after the challenge they revealed the original source of the data[6] and they released the code they used for obfuscation[7]. Thanks to this information, we were able to retrieve a large amount of data and obfuscate it, and thus recreate the original conditions of the challenge. However, differently from the challenge, we retain the labels in order to explore the structure of the data. (iii) During the challenge, samples from the data sets were processed without pre-processing. Since sparse filtering was directly applied to the samples, we can analyze the structure of the samples straightforwardly. This condition is not always true. If we consider other data sets on which sparse filtering provided good results, such as CIFAR-10 or STL-10, sparse filtering was not applied to the original samples but to random patches extracted from the images; in this case, we should not analyze the data structure of the

6. `http://ufldl.stanford.edu/housenumbers/`
7. `https://www.kaggle.com/c/challenges-in-representation-learning-the-black-box-learning-challenge/forums/t/5167/the-data`

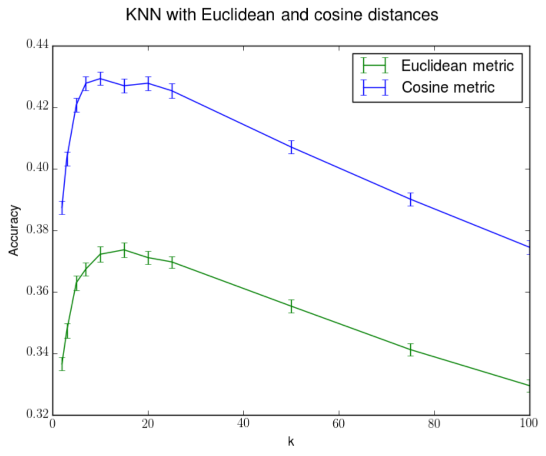


Figure 10: Analysis of the data structure of the Kaggle Black Box Learning Challenge data set.
The KNN with Euclidean metric (green line) and with cosine metric (blue line) has been used to explore the structure of the data. The plot shows the average accuracy and the standard error of KNN (over five simulations).

original samples, but the data structure of the patches. However, patches are not labeled, which hinders our ability to carry out an analysis of the data structure.

In exploring the structure of the data (with respect to the digit labels), we aim at evaluating whether the Euclidean distance or the cosine distance better explains the structure of the data. We run the KNN with the same settings as in the previous experiment. In each simulation a random subset of 10000 samples from the data set was selected and then partitioned into a training data set (9000 samples) and a test data set (1000 samples). KNN was then trained and tested using one of the two available metrics.

Figure 10 confirms our intuition. For all the different values of k we considered, the cosine distance proved to be a better metric to explain the structure of the data in the Kaggle Black Box Learning Challenge. This provides an explanation why sparse filtering proved so useful with the KBBLC data, when compared to other standard unsupervised learning algorithms, especially those based on the Euclidean metric. This result agrees with the fact that the Euclidean metric is not a suitable metric for measuring distances among samples of digits represented in the pixel space; other distances less sensitive to irrelevant transformations, such as tangent distance (Simard et al., 1998), are known to be better choices.

6. Discussion

The theoretical analysis and the empirical verification we performed allow us to conclude that our thesis is correct: sparse filtering satisfies the informativeness principle through the maximization of the proxy of sparsity, and it satisfies the infomax principle through the constraint of preservation of the radial structure of the data. In particular, sparse filtering is able to implicitly preserve the mutual information between the original representations $\mathbf{x}^{(i)}$ and the learned representations $\mathbf{z}^{(i)}$ through the preservation of the structure defined by cosine neighborhoodness.

In our experiments, we showed that sparse filtering operates as an unsupervised soft clustering algorithm based on the cosine metric. This allowed us to contrast the results of sparse filtering with other standard algorithms for clustering based on the Euclidean metric (for instance, soft k-means or Gaussian mixture models). Sparse filtering, thus, does not provide a better processing of the data in absolute terms, but instead provides an alternative interpretation of the data based on a different metric.

Consequently, we have been able to highlight the conditions under which sparse filtering may be expected to perform significantly better than the standard Euclidean-based alternatives. We showed that whenever the structure of the data (with respect to a set of labels) can be explained through cosine distances, sparse filtering is able to provide cutting-edge performance. Indeed, sparse filtering may be seen as an algorithm approximately transforming cosine distances in the original space into Euclidean distances in the representation space; if cosine distances are meaningful (with respect to a set of labels), then sparse filtering will provide a representation that is especially useful for the large set of standard classifiers that rely on the Euclidean metric in their learning process.

