Universidad de Sevilla

Facultad de Física
Departamento de electrónica y electromagnetismo

Inference on compressive measurements

Bachelor's Thesis

Author:
Marina Jiménez Cómez

Supervisors:
Jorge Fernández Berni
Riu Rodríguez Sakamoto

September 2021

Abstract

Compressed sensing is a new paradigm capable of sampling and compressing signals in one step. Its original purpose was to compress sparse or compressible signals in a way that reconstruction from the measurements taken were possible. However, many applications do not require signal recovery. Therefore, a new branch of compressed sensing aims to directly perform inference using the information encoded in the samples, instead of recovering the complete signal and then solving inference problems over the reconstructed signal. In this thesis, the mathematical framework for detection, classification, estimation and filtering on compressed measurements was studied. Moreover, applications of inference mainly based on machine learning as an implementation tool were reviewed. Finally, machine-learning algorithms were tested on compressed measurements.

Keywords— Compressed Sensing, Inference, Detection, Classification, Estimation, Filtering, Compressed Learning, Machine Learning, Deep Neural Networks

Resumen

El muestreo compresivo es un nuevo paradigma capaz de muestrear y comprimir señales en un solo paso. Su objetivo original era el de comprimir señales sparse o compresibles de tal modo que su reconstrucción a partir de las medidas tomadas fuera posible. Sin embargo, muchas aplicaciones no requieren recuperar la señal. Por lo tanto, una nueva rama del muestreo compresivo pretende realizar problemas de inferencia directamente usando la información codificada en las muestras en lugar de recuperar completamente la señal y posteriormente resolver el problema de inferencia sobre la señal reconstruida. En este trabajo se estudia el marco matemático de los problemas de detección, clasificación, estimación y filtrado. Además, se revisan aplicaciones de inferencia principalmente basadas en el uso de herramientas de aprendizaje automático. Por último, se prueban algoritmos de aprendizaje automático sobre medidas comprimidas.

Acknowledgements

This project is dedicated to my parents and grandparents. They have supported me throughout the years and encourage me to pursue my goals. I would also like to thank my supervisors Jorge Fernández Berni and Riu Rodríguez Sakamoto for their guidance and patience.

Contents

List of Figures

List of Tables

Acronyms

CL Compressed Learning. 49, 50, 62

CNN Convolutional Neural Network. 50–52

CoSaMP Compressive Sampling Matching Pursuits. ix, 14, 16

CS Compressed Sensing. 2–4, 6, 11, 13, 16, 23–25, 30, 43, 49, 53, 61, 62

DCT Discrete Cosine Transform. ix, 2, 16–18, 20

DMD Digital Micromirror Device. 24

DNN Deep Neural Network. 48, 49, 52, 53, 58, 59, 62, 63

ECG Electrocardiogram. 54

FC Fully Connected. 48

IHT Iterated Hard Thresholding. 10

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 48, 52

MRI Magnetic Resonance Imaging. 24

NN Neural Network. 47, 48, 62

OMP Orthogonal Matching Pursuit. ix, 10, 13, 14

ReLU Rectified Nonlinear Unit. 58

RIP Restricted Isometry Property. 7–9, 44

ROC Receiver Operating Characteristic. 35

SNR Signal to Noise Ratio. 23, 35, 36, 38, 44, 45

SPC Single Pixel-Camera. 4, 23, 50, 51

SVM Supported Vector Machine. 49, 50, 56, 57, 62

Chapter 1

Introduction

Paper documents, vinyls, photo negatives, physical books... Some of these objects are already in the past and the others could soon be too. Physical information support is disappearing due to digitization. Moreover, we are creating more and more digital contents every second through smartphones, PC, GPS, sensors... They are examples of devices that surround us everyday and generate data such as images or videos. To have an idea of the rapid increase in the volume of information, according to International Data Corporation (IDC), in 2006 there were 161 exabytes of digital data and by 2020 the estimate was 59 zettabytes. Taking into account that 1 zettabyte equals 1000 exabytes, the increase in data to manage is enormous.

All this information brings about challenges related to data management, storage and analysis. Proper data representation is needed, redundancy must be reduced and compression has to be applied. Another challenge is energy consumption. Processing, storage and transmission of massive data consume a lot of electrical energy. In this context, signal processing plays a major role to improve the situation with the development of new theories able to simplify the work with signals. In particular, a focus of this thesis is on how to process data acquired by sensors.

In classical signal processing, for transmission or storage purposes, a signal is first sampled. In this process n elements are obtained. Subsequently, compression is applied, which means that from these n elements, just $k \leq n$ are taken and the rest are discarded. Then, this compressed signal can be either stored in a storage device or in the cloud, or it can be transmitted. In this second case, the k elements are sent to the receiver, where a decompression or reconstruction process is conducted.

Regarding sampling, the Nyquist-Shannon theorem [56] sets how to sample a band-limited time-varying signal in order to be able to reconstruct it from the corresponding measurements. In other words, this theorem determines how to keep the relevant information of the signal, which must be sampled uniformly at a rate that exceeds twice the signal's highest frequency. Although this result has been widely used, it presents some drawbacks. As it has been mentioned before, after n measurements are taken, they are compressed into k samples for storage or transmission purposes. Consequently, $n - k$ samples are discarded, which means it was not necessary to acquire them in the first place. Additionally, for signals containing high frequencies, the sampling rate required by Nyquist criterion might be too high to be practically implemented [64]. Furthermore, the theorem is only applicable to band-limited signals whose band-limit is known a priori.

The aim of compression is to reduce redundancy and irrelevancy [36]. For example, neighbouring pixels in an image are correlated, so there is intrinsic redundancy. However, there can be some loss in the process of compression. In fact, well-known compression algorithms, such as JPEG for images, are lossy because when compression is performed, too much information is discarded, thereby causing degradation in the reconstruction with respect to the original data. There are also lossless compression methods, for example the algorithm related to the PNG format [8].

One of the most important techniques for compression is transform coding. Different transforms provide representations of the original data in new domains where the number of coefficients that gather the information is smaller than in the original domain. For example, JPEG is based on the Discrete Cosine Transform (DCT) [50], which is an orthogonal transform that represents the image in the frequency domain.

Based on the idea of transform coding, and aiming to improve Nyquist-Shannon's rate, Compressed Sensing (CS) appeared in 2006 introduced by Donoho, Candès, Romberg, and Tao [28, 11]. In [49] and [51], a theoretical review of CS is presented together with some of its applications.