The ideal scenario in which to employ sparse filtering is one in which relevant information is brought by the radial structure of the data. It is normally assumed that the data points $\mathbf{x}^{(i)}$ are best explained as samples from a multivariate random variable $X = (X_1, X_2, \ldots, X_O)$, where each random variable X_j describes a component $\mathbf{x}_j^{(i)}$. However, given the data points $\mathbf{x}^{(i)}$, it is possible to assume that the generating process is better described by a multivariate random variable $X' = \left(X_1', X_2', \ldots, X_{O-1}'\right)$, where each random variable describes an angular coordinate θ_j of $\mathbf{x}_j^{(i)}$. In such a case, sparse filtering is a very reasonable choice for unsupervised representation learning.

In our experiments, we were aware a priori of the metric (either Euclidean or cosine) underlying a synthetic data set. However, in a real-world setting, such knowledge may not be available. Sparse filtering, though, is a scalable and efficient algorithm and it may be possible to test it on the data even without knowing a priori the metric underlying it. The usefulness of polar coordinates in several scientific fields and physical applications may suggest that interpreting data according to cosine distance could be a sensible choice. More rigorously, it is possible to run a simple exploratory analysis of the data (using, for instance KNN) to assess which metric seems better suited for the data set.

Additionally, we proved that, even if we believe that the structure of the data is better explained in terms of Euclidean distances in Cartesian coordinates, in high dimensions, sparse filtering may still provide good results. This is justified by the fact that, under the assumptions we made, as we increase the number of dimensions, the probability that unrelated points with high Euclidean distance will have the same angular coordinates θ_i will necessarily decrease (Section 4.11).

7. Concluding Remarks

In this paper, we have explained *why* sparse filtering works (by proving its property of preservation of cosine neighborhoodness) and *when* it should be expected to provide useful representations (by considering the data structure of the samples).

At the foundation of our analysis lies the understanding that sparse filtering must preserve some information carried by the pdf $p(X)$ about the true pdf $p\left(X^{*}\right)$. Despite sparse filtering ignoring the problem of explicitly modeling the true pdf $p\left(X^{*}\right)$, the algorithm is hard-coded with an implicit constraint that guarantees the preservation of some data structure. This is clearly a specific conclusion about the particular algorithm of sparse filtering, but we can expect that this principle will be applicable to the whole class of feature distribution learning algorithms. We might expect that any feature distribution learning algorithm, in order to be successful, must take into account, through constraints or priors, the problem of preserving the mutual information between the original representations $\mathbf{x}^{(i)}$ and the learned representations $\mathbf{z}^{(i)}$. Being aware of this requirement could be of help in designing and analyzing new feature distribution learning algorithms; it may, for instance, help us to avoid solutions (such as sparse filtering with a sigmoid or ReLU non-linearity) that, being unable to preserve any structure of the data, are bound to produce unsatisfactory representations.

Our theoretical analysis and simulations were not designed to show that sparse filtering is able to provide state-of-the-art performance against other algorithms, but, instead, to show how the implicit assumptions and constraints of sparse filtering make it better suited for certain scenarios instead of others. In particular, we showed that the success conditions of sparse filtering are tied to the structure underlying the data. Consistently with the no-free lunch theorem (Wolpert and Macready, 1997), we reached the conclusion that sparse filtering is not a better algorithm than other Euclidean-based clustering algorithms, but that there is a specific set of problems (in which the data structure is explained by the cosine metric) where the performance of sparse filtering is excellent, balanced by a set of problems (in which the data structure is explained by the Euclidean metric) where its performance is less outstanding. This led us to interpret the representation of sparse filtering as a "view" of the data according to the cosine metric. Whenever the underlying structure of the data is unknown, it may also be possible to combine this "cosine view" of the data provided with a more standard "Euclidean view" of the data. Combining these two different views could provide representations with more discriminative power.

A promising avenue in our ongoing research is the extension of sparse filtering to semi-supervised learning. Indeed, the paradigm of feature distribution learning seems perfectly suited for the scenario in which we are provided with few labeled samples and many unlabeled samples: following the approach of feature distribution learning we may exploit the information carried by the labeled samples to better shape the feature distribution $p(Z)$, without addressing the problem of estimating the true pdf $p\left(X^{*}\right)$; at the same time, the constraint of sparsity would help us to not overfit, and the constraint of structure preservation would help us to preserve the information conveyed by $p(X)$. In particular, assuming some regularity in the original representation space, we hypothesize that we could use the information in the labeled samples to address the problem of covariate shift (Sugiyama and Kawanabe, 2012) in a semi-supervised learning scenario.

Appendix A. Additional Proofs

We here provide the proofs of propositions and lemmas in the main text.

A.1 Preservation of Collinearity under Linear Transformation

Lemma. Let us consider $\mathbf{v}, \mathbf{u} \in \mathbb{R}^O$, two generic collinear vectors, and let $f : \mathbb{R}^O \to \mathbb{R}^L$ be a linear transformation encoded by the matrix $\mathbf{W}$. Then $\mathbf{Wv}, \mathbf{Wu} \in \mathbb{R}^L$ are also collinear.

Proof. Let us consider the two collinear vectors $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & ... & v_o \end{bmatrix}^T$ and $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & ... & u_o \end{bmatrix}^T$ in $\mathbb{R}^O$. Since the two vectors are collinear, they share the same angular coordinates ϕ_i to the origin. We can therefore rewrite the vectors using spherical coordinates:

$$\mathbf{v} = \begin{bmatrix} \rho_v \cos(\phi_1) \\ \rho_v \sin(\phi_1)\cos(\phi_2) \\ \rho_v \sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ ... \\ \rho_v \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} \rho_u \cos(\phi_1) \\ \rho_u \sin(\phi_1)\cos(\phi_2) \\ \rho_u \sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ ... \\ \rho_u \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \end{bmatrix}.$$

Since both the radial coordinate ρ_v and ρ_u are real numbers, we can definitely find $\alpha \in \mathbb{R}$ such that $\alpha = \frac{\rho_u}{\rho_v}$. Now, substituting ρ_u with $\alpha\rho_v$ we can rewrite:

$$\mathbf{u} \quad = \alpha \quad \begin{bmatrix} \rho_v \cos(\phi_1) \\ \rho_v \sin(\phi_1)\cos(\phi_2) \\ \rho_v \sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ ... \\ \rho_v \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \end{bmatrix}.$$

Let us now consider the linear transformation described by $\mathbf{W} \in \mathbb{R}^{L \times O}$:

$$\mathbf{W} = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1O} \\ W_{21} & W_{22} & \cdots & W_{2O} \\ \cdots & \cdots & \cdots & \cdots \\ W_{L1} & W_{L2} & \cdots & W_{LO} \end{bmatrix}.$$

The projection $\mathbf{Wv}$ can be computed as:

$$\mathbf{Wv} \quad = \quad \begin{bmatrix} W_{11}\rho_v \cos(\phi_1) + W_{12}\rho_v \sin(\phi_1)\cos(\phi_2) + \cdots + W_{1O}\rho_v \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \\ W_{21}\rho_v \cos(\phi_1) + W_{22}\rho_v \sin(\phi_1)\cos(\phi_2) + \cdots + W_{2O}\rho_v \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \\ \\ W_{L1}\rho_v \cos(\phi_1) + W_{L2}\rho_v \sin(\phi_1)\cos(\phi_2) + \cdots + W_{LO}\rho_v \sin(\phi_1) ... \sin(\phi_{O-2})\cos(\phi_{O-1}) \end{bmatrix}.$$

Let us call the new components of $\mathbf{Wv}$ as a_i:

$$\mathbf{Wv} = \left[\begin{array}{ccccc} a_1 & a_2 & a_3 & ... & a_L \end{array} \right]^T.$$

If we want to find the new radial coordinates ξ_i of $\mathbf{Wv}$ we could resort to the standard formula:

$$\xi_i = \arccos \frac{a_i}{\sqrt{a_L^2 + a_{L-1}^2 + \cdots + a_i^2}}.$$

The projection $\mathbf{Wu}$ can be computed as:

$$\mathbf{Wu} = \mathbf{W}\left(\alpha \mathbf{v}\right),$$

and by linearity:

$$\mathbf{W}\left(\alpha \mathbf{v}\right) = \alpha \left(\mathbf{Wv}\right) = \alpha \left[\begin{array}{ccccc} a_1 & a_2 & a_3 & ... & a_L \end{array} \right]^T.$$

If we want to find the new radial coordinates ζ_i of $\mathbf{Wu}$ we could resort to the same formula used above:

$$\begin{aligned} \zeta_i &= \arccos \frac{\alpha a_i}{\sqrt{\left(\alpha a_L\right)^2 + \left(\alpha a_{L-1}\right)^2 + \cdots + \left(\alpha a_i\right)^2}} \\ &= \arccos \frac{a_i}{\sqrt{\left(a_L^2 + a_{L-1}^2 + \cdots + a_i^2\right)}} \\ &= \xi_i. \end{aligned}$$

Therefore the radial coordinates ξ_i of $\mathbf{Wv}$ and the radial coordinates ζ_i of $\mathbf{Wu}$ are the same. Hence $\mathbf{Wv}$ and $\mathbf{Wu}$ are collinear. ■

A.2 Preservation of Collinearity under Absolute-Value

Lemma. Let us consider $\mathbf{v}, \mathbf{u} \in \mathbb{R}^L$, two generic collinear vectors, and let $f : \mathbb{R}^L \rightarrow \mathbb{R}^L$ be the element-wise absolute-value function. Then $f\left(\mathbf{v}\right), f(\mathbf{u}) \in \mathbb{R}^L$ are also collinear.