The aim of CS is to provide a theoretical framework in which the reconstruction of a signal can be obtained from fewer measurements than using the Nyquist-Shannon rate. This theory works with sparse or compressible representations of signals. This means that, in a certain basis, the targeted signals are represented by a small set of coefficients or by a set of coefficients that, if sorted, exhibit rapidly decaying values. Provided that the signals of interest present this characteristic, fewer measurements than those resulting from applying the Nyquist criterion can be taken randomly while keeping all relevant information. The fact that the acquisition and compression of measurements are carried out concurrently gave its name to the theory: Compressed Sensing. Once simultaneous sampling and compression is performed, the compressed signal would be transmitted to the receiver for reconstruction.

The fundamental idea of CS is solving an underdetermined linear system to implement the recovery:

$$y = \Phi f \tag{1.1}$$

In this expression, y is the compressed signal, Φ is a matrix that represents how the signal was sampled, and f is the original signal that we need to reconstruct. To solve this, an optimization problem is set that includes restrictions over the sparsity of the signal recovered. Many different approaches have been followed to develop algorithms capable of reconstructing a signal under the CS framework. Some of them are reviewed in this thesis.

However, signal recovery is computationally expensive. Moreover, for many signal processing problems, recovery is not always needed. Would it be possible to work just with the compressed measurements? Let us focus on computer vision for example. Pattern recognition, detection and classification are three major problems addressed by this field. For any of them, the same steps are typically followed. For instance, if we wished to classify an image, first we would need to acquire the visual information using a CS sensor. The information with the compressed measurements would be sent to the device that is going to analyze the data. In this device, the image is reconstructed and then the classification is performed. However, knowing that the aforementioned problems are solved by extracting the most important features from the original raw data and that with CS such features are included in the measurements, the question arising is *would it be possible to avoid reconstruction and directly conduct classification on the compressed samples?* How can we achieve this? A mathematical framework to solve inference problems on compressive measurements is needed.

Statistical inference is a mathematical field that provides methods to obtain relevant information about a statistical population with just partial information about that population. Using techniques provided by statistics we could infer information from compressed measurements. In fact, [22] provides mathematical bounds within which four inference problems are solved when working with compressed measurements. These problems are detection, classification, estimation and filtering. Two key points are that measurements are taken randomly, as mentioned before, and that for inference there is no need to know in which basis the signal is sparse or compressible.

Once this framework was developed, many approaches for achieving CS-based direct inference used neural networks. Machine learning is a branch of artificial intelligence that involves inference and database processing. Therefore, it seems natural to try to join compressed sensing and neural networks. The idea is to have a sensing device in which the compressive process is performed during the acquisition of the signal, obtaining compressed samples, and then inference is implemented using machine learning. Usually the samples are said to be in a low dimension called *compressed domain* while the original signal is in a high dimension called *data domain.* The aim is to perform inference with similar accuracy in both dimensions. Thus, we would be achieving a dimensionality reduction while preserving the most relevant information included in the original data.

Devices that directly acquire signals in a compressed form according to the requirements set by CS have already been implemented. A well-known example is the Single Pixel-Camera (SPC) [30]. Current research is focused on applying inference to the measurements taken with this type of devices.

The main objective of this thesis was to review state-of-the-art research about inference on compressed measurements taken according to the CS theory and provide its mathematical background. The thesis is organised as follows. Chapter 2 explains CS theory, providing its mathematical framework and describing algorithms developed for reconstruction, which was the original purpose of this theory; in this context, some of the most important applications of CS are also reviewed. In Chapter 3, we introduce statistical inference concepts and explain detection, classification, estimation and filtering with compressed measurements. Inference applications are reviewed in Chapter 4 and we also show how inference can be implemented using Python code. Finally, some concluding remarks are provided in Chapter 5. The complete code developed throughout this thesis is described in the Appendix.

Chapter 2

Compressed sensing

2.1 Background

Suppose a signal which is intended to be sampled so that the most important information is preserved. Nyquist-Shannon theorem [56] shows how to sample a band-limited signal to be able to reconstruct the original information from the measurements taken. To present the theorem, some definitions are needed.

Definition 1 (Fourier transform)**.** *Let $f \in L^1(\mathbb{R})$ be a continuous time function. Its Fourier transform is defined as*

$$\hat{f}(q) = \int_{\mathbb{R}} f(t) e^{-2\pi i t q} dt \tag{2.1}$$

where $q \in \mathbb{R}$.

Definition 2 (Band-limit signal)**.** *Let $f \in L^1(\mathbb{R})$ be a continuous time function and $\hat{f}$ its Fourier transform. It is said that f is band-limited with bandwidth B if $\hat{f} \subset [-B, B]$.*

The Nyquist-Shannon theorem states that band-limited signals must be sampled at a rate twice its bandwidth. If this condition is satisfied, the signal can be completely reconstructed from the samples.

Theorem 1 (Nyquist-Shannon [31])**.** *Let $f \in L^1(\mathbb{R})$ be a band-limited continuous time function with bandwidth B. Then it occurs that*

$$f(t) = \sum_{k \in \mathbb{Z}} f\left(\frac{k}{2B}\right) sinc(2\pi B t - \pi k) \tag{2.2}$$

where

$$sinc(t) = \begin{cases} \frac{\sin(t)}{t} & t \neq 0 \\ 1 & t = 0 \end{cases} \tag{2.3}$$

To apply this theorem, first the signal needs to be band-limited and second the value of this limit must be known in advance, which may not always be possible. Also, we have to take into account that the frequency along the signal varies. Therefore, in fragments where the frequency is low we could be taking more measurements than truly needed.

The Nyquist-Shannon theorem shows how to sample a signal in order to reconstruct it from the measurements taken. However, we want to do better and sample below the Nyquist rate. With this purpose, the theory called Compressed Sensing (CS) has being developed during the last few years. It provides certain hypotheses regarding sparsity of the signal so that all the relevant information is preserved in compressed measurements. Different algorithms have been developed to implement compressed sensing.

For mathematical treatment, signals are represented as elements of a Hilbert space H. We are going to take $H = \mathbb{R}^N$. Therefore, $x \in \mathbb{R}^N$ is a signal. To model the extraction of $M \ll N$ measurements we use a sensing matrix $\Phi \in \mathcal{M}_{M\times N}(\mathbb{R})$ and let $y \in \mathbb{R}^M$ be the measurement vector. Finally, the mathematical expression of the problem addressed by CS is to obtain x knowing that $y = \Phi x$.

2.2 Theoretical framework

In this section, we introduce definitions and mathematical results of CS [23].

Definition 3 (Sparse signal)**.** *A vector $\alpha \in \mathbb{R}^N$ is said to be sparse when the number of non-zero coefficients that represents is small. It is said to be S-sparse if S is the number of non-zero coefficients: $S = |\{i \in \{1, 2, .., N\} : \alpha_i \neq 0\}|$.*

Definition 4 (Set of sparse signals)**.** *The set of all S-sparse vectors is defined as*

$$\Sigma_S = \{\alpha \in \mathbb{R}^N : \|\alpha\|_0 := |supp(\alpha)| \leq S\}$$

where $supp(\alpha) = \{i \in \{1, 2, .., N\} : \alpha_i \neq 0\}$.