Proof. Let us consider the two collinear vectors $\mathbf{v} = \left[\begin{array}{ccccc} v_1 & v_2 & v_3 & ... & v_L \end{array} \right]^T$ and $\mathbf{u} = \left[\begin{array}{ccccc} u_1 & u_2 & u_3 & ... & u_L \end{array} \right]^T$ in $\mathbb{R}^L$. Since the two vectors are collinear, they share the same angular coordinates ϕ_i to the origin. We can therefore rewrite the vectors using spherical coordinates:

$$\mathbf{v} = \left[\begin{array}{c} \rho_v \cos\left(\phi_1\right) \\ \rho_v \sin\left(\phi_1\right) \cos\left(\phi_2\right) \\ \rho_v \sin\left(\phi_1\right) \sin\left(\phi_2\right) \cos\left(\phi_3\right) \\ ... \\ \rho_v \sin\left(\phi_1\right) ... \sin\left(\phi_{L-2}\right) \cos\left(\phi_{L-1}\right) \end{array} \right]; \quad \mathbf{u} = \left[\begin{array}{c} \rho_u \cos\left(\phi_1\right) \\ \rho_u \sin\left(\phi_1\right) \cos\left(\phi_2\right) \\ \rho_u \sin\left(\phi_1\right) \sin\left(\phi_2\right) \cos\left(\phi_3\right) \\ ... \\ \rho_u \sin\left(\phi_1\right) ... \sin\left(\phi_{L-2}\right) \cos\left(\phi_{L-1}\right) \end{array} \right].$$

Applying the absolute-value to $\mathbf{v}$ and decomposing, we have:

$$f(\mathbf{v}) = |\rho_v| \begin{bmatrix} |\cos(\phi_1)| \\ |\sin(\phi_1)|\,|\cos(\phi_2)| \\ |\sin(\phi_1)|\,|\sin(\phi_2)|\,|\cos(\phi_3)| \\ ... \\ |\sin(\phi_1)| \ldots |\sin(\phi_{L-2})|\,|\cos(\phi_{L-1})| \end{bmatrix}.$$

Analogously, computing the application of absolute-value to $\mathbf{u}$ we have:

$$f(\mathbf{u}) = |\rho_u| \begin{bmatrix} |\cos(\phi_1)| \\ |\sin(\phi_1)|\,|\cos(\phi_2)| \\ |\sin(\phi_1)|\,|\sin(\phi_2)|\,|\cos(\phi_3)| \\ ... \\ |\sin(\phi_1)| \ldots |\sin(\phi_{L-2})|\,|\cos(\phi_{L-1})| \end{bmatrix}.$$

Now, we can solve the absolute values of the trigonometric functions according to the following formulas:

$$|\cos(\phi)| = \begin{cases} \cos(\phi) & 0 \le \phi < \frac{\pi}{2} \\ \cos(\pi - \phi) & \frac{\pi}{2} \le \phi < \pi \\ \cos(\phi - \pi) & \pi \le \phi < \frac{3}{4}\pi \\ \cos(\pi - \phi) & \frac{3}{4}\pi \le \phi < 2\pi \end{cases} \quad ; \quad |\sin(\phi)| = \begin{cases} \sin(\phi) & 0 \le \phi < \frac{\pi}{2} \\ \sin(\pi - \phi) & \frac{\pi}{2} \le \phi < \pi \\ \sin(\phi - \pi) & \pi \le \phi < \frac{3}{4}\pi \\ \sin(\pi - \phi) & \frac{3}{4}\pi \le \phi < 2\pi \end{cases}.$$

We can therefore remove the absolute value by substituting each angle ϕ_i with the angle ϕ_i^* given by the identities above:

$$f(\mathbf{v}) = |\rho_v| \begin{bmatrix} \cos(\phi_1^*) \\ \sin(\phi_1^*)\cos(\phi_2^*) \\ \sin(\phi_1^*)\sin(\phi_2^*)\cos(\phi_3^*) \\ ... \\ \sin(\phi_1^*) \ldots \sin\left(\phi_{L-2}^*\right)\cos\left(\phi_{L-1}^*\right) \end{bmatrix};$$

$$f(\mathbf{u}) = |\rho_u| \begin{bmatrix} \cos(\phi_1^*) \\ \sin(\phi_1^*)\cos(\phi_2^*) \\ \sin(\phi_1^*)\sin(\phi_2^*)\cos(\phi_3^*) \\ ... \\ \sin(\phi_1^*) \ldots \sin\left(\phi_{L-2}^*\right)\cos\left(\phi_{L-1}^*\right) \end{bmatrix}.$$

But then, again, $f(\mathbf{v})$ and $f(\mathbf{u})$ have the same angular coordinates. Therefore $f(\mathbf{v})$ and $f(\mathbf{u})$ are collinear. ■

A.3 Preservation of Collinearity under Normalization Along the Features

Lemma. Let us consider $\mathbf{v}, \mathbf{u} \in \mathbb{R}^L$, two positive[8] collinear vectors, and let $f : \mathbb{R}^L \to \mathbb{R}^L$ be the ℓ_2-normalization along the features. Then $f(\mathbf{v}), f(\mathbf{u}) \in \mathbb{R}^L$ are also collinear.

Proof. Let us consider the two positive collinear vectors $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & ... & v_L \end{bmatrix}^T$ and $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & ... & u_L \end{bmatrix}^T$ in $\mathbb{R}^L$, where $v_i \geq 0$ and $u_i \geq 0$. Since the two vectors are collinear, they share the same angular coordinates ϕ_i to the origin:

$$\phi_i = \arccos \frac{v_i}{\sqrt{v_L^2 + v_{L-1}^2 + \ldots + v_i^2}} = \arccos \frac{u_i}{\sqrt{u_L^2 + u_{L-1}^2 + \ldots + u_i^2}}.$$

Normalizing along the features means dividing each component i of a vector by a constant k_i equal to the ℓ_2-norm of the component i across all the available vectors. The ℓ_2-normalized vectors are then $f(\mathbf{v}) = \tilde{\mathbf{v}} = \begin{bmatrix} \frac{v_1}{k_1} & \frac{v_2}{k_2} & \frac{v_3}{k_3} & ... & \frac{v_L}{k_L} \end{bmatrix}^T$ and $f(\mathbf{u}) = \tilde{\mathbf{u}} = \begin{bmatrix} \frac{u_1}{k_1} & \frac{u_2}{k_2} & \frac{u_3}{k_3} & ... & \frac{u_L}{k_L} \end{bmatrix}^T$. Now, given that each component is divided by the same constant k_i, the ratios defining the angular coordinate ϕ_i are still equivalent:

$$\begin{aligned} \arccos \frac{\frac{v_i}{k_i}}{\sqrt{\left(\frac{v_L}{k_L}\right)^2 + \left(\frac{v_{L-1}}{k_{L-1}}\right)^2 + \ldots + \left(\frac{v_i}{k_i}\right)^2}} &= \\ = \arccos \frac{\frac{u_i}{k_i}}{\sqrt{\left(\frac{u_L}{k_L}\right)^2 + \left(\frac{u_{L-1}}{k_{L-1}}\right)^2 + \ldots + \left(\frac{u_i}{k_i}\right)^2}} &= \tilde{\phi}_i. \end{aligned}$$

The angular coordinates of $f(\mathbf{v})$ and $f(\mathbf{u})$ are still the same. Therefore $f(\mathbf{v})$ and $f(\mathbf{u})$ are collinear. ■

A.4 Preservation of Collinearity under Normalization Along the Samples

Lemma. Let us consider $\mathbf{v}, \mathbf{u} \in \mathbb{R}^L$, two positive[9] collinear vectors, and let $f : \mathbb{R}^L \to \mathbb{R}^L$ be the ℓ_2-normalization along the samples. Then $f(\mathbf{v}), f(\mathbf{u}) \in \mathbb{R}^L$ are also collinear.

Proof. Let us consider the two positive collinear vectors $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & ... & v_L \end{bmatrix}^T$ and $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & ... & u_L \end{bmatrix}^T$ in $\mathbb{R}^L$, where $v_i \geq 0$ and $u_i \geq 0$. Since the two vectors are collinear, they share the same angular coordinates ϕ_i to the origin:

$$\phi_i = \arccos \frac{v_i}{\sqrt{v_L^2 + v_{L-1}^2 + \ldots + v_i^2}} = \arccos \frac{u_i}{\sqrt{u_L^2 + u_{L-1}^2 + \ldots + u_i^2}}.$$

8. Notice that we can safely make the assumption of positivity in sparse filtering since $\mathbf{v}$ and $\mathbf{u}$ are the output of an absolute-value function.
9. Notice that we can safely make the assumption of positivity in sparse filtering since $\mathbf{v}$ and $\mathbf{u}$ are the output of the normalization along the feature which preserves the positivity.