There are also signals that, even though their coefficients are mostly non-zero, they decay rapidly when they are sorted from high to low values. This means that the information is encoded in the coefficients with higher values.

Definition 5 (Compressible signal)**.** *$\alpha \in \mathbb{R}^N$ is compressible if the sorted coefficients $|\alpha_1| \geq |\alpha_2| \geq ... \geq |\alpha_N|$ obey*

$$|\alpha_i| \leq Ci^{-q}$$

where C is constant, $i \in 1, 2, ...N$ and q is the parameter that controls the decay.

The signals of interest for the CS theory are either sparse or compressible. Now let us suppose that the signal we are working with is neither of them. In this case, it must be transformed somehow in order to obtain a valid representation. The solution to this problem implies finding a basis of the space in which the signal is sparse or compressible.

Definition 6 (Basis)**.** *The set $\Psi = \{\psi_i\}_{i \in \mathcal{I}}$ with $\mathcal{I} = \{1, 2, ...N\}$ is a basis of $\mathbb{R}^N$ if ψ_i are linearly independent $\forall i \in \mathcal{I}$.*

Ψ can be considered as a matrix $\mathcal{M}_{N\times N}(\mathbb{R})$, called a transformation matrix, whose columns are ψ_i. The representation of the signal $x \in \mathbb{R}^N$ in a basis $\{\psi_i\}$ is done as a linear combination of the columns of Ψ.

$$x = \sum_{i=1}^{N} \alpha_i \psi_i = \Psi\alpha$$

where $\alpha_i = \langle x, \psi_i \rangle$ and $\alpha \in \mathbb{R}^N$.

Once a sparse representation of the targeted signal is found, we can extract measurements from it with the sampling matrix $\Phi \in \mathcal{M}_{M\times N}(\mathbb{R})$ with $M \ll N$. The product of both sensing and transformation matrices, $\Theta = \Phi\Psi$, is usually denoted as the recovery matrix. In order to ensure that we are keeping all the important information and that reconstruction is possible, the recovery matrix must have certain properties. In the first place, we introduce the Restricted Isometry Property (RIP).

Definition 7 (Restricted Isometry Property)**.** *Let $\Psi \in \mathcal{M}_{N\times N}(\mathbb{R})$ be a basis and let $\Phi \in \mathcal{M}_{M\times N}(\mathbb{R})$ be a sensing matrix with $M \ll N$. It is said that RIP of order S is satisfied if there exists $\delta_S \in (0, 1)$ with*

$$(1 - \delta_S) \|\alpha\|_2^2 \leq \|(\Phi\Psi)\alpha\|_2^2 \leq (1 + \delta_S) \|\alpha\|_2^2 \tag{2.4}$$

for all S-sparse vector $\alpha \in \Sigma_S$.

If the RIP of order $2S$ is satisfied, it follows that for $\alpha_1, \alpha_2 \in \Sigma_S$ with $\alpha_1 \neq \alpha_2$ then

$$\|\alpha_1 - \alpha_2\| \approx \|\Phi\Psi\alpha_1 - \Phi\Psi\alpha_2\|_2$$

and therefore, $\Phi\Psi$ is an approximate isometry for S-sparse vectors, [12]. Moreover, all subsets of $2S$ columns of $\Phi\Psi$ are nearly orthogonal, and S-sparse signals are not in the null space of the matrix. Note also that $\Phi\Psi\alpha_1 \neq \Phi\Psi\alpha_2$ if $\alpha_1 \neq \alpha_2$.

Sensing matrices are usually designed before extracting measurements. During their generation, it must be ensured that they satisfy the properties needed.

Definition 8 (Random matrix)**.** *Let A be a matrix. A is said to be a random matrix if its entries are random independent variables identically distributed to a Bernoulli, Gaussian or sub-Gaussian distribution (in the two last cases, with zero mean, $\mu = 0$, and unit variance, $\sigma^2 = 1$).*

In fact, Bernoulli and Gaussian matrices are also sub-Gaussian matrices. We can consider A as a random matrix, and $\Phi = \frac{1}{\sqrt{M}}A$. These matrices are commonly used because, with high probability, they verify the RIP. Moreover, it is fulfilled that $\mathbb{E}[\|\frac{1}{\sqrt{M}}Ax\|_2^2] = \|x\|_2^2$ for any $x \in \mathbb{R}^N$ and A is a sub-Gaussian matrix [31].

The selection of the sensing matrix is important because the value of M that can be achieved depends on it. For example, for Gaussian or Bernoulli matrices, we would have $M \geq CS\log(N/S)$. The proof can be found in [31].

Definition 9 (Coherence)**.** *The coherence μ of matrices Φ and Ψ is defined as*

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{1 \leq i \leq M, 1 \leq k \leq N} |\langle \phi_i, \psi_k \rangle| \tag{2.5}$$

It measures the maximum correlation between elements of the two different matrices. We have $\mu(\Phi, \Psi) \in \left[1, \sqrt{N}\right]$. The goal in CS is to minimise this correlation. Consequently, we are maximizing incoherence [12].

Finally, we define our problem mathematically. Suppose that $x \in \mathbb{R}^N$ is the signal of interest, Ψ is the sparsifying basis in which $x = \Psi\alpha$ with $\alpha \in \mathbb{R}^N$ is a S-sparse vector, Φ is the sensing matrix and $y \in \mathbb{R}^M$ is the measurement vector with $M \ll N$. The recovery of α from the samples in y consists in solving $y = \Phi\Psi\alpha$. This is an underdetermined system so it has infinite solutions. However, by formulating the reconstruction problem as an optimization problem that minimises for the sparsest solution, that is, where α is S-sparse with minimum S, we can achieve a unique solution. Taking into account the sparsity property, the first attempt to write the convex problem would be:

$$\min_{\boldsymbol{\alpha}} \|\alpha\|_0 \quad \text{s.t.} \quad y = \Phi\Psi\alpha \tag{2.6}$$

However, this is an NP problem whose implementation demands an enormous amount of computational load. The proof is provided in [31]. We could have l_2-minimization, but it does not return sparse signals. Therefore, l_1 norm is used:

$$\min_{\alpha} \|\alpha\|_1 \quad \text{s.t.} \quad y = \Phi\Psi\alpha \tag{2.7}$$

Finally, if we consider that measurements can contain error, then the following theorem provides the conditions for which reconstruction is possible:

Theorem 2 (Candès [12]). *Suppose $\Phi\Psi$ satisfies the RIP of order 2S with $\delta_{2S} < \sqrt{2} - 1$. Suppose also that the measurement vector has the form $y = \Phi\Psi\alpha + e$ with $\|e\|_2 < \epsilon$. Then the solution*

$$\hat{\alpha} = \min_{\alpha} \|\alpha\|_1 \quad s.t. \quad \|\Phi\Psi\alpha - y\|_2 \leq \epsilon \tag{2.8}$$

obeys

$$\|\hat{\alpha} - \alpha\|_2 \leq C_0\epsilon + C_1\frac{\|\alpha - \alpha_S\|_1}{\sqrt{S}} \tag{2.9}$$

where α_S is the vector α with all but its largest S coefficients set to zero. C_0 and C_1 are constants. If α is S-sparse, then the recovery is exact.

2.3 Reconstruction algorithms

According to [51] and [49], six categories of algorithms can be distinguished depending on the approach used for reconstruction.

1. `Convex optimization`

   A convex optimization problem is solved through linear programming. In general, this type of algorithms are computationally expensive and the reconstruction time is high.

An example is the Basis Pursuit (BP) algorithm [15]. The convex problem it solves is

$$\hat{\alpha} = \min_{\alpha} \|\alpha\|_1 \quad \text{s.t.} \quad y = \Phi\Psi\alpha \tag{2.10}$$

This is functional when the measurement vector has no noise. Otherwise, a suitable algorithm is Basis Pursuit De-Noising (BPDN) [15], which solves the problem defined in Theorem 2:

$$\hat{\alpha} = \min_{\alpha} \|\alpha\|_1 \quad \text{s.t.} \quad \|\Phi\Psi\alpha - y\|_2 \leq \epsilon \tag{2.11}$$

2. `Greedy algorithms`

   This category is formed by iterative algorithms that obtains a better approximation of the solution in every iteration by selecting only columns of $\Phi\Psi$ that are highly correlated with the measurements. Every algorithm has its own stopping condition. The most used ones are Matching Pursuit (MP) [46] and Orthogonal Matching Pursuit (OMP) [48]. However, when the signal is not sparse, the computational cost becomes too high. To solve this, modifications to their structure were proposed, giving rise to other algorithms such as Compressive Sampling Matching Pursuits (CoSaMP) [47].

3. `Iterative thresholding algorithms`

   This class consists of iterative algorithms that use a threshold in every update of the solution. The most popular algorithms are Iterative Hard-Thresholding (IHT) [6] and Iterative Soft-Thresholding (IST) [21]. The iterator used in both cases is

$$\alpha_{k+1} = \mathbf{T}\left[\alpha_k + \lambda(\Phi\Psi)^T (y - \Phi\Psi\alpha_k)\right] \tag{2.12}$$

where λ is the step size and $\mathbf{T}()$ is the thresholding operator. Given a positive scalar T, the Hard-Thresholding operator is defined as:

$$\mathbf{T}(x)_i = \begin{cases} x_i & |x_i| \geq T \\ 0 & |x_i| < T \end{cases} \tag{2.13}$$

Concerning the Soft-Thresholding operator, it is given by:

$$\mathbf{T}_T(x)_i = (|x_i| - T)_+ \operatorname{sign}(x_i) \tag{2.14}$$

4. `Combinatorial algorithms`

   These algorithms pre-date CS literature but have been adapted for sparse recovery [17]. Representative algorithms are Heavy Hitters on Steroids (HHS) [33] or Chaining Pursuits [32].

5. `Non-convex minimization algorithms`

   This approach uses the ℓ_p norm instead of the ℓ_1 norm in Eq. (2.7). It is used with $p \in (0, 1)$; however, note that when $0 \leq p \leq 1$, $\|\cdot\|_p$ is not strictly a norm. With this change, the minimization problem is no longer convex. However, it is still possible to achieve sparse recovery with few measurements [14], [13]. As an example, we can mention Focal Underdetermined System Solution (FOCUSS) [65].

6. `Bregman iterative algorithms`

   This type of algorithms solve the constrained minimization problem by solving a series of unconstrained problems generated by a Bregman iterative regularization scheme [27]. The unconstrained problems have the form

$$\hat{\alpha} = \min_{\alpha}\{\mu\|\alpha\|_1 + \frac{1}{2}\|\Phi\Psi\alpha - y\|_2^2\} \tag{2.15}$$

   where $\mu \geq 0$.

2.4 Practical Examples

Based on [60], we next present CS concepts using Python code. First, we will represent a 1D sparse signal and apply some greedy reconstruction algorithms. Then, in order to show how a basis transform works, we will employ a non-sparse 1D signal and apply convex reconstruction. Finally, we will show an example in which an image is reconstructed.

In all cases we provide the reconstruction error defined as

$$\varepsilon = \frac{\text{RMSE}}{\text{Dinamic range}} \tag{2.16}$$

The dynamic range of a signal is the range of values that its coefficients take. This is, the maximum value minus the minimum one. RMSE stands for Root Mean Square Error and is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x_i^*)^2} \tag{2.17}$$

where x is the original signal and x^* is the reconstructed one.

2.4.1 Sparsity and reconstruction

We first plot an example of sparse signal obtained randomly. The libraries employed in this section are Numpy and Matplotlib.

```
import numpy as np
import matplotlib.pyplot as plt
```

A sparse signal can be created with the following commands. It generates S random values according to a standard normal distribution and puts them in random positions. In this example, the length of the signal is $N = 1000$ and the sparsity is $S = 10$.

```
#Generate S-Sparse 1D signal of length N
N=1000 #Length
S=10 #Sparsity
x = np.zeros((N,1))
x[0:S,:] = np.random.randn(S,1)
np.random.shuffle(x)
```

We obtain the signal shown in Fig. 2.1.

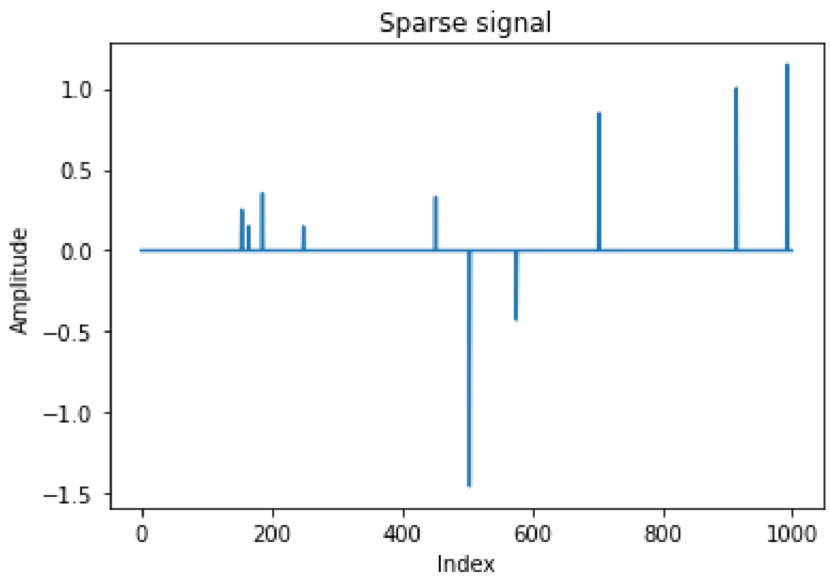

Figure 2.1: 1D sparse signal with sparsity S=10.