Normalizing along the samples means dividing each vector by a constant given by the ℓ_2-norm of the vector itself. The ℓ_2-normalized vectors can then be computed as $f(\mathbf{v}) = \hat{\mathbf{v}} = \begin{bmatrix} \frac{v_1}{\ell_2(\mathbf{v})} & \frac{v_2}{\ell_2(\mathbf{v})} & \frac{v_3}{\ell_2(\mathbf{v})} & \cdots & \frac{v_L}{\ell_2(\mathbf{v})} \end{bmatrix}^T$ and $f(\mathbf{u}) = \hat{\mathbf{u}} = \begin{bmatrix} \frac{u_1}{\ell_2(\mathbf{u})} & \frac{u_2}{\ell_2(\mathbf{u})} & \frac{u_3}{\ell_2(\mathbf{u})} & \cdots & \frac{u_L}{\ell_2(\mathbf{u})} \end{bmatrix}^T$.
Now, dividing each component of a vector by the same constant does not affect the value of the angular coordinates ϕ_i :

$$\begin{aligned} \arccos \frac{\frac{v_i}{\ell_2(\mathbf{v})}}{\sqrt{\left(\frac{v_L}{\ell_2(\mathbf{v})}\right)^2 + \left(\frac{v_{L-1}}{\ell_2(\mathbf{v})}\right)^2 + \ldots + \left(\frac{v_i}{\ell_2(\mathbf{v})}\right)^2}} &= \\ = \arccos \frac{\frac{v_i}{\ell_2(\mathbf{v})}}{\sqrt{\frac{1}{(\ell_2(\mathbf{v}))^2}\left(v_L^2 + v_{L-1}^2 + \ldots + v_i^2\right)}} &= \phi_i. \end{aligned}$$

Analogously, this holds for $f(\mathbf{u})$, too. The angular coordinates of $f(\mathbf{v})$ and $f(\mathbf{u})$ are unchanged. Therefore $f(\mathbf{v})$ and $f(\mathbf{u})$ are collinear. ■

A.5 Equivalence of Minimizing ℓ_1-norm and Maximizing $R^{(\mathbf{p}_j)}$

Lemma. Let $\mathbf{x}^{(i)} \in \mathbb{R}^O$ be an original representation and $\mathbf{z}^{(i)} \in \mathbb{R}^L$ be a learned representation computed through sparse filtering, that is $\mathbf{z}^{(i)} = f_{A1:A4}\left(\mathbf{x}^{(i)}\right)$. Let $R^{\mathbf{p}^{(j)}} : \mathbb{R}^O \to \mathbb{R}_{\geq 0}$ the representation-cone function mapping points in the original representation space $\mathbb{R}^O$ to their (Euclidean) distance from the pole $\mathbf{p}^{(j)}$. Then the optimal solution for the minimization of $\sum_{i=1}^N \ell_1\left(\mathbf{z}^{(i)}\right)$, under the constraint $\ell_2\left(\mathbf{z}^{(i)}\right) = 1$, is the same as the optimal solution for $\sum_{i=1}^N \sum_{j=1}^L R^{\mathbf{p}^{(j)}}\left(\mathbf{x}^{(i)}\right)$, under the same constraint.

Proof. Recall that, by definition, the ℓ_1-norm of $\mathbf{z}^{(i)}$ is:

$$\min \sum_{i=1}^N \ell_1\left(\mathbf{z}^{(i)}\right) = \min \sum_{i=1}^N \sum_{j=1}^L \left(\mathbf{z}_j^{(i)}\right).$$

We already know that, given the constraint $\ell_2\left(\mathbf{z}^{(i)}\right) = 1$, the optimal solution for this optimization problem is given by the set of poles $\mathbf{p}^{(j)}$ in $\mathbb{R}^L$ (Ngiam et al., 2011).
Now, let us consider the minimization of the representation-cone function and let us make explicit its argument:

$$\begin{aligned} \min \sum_{i=1}^N \sum_{j=1}^L R^{\mathbf{p}^{(j)}}\left(\mathbf{x}^{(i)}\right) &= \min \sum_{i=1}^N \sum_{j=1}^L \left\|\mathbf{p}^{(j)} - f_{A1:A4}\left(\mathbf{x}^{(i)}\right)\right\|_{L2} \\ &= \min \sum_{i=1}^N \sum_{j=1}^L \left\|\mathbf{p}^{(j)} - \mathbf{z}^{(i)}\right\|_{L2}. \end{aligned}$$