This signal is already sparse. Therefore, we do not have to apply any further transformation. In the theoretical framework described previously in the present chapter, this case corresponds to $\Psi = I$, where I represents the identity matrix of the corresponding shape, which in this case is 1000×1000.

The next step in CS is sampling. First, we set the number of measurements. In our example, this is $M = 200$. This means that our compression ratio is $M/N = 0.20$. Therefore, we are discarding 80% of the information in our original signal. Our sensing matrix Φ is a 200×1000 normal random matrix according to Definition 8.

```
# Random sensing matrix
# 200 measurements taken
M = 200
Phi = np.random.randn(M,N) / np.sqrt(M)
print ("Compression ratio {0}".format(M/N)) # M/N=0.2
y = Phi @ x # Measurement vector
```

Once the samples are taken, the next step is to implement an algorithm to recover the signal. We present two examples of implementations of greedy algorithms based on the code reported in [54].

Example 1 (OMP)**.** As mentioned in the previous section, OMP is a greedy algorithm that is implemented as follows for a sparse signal:

1. Initialization:

   Inputs: sensing matrix Φ, measurement vector y, sparsity S
   Initial values: $r = y$, $\Omega_k = \emptyset$, $k = 0$

2. k-th iteration with $k \in \{1, ..., S\}$:

   (a) Select the index of the most correlated column of Φ with r: $\lambda_k = \arg\max_j |\langle \Phi_j, r_k \rangle|$

   (b) Supports are joined: $\Omega_k = \Omega_{k-1} \cup \{\lambda_k\}$

   (c) Solution in the k-th iteration: $x_k = (\Phi_{\Omega_k}^T \Phi_{\Omega_k})^{-1} \Phi_{\Omega_k}^T y$

   (d) Update residual: $r_{k+1} = y - \Phi_{\Omega_k} x_k$

We can directly implement this pseudo-code in Python.

```
def omp(A,y,S):

    x_k = np.zeros_like(x)
    r_k = np.copy(y)
    Omega_k = []
    for k in range(S):
        lambda_k = np.argmax(np.abs(A.T @ r_k))
        Omega_k.append(lambda_k)
        x_k[Omega_k,:] = np.linalg.pinv(A[:,Omega_k]) @ y
        r_k = y - A @ x_k

    return x_k

omp_recovery = omp(Phi,y,S)
```

The outcome is shown in Fig. 2.2:

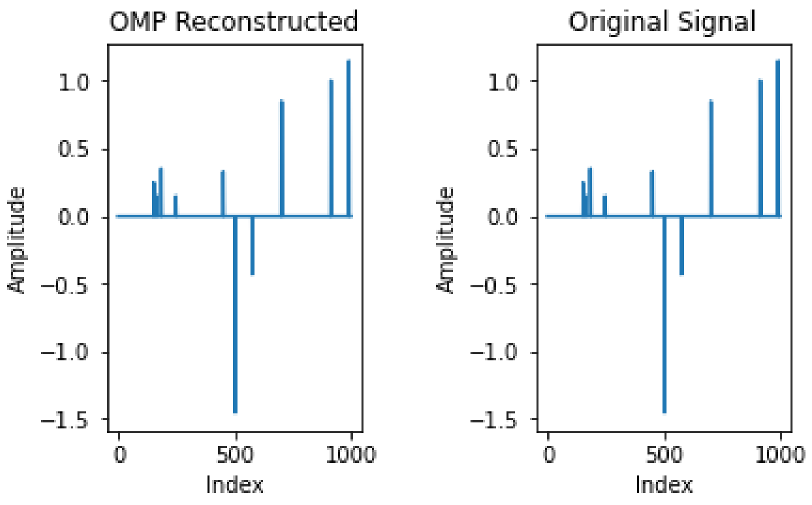

Figure 2.2: 1D sparse signal recovery with OMP.

The error of the reconstructed signal was $\varepsilon = 6 \cdot 10^{-15}\%$.

Example 2 (CoSaMP). This algorithm has the following scheme.

1. Initialization:

   Inputs: sensing matrix Φ, measurement vector y, sparsity S
   Initial values: $r = y$, $\Omega = \emptyset$, $k = 0$

2. k-th iteration with $k \in \{1, ..., S\}$:

   (a) Find the indices of the largest $2S$ coefficients of $\Phi^T r$ and save them in Ω

   (b) $T = \Omega \cup supp(x_{k-1})$.

   (c) $x_k = \arg\min_{x:supp(x)=T} \|\Phi x - y\|_2$

   (d) Keep only the largest S coefficients of x_k using hard-thresholding operator

(e) $r = y - \Phi x_k$

To implement this algorithm, the hard-thresholding operator is also needed.

```python
def hard_thresholding(x, k):
    x = x.flatten()
    length = x.size
    H_k_x = np.zeros_like(x)
    indices = list(np.argsort(np.absolute(x))[::-1][0:k].flatten())
    H_k_x[indices] = x[indices]
    H_k_x = np.reshape(H_k_x, (length, 1))

    return indices, H_k_x # Returns the largest k coefficients and their
        indices

def cosamp(A,y,S):

    x_previous = np.zeros_like(x)
    r = np.copy(y)
    T = []
    A_pinv_T_k = np.zeros_like(A)

    for k in range(S):
        Omega = hard_thresholding(A.T @ r,2*S)[0]
        supp_x_previous = x_previous.flatten().nonzero()[0].tolist()
        T = list(set().union(T,Omega,supp_x_previous))
        A_pinv_T_k[:,T] = A[:,T]
        x_k = hard_thresholding(np.linalg.pinv(A_pinv_T_k) @ y, S)[1]

        r = y - A @ x_k

    return x_k
cosamp_recovery = cosamp(Phi,y,S)
```

The outcome is depicted in Fig. 2.3 and the error obtained is $\varepsilon = 3 \cdot 10^{-15}\%$.

For both examples, the errors obtained are extremely small, around 10^{-15}, which may be associated to numerical error introduced by Python when computing the different operations. Therefore, we can assume that the reconstructions achieved are perfect.

2.4.2 Non-sparse signals

Let us consider now a non-sparse input signal as done in the code reported in [19]. In particular, let us consider a 1D signal, i.e., an artificial sound wave. It is the result of mixing two sine functions with different frequencies.

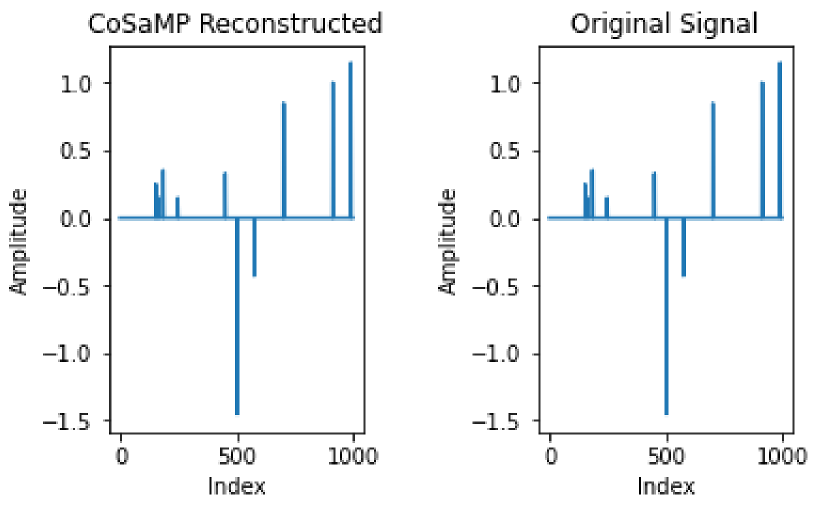

Figure 2.3: 1D sparse signal recovery with CoSaMP.

```
# Generate non sparse signal: artificial sound wave
N = 5000
t = np.linspace(0, 1/8, N)

f1 = 200
f2 = 3900
X = np.sin(2*np.pi*f1*t) + np.sin(2*np.pi*f2*t)
```

As shown in Fig. 2.4, the resulting signal is not sparse.

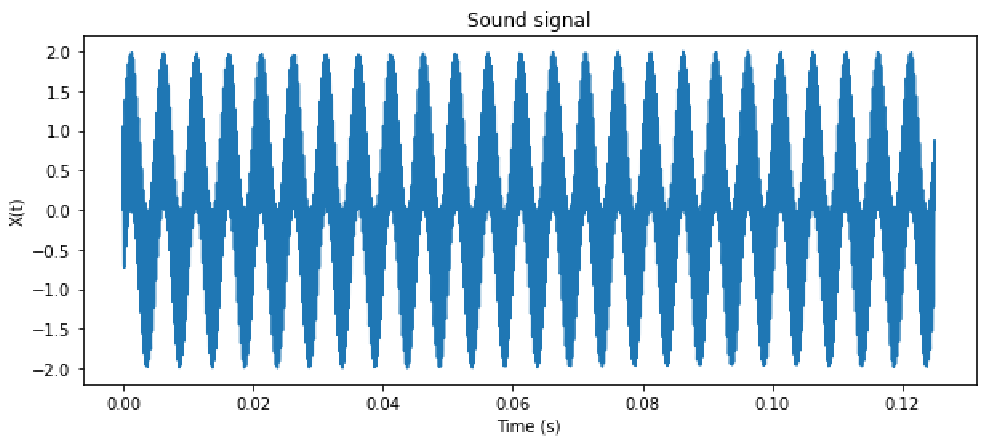

Figure 2.4: Artificial sound signal.

At this point, we need to apply a transform in order to have a sparse or compressible signal to be able to apply CS. Specifically, we will use the Discrete Cosine Transform (DCT) [2]:

```
# Compressible representation of a non-sparse signal
```

```
from scipy.fftpack import dct, idct
Xdct = dct(X,norm='ortho')
```

The DCT coefficients $y \in \mathbb{R}^N$ of a vector $x \in \mathbb{R}^N$ implemented by this function are:

$$y_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \tag{2.18}$$

$$y_k = \frac{2}{\sqrt{2N}} \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right) \text{ for } k \in \{1, .., N-1\} \tag{2.19}$$

The resulting DCT transform for the sound signal is shown in Fig. 2.5.

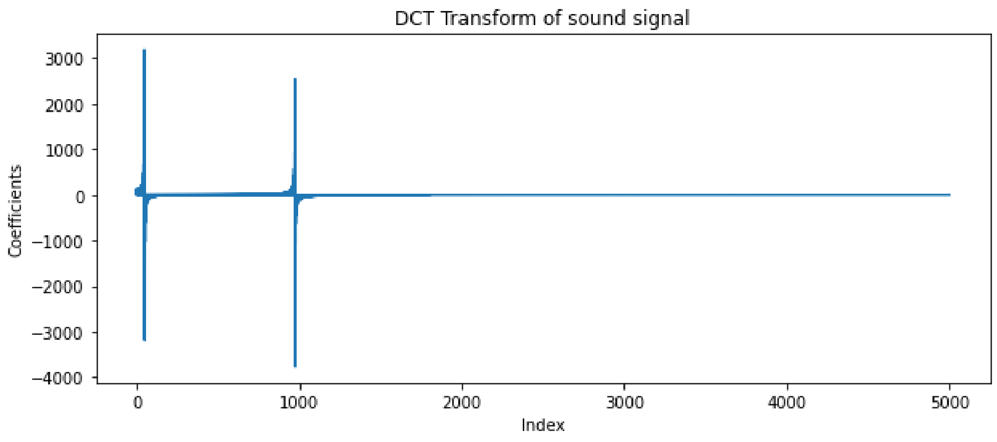

Figure 2.5: DCT transform of the sound signal.

Just as we did for the sparse case, we next set the number of measurements to take ($M = 500$, resulting in a compression ratio $M/N = 0.1$), build the sensing matrix, and apply this matrix. The only difference now is that it is applied to the transformed signal:

```
M=500
Phi= np.random.randn(M, N)/np.sqrt(M)
print ("Compression ratio {0}".format(M/N)) # M/N=0.1
reshape_signal=np.reshape(Xdct, (Xdct.shape[0],1))
y=np.dot(Phi,reshape_signal)
```

Finally, we have to choose a reconstruction algorithm. Let us consider convex minimization. There are various possible approaches. First, we could use the Python's library called CVXPY, tailored for convex optimization problems. The following code would solve the problem:

```
import cvxpy as cvx
```

```
vx = cvx.Variable((N,1))
objective = cvx.Minimize(cvx.norm(vx, 1))
constraints = [Phi@vx == y]
prob = cvx.Problem(objective, constraints)
result = prob.solve()
```

Another command that we can use is *Lasso* [61] from the Scikit-Learn library. It minimizes

$$\min_x \left\{ \frac{1}{2n} \|y - \Phi x\|_2^2 + \alpha \|x\|_1 \right\} \tag{2.20}$$

where α is a constant.

```
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=0.01)
lasso.fit(Phi,y)
```

Figure 2.6 shows the reconstruction of the transformed signal performed with Lasso.