Given the constraint $\ell_2\left(\mathbf{z}^{(i)}\right) = 1$, the distances between a representation $\mathbf{z}^{(i)}$ and the poles $\mathbf{p}^{(j)}$ are minimized when the representations $\mathbf{z}^{(i)}$ are identified with the poles. If a representation $\mathbf{z}^{(i)}$ lies anywhere else on the surface of the hyper-sphere defined by the constraint, then, by the triangle inequality, the distance between $\mathbf{z}^{(i)}$ and the poles $\mathbf{p}^{(j)}$ is greater. Therefore, both the optimization problems have an optimal solution in having the representations $\mathbf{z}^{(i)}$ mapped to the poles $\mathbf{p}^{(j)}$. ■

A.6 Limit of the Ratio of Gamma Functions

Lemma. Let us consider $x \in \mathbb{R}$. Then $\lim_{x\to\infty} \frac{\Gamma(x)}{\Gamma(x+\frac{1}{2})} = \frac{1}{\sqrt{x}}$.

Proof. This statement follows from an application of the Stirling's formula to the gamma function. Given the following limit proven in Equation 2.3.19 in Freeden and Gutting (2013):

$$\lim_{x\to\infty} \frac{\Gamma(x+a)}{x^a \Gamma(x)} = 1,$$

let us substitute a with $\frac{1}{2}$:

$$\lim_{x\to\infty} \frac{\Gamma(x+\frac{1}{2})}{x^{\frac{1}{2}} \Gamma(x)} = 1.$$

Reordering:

$$\lim_{x\to\infty} \frac{\Gamma(x+\frac{1}{2})}{\sqrt{x}\Gamma(x)} = 1$$

$$\lim_{x\to\infty} \frac{\Gamma(x)}{\Gamma\left(x+\frac{1}{2}\right)} = \frac{1}{\sqrt{x}},$$

which is the formula we wanted to prove. ■

References

Baktash Babadi and Haim Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213–1226, 2014.

Keith Ball. An elementary introduction to modern convex geometry. *Flavors of geometry*, 31:1–58, 1997.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

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

Christopher M. Bishop. *Pattern recognition and machine learning.* Springer, 2007.

Neil D.B. Bruce, Shafin Rahman, and Diana Carrier. Sparse coding in early visual representation: from specific properties to general principles. *Neurocomputing*, 171:1085–1098, 2016.

Felix Burkhardt, Astrid Paeschke, Miriam Rolfes, Walter F. Sendlmeier, and Benjamin Weiss. A database of German emotional speech. In *Interspeech*, volume 5, pages 1517–1520, 2005.

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.

Matteo Carandini and David J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62, 2012.

Donald G. Childers, David P. Skinner, and Robert C. Kemerait. The cepstrum: a guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, 1977.

Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random structures and algorithms*, 22(1):60–65, 2003.

Misha Denil and Nando de Freitas. Recklessly approximate sparse coding. *arXiv preprint arXiv:1208.0959*, 2012.

Zhen Dong, Mingtao Pei, Yang He, Ting Liu, Yanmei Dong, and Yunde Jia. Vehicle type classification using unsupervised convolutional neural network. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 172–177. IEEE, 2014.

Zhen Dong, Yuwei Wu, Mingtao Pei, and Yunde Jia. Vehicle type classification using a semisupervised convolutional neural network. *Intelligent Transportation Systems, IEEE Transactions on*, 16(4):2247–2256, 2015.

Peter Földiák and Malcom P. Young. Sparse coding in the primate cortex. *The handbook of brain theory and neural networks*, 1:1064–1068, 1995.

Willi Freeden and Martin Gutting. *Special functions of mathematical (geo-) physics*. Springer Science & Business Media, 2013.

Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience*, 35: 485–508, 2012.

Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Unsupervised and supervised visual codes with restricted Boltzmann machines. In *Computer Vision–ECCV 2012*, pages 298–311. Springer, 2012.

Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Learning deep hierarchical visual feature coding. *Neural Networks and Learning Systems, IEEE Transactions on*, 25 (12):2212–2225, 2014.

Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: a report on three machine learning contests. In *Neural information processing*, pages 117–124. Springer, 2013.

Zhongyi Gu, Lin Zhang, Xiaoxu Liu, Hongyu Li, and Jianwei Lu. Learning quality-aware filters for no-reference image quality assessment. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.

William Edward Hahn, Stephanie Lewkowitz, Daniel C. Lacombe Jr, and Elan Barenholtz. Deep learning human actions from video via sparse filtering and locally competitive algorithms. *Multimedia Tools and Applications*, pages 1–14, 2015.