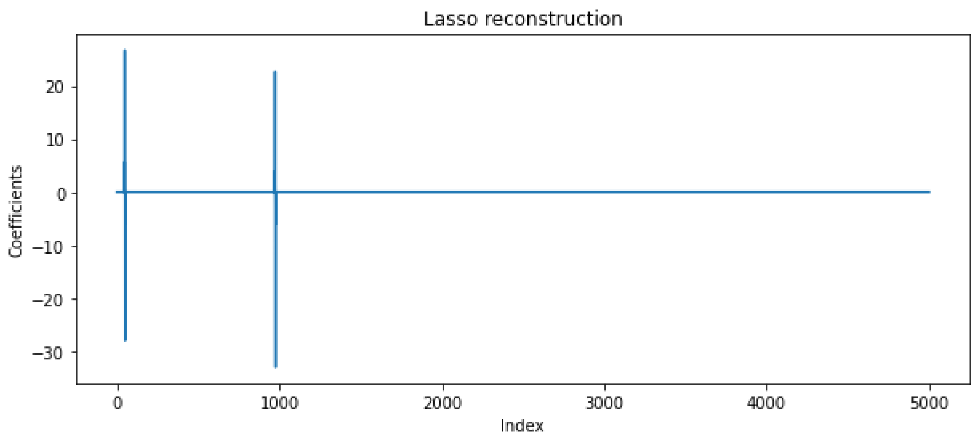

Figure 2.6: Lasso recovery of the transformed signal.

The last step it to reverse the DCT transform.

```
Xhat = idct(lasso.coef_,norm='ortho')
```

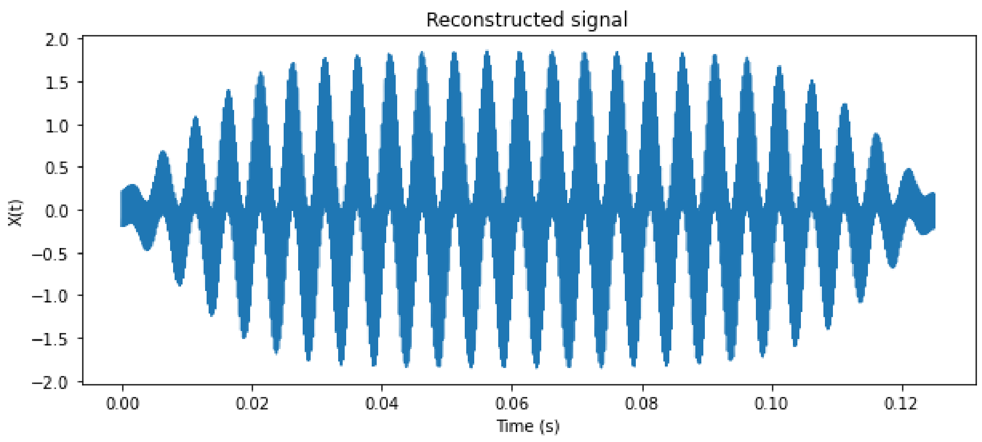

Figure 2.7: Reconstructed sound signal

The error obtained is $\varepsilon = 8\%$. However, the reconstruction shows a tendency towards zero in both extremes that cannot be controlled because there is no option to introduce boundary conditions in the reconstruction. Therefore, we focus on the middle section of the wave, from 0.04 s to 0.08 s. In this section the error is $\varepsilon = 2\%$. To appreciate the difference between the original signal and its reconstruction from compressed samples, we can zoom a section and represent it for both cases, as depicted in Fig. 2.8.

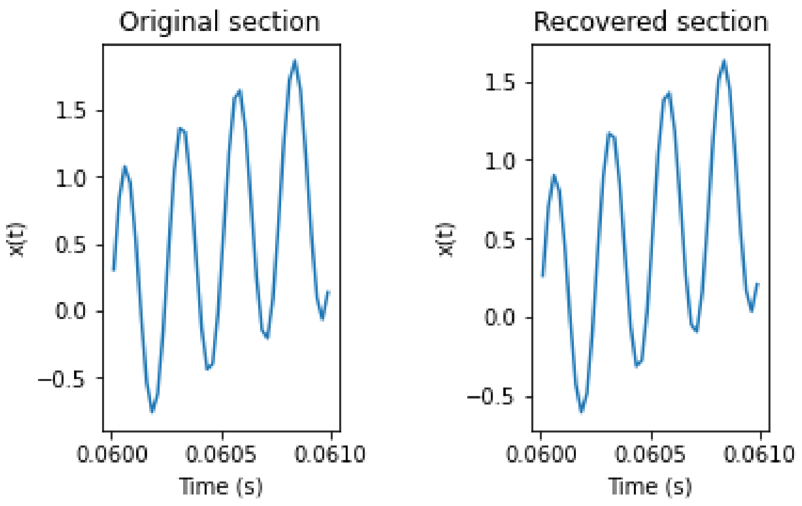

Figure 2.8: Comparison between an original section and its reconstruction.

2.4.3 Image Reconstruction

Images are a collection of pixels. If we have a grayscale image, then each pixel takes a value that represents brightness. This number is stored as an 8-bit integer. With 8 binary digits we can express numbers from 0 to 255. Let 0 be the value for black and 255 for white. The image has the form of a matrix with shape $width \times height$ where the elements are the values of the pixels.

Now let us suppose that our image is a color image. It will be usually encoded in RGB representation, where R stands for red, G for green and B for blue. The idea of this format is to divide the image into 3 layers called channels. Each one corresponds to one of the three colors. In each channel every pixel is represented with values from 0 to 255, where 0 is the color of the channel and 255 is white. Therefore, in this case the image is a matrix of size $width \times height \times channels$.