Yoonchang Han and Kyogu Lee. Hierarchical approach to detect common mistakes of beginner flute players. In *ISMIR*, pages 77–82, 2014.

Yoonchang Han and Kyogu Lee. Detecting fingering of overblown flute sound using sparse feature learning. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016(1):2, 2016.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Niall Hurley and Scott Rickard. Comparing measures of sparsity. *Information Theory, IEEE Transactions on*, 55(10):4723–4741, 2009.

William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Andrei S. Kozlov and Timothy Q. Gentner. Central auditory neurons have composite receptive fields. *Proceedings of the National Academy of Sciences*, page 201506903, 2016.

Johannes Lederer and Sergio Guadarrama. Compute less to get more: using ORC to improve sparse filtering. *arXiv preprint arXiv:1409.4689*, 2014.

Yaguo Lei, Feng Jia, Jing Lin, Saibo Xing, and Steven Ding. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *Industrial Electronics, IEEE Transactions on*, PP:1, 2015.

Ralph Linsker. An application of the principle of maximum information preservation to linear systems. In *Advances in neural information processing systems*, pages 186–194, 1989.

David J.C. MacKay. *Information theory, inference, and learning algorithms*, volume 7. Cambridge University Press, 2003.

Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, Spain, 2011.

Jiquan Ngiam, Zhenghao Chen, Sonia A. Bhaskar, Pang W. Koh, and Andrew Y. Ng. Sparse filtering. In *Advances in Neural Information Processing Systems*, pages 1125–1133, 2011.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

Kiran B. Raja, R. Raghavendra, Vinay Krishna Vemuri, and Christoph Busch. Smartphone based visible iris recognition using deep sparse filtering. *Pattern Recognition Letters*, 57: 33–42, 2015.

Marc'Aurelio Ranzato, Cristopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *Proceedings of Neural Information Processing Systems*, 2006.

Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Proceedings of Neural Information Processing Systems*, 2007.

Edmund T. Rolls and Alessandro Treves. The relative advantages of sparse versus distributed encoding for associative neuronal networks in the brain. *Network: computation in neural systems*, 1(4):407–421, 1990.

Lukasz Romaszko. A deep learning approach with an ensemble-based neural network classifier for black box ICML 2013 contest. In *Workshop on Challenges in Representation Learning, ICML*, 2013.

Adriana Romero, Petia Radeva, and Carlo Gatta. No more meta-parameter tuning in unsupervised sparse feature learning. *arXiv preprint arXiv:1402.5766*, 2014.

Shaun K. Ryman, Neil D.B. Bruce, and Michael S. Freund. Temporal responses of chemically diverse sensor arrays for machine olfaction using artificial intelligence. *Sensors and Actuators B: Chemical*, 2016.

Björn Schuller, Anton Batliner, Stefan Steidl, and Dino Seppi. Recognising realistic emotions and affect in speech: state of the art and lessons learnt from the first challenge. *Speech Communication*, 53(9):1062–1087, 2011.

Dandan Si, Yuanyuan Hu, Zongliang Gan, Ziguan Cui, and Feng Liu. Edge directed single image super resolution through the learning based gradient regression estimation. In *Image and Graphics*, pages 226–239. Springer, 2015.

Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition - tangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274. Springer, 1998.

Masashi Sugiyama and Motoaki Kawanabe. *Machine learning in non-stationary environments: introduction to covariate shift adaptation*. MIT Press, 2012.

Hao Sun, Huanxin Zou, and Shilin Zhou. Accumulating pyramid spatial-spectral collaborative coding divergence for hyperspectral anomaly detection. In *2015 ISPRS International Conference on Computer Vision in Remote Sensing*, pages 99010J–99010J. International Society for Optics and Photonics, 2016.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

Dongrui Wu, Thomas D. Parsons, and Shrikanth S. Narayanan. Acoustic feature analysis in speech emotion primitives estimation. In *INTERSPEECH*, pages 785–788, 2010.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, pages 521–528, 2003.

Chao Yan, Frans Coenen, and Bailing Zhang. Driving posture recognition by convolutional neural networks. *IET Computer Vision*, 2015.

Zhao Yang, Lianwen Jin, Dapeng Tao, Shuye Zhang, and Xin Zhang. Single-layer unsupervised feature learning with l2 regularized sparse filtering. In *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit & International Conference on*, pages 475–479. IEEE, 2014.

Shaohua Zhang, Hua Yang, and Zhouping Yin. Performance evaluation of typical unsupervised feature learning algorithms for visual object recognition. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pages 5191–5196. IEEE, 2014.

Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. A survey of sparse representation: algorithms and applications. *Access, IEEE*, 3:490–530, 2015.