Next, we describe how to reconstruct a color image by means of convex optimization using the code provided in [60]. The underlying algorithm is OWL-QN [3]. It minimizes a scalar function of the form:

$$f(x) = l(x) + c\|x\|_1 \tag{2.21}$$

where $l : \mathbb{R}^N \to \mathbb{R}$ is a differentiable convex loss function, $\nabla l(x)$ is $L-$Lipschitz continuous for some $L > 0$, and C is a constant. The expression becomes

$$f(x) = \|\Phi x - y\|_2^2 \tag{2.22}$$

The following libraries are used:

```python
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import scipy.optimize as spopt
import scipy.fftpack as spfft
from skimage import io
from pylbfgs import owlqn
```

Note that images are not usually sparse. Therefore, we need to use a transform basis, in this case the DCT transform. We need a two-dimensional transform and, of course, we also need the inverse 2D transform. Python does not provide them, hence the following functions are defined:

```python
def dct2(x):
    return spfft.dct(spfft.dct(x.T, norm='ortho', axis=0).T, norm='ortho',
        axis=0)

def idct2(x):
    return spfft.idct(spfft.idct(x.T, norm='ortho', axis=0).T, norm='ortho'
        , axis=0)
```

The code begins defining the sampling rate. This is M/N in our theoretical framework. Then the image is loaded into memory and its shape stored. An auxiliary variable Z collects the reconstructions done for different compressions. Likewise, an auxiliary function called `evaluate`, needed by the algorithm, is also defined. The code provided in [60] is presented below, with the notation $f(x) = \|Ax - b\|_2^2$ and $\nabla f(x) = 2A^T(Ax - b)$.

```
def evaluate(x, g, step):
    """An in-memory evaluation callback."""

    # we want to return two things:
    # (1) the norm squared of the residuals, sum((Ax-b).^2), and
    # (2) the gradient 2*A'(Ax-b)

    # expand x columns-first
    x2 = x.reshape((nx, ny)).T

    # Ax is just the inverse 2D dct of x2
    Ax2 = idct2(x2)

    # stack columns and extract samples
    Ax = Ax2.T.flat[ri].reshape(b.shape)

    # calculate the residual Ax-b and its 2-norm squared
    Axb = Ax - b
    fx = np.sum(np.power(Axb, 2))

    # project residual vector (k x 1) onto blank image (ny x nx)
    Axb2 = np.zeros(x2.shape)
    Axb2.T.flat[ri] = Axb # fill columns-first

    # A'(Ax-b) is just the 2D dct of Axb2
    AtAxb2 = 2 * dct2(Axb2)
    AtAxb = AtAxb2.T.reshape(x.shape) # stack columns

    # copy over the gradient vector
    np.copyto(g, AtAxb)

    return fx

# fractions of the scaled image to randomly sample at
sample_sizes = (0.1, 0.01)

# read original image
Xorig = io.imread("leon.PNG")
ny,nx,nchan = Xorig.shape
```

```
# for each sample size
Z = [np.zeros(Xorig.shape, dtype='uint8') for s in sample_sizes]
masks = [np.zeros(Xorig.shape, dtype='uint8') for s in sample_sizes]
for i,s in enumerate(sample_sizes):

    # create random sampling index vector
    k = round(nx * ny * s)
    ri = np.random.choice(nx * ny, k, replace=False) # random sample of
        indices

    # for each color channel
    for j in range(nchan):

        # extract channel
        X = Xorig[:,:,j].squeeze()

        # take random samples of image, store them in a vector b
        b = X.T.flat[ri].astype(float)

        # perform the L1 minimization in memory
        Xat2 = owlqn(nx*ny, evaluate, None, 5)

        # transform the output back into the spatial domain
        Xat = Xat2.reshape(nx, ny).T # stack columns
        Xa = idct2(Xat)
        Z[i][:,:,j] = Xa.astype('uint8')
```

The results obtained for this code are:

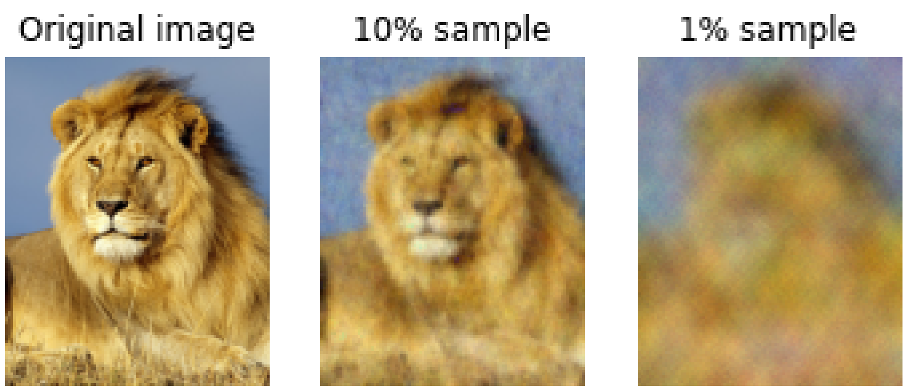

Figure 2.9: Image recovery.

The error for a 10% compression rate is $\varepsilon = 3\%$ and for a compression rate of 1% we have $\varepsilon = 3.4\%$.

Another way of evaluating the quality of the reconstruction is with the Signal to Noise Ratio (SNR). It compares in relative terms how the reconstructed image departs from the original one [35]. It is usually expressed in decibels.

Definition 10 (Signal to Noise Ratio)**.** *Let P_i be the value of the i-th pixel in the original image and let Q_i be the value of the i-th pixel of the reconstructed image. SNR can be defined as follows:*

$$\text{SNR} = 20 \log_{10} \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} P_i^2}}{\sqrt{MSE}} \tag{2.23}$$

where $\text{MSE} = \frac{1}{N}\sum_i (P_i - Q_i)^2$ *is the Mean Square Error and* $\sqrt{\frac{1}{N}\sum_{i=1}^{N} P_i^2}$ *is the Root Mean Square of the original image.*

For $M/N = 0.1$ we have SNR= 1.3 dB and for $M/N = 0.01$ the result is $SNR = 0.4$ dB. This makes sense because the higher the value of SNR the better the reconstruction achieved.

2.5 Applications

There is a variety of application fields of compressed sensing. In [51] and [49], the most important applications are reviewed. We next briefly summarised some of them.

1. `Compressive imaging`

   Compressed sensing allows saving memory when images are stored because only a reduced set of measurements are needed for reconstruction. Therefore, new technologies are being developed to directly obtain compressive visual samples.

   An example of these devices is the Single Pixel-Camera (SPC) [30]. The idea is two perform inner products between an N-pixel sampled version of the scene and random vectors formed by elements with values in $\{1, 0\}$ or in $\{1, -1\}$. M inner products $\langle x, \phi_i \rangle$ for $i \in \{1, 2..., M\}$ have to be computed in order to obtain the M measurements we need for CS. The vector ϕ_i represents a column of the sensing matrix.

The operation of this camera is based on a Digital Micromirror Device (DMD), which consists on an array of N mirrors that can rotate into two positions that corresponds to be switched on or off. Light reflects in the image and reaches these micromirrors, which deviate part of the beam onto a single photon detector. This is how an inner product is done, with mirrors acting as the elements of ϕ_i.To perform a random measurement as proposed by CS, the orientations of the mirrors are set randomly. Finally, the output of the detector, where the product is integrated, is a voltage that is finally digitised by an A/D converter.

With each micromirror we can represent a value of 1 or 0. If we want ϕ_i to take values in $\{1, -1\}$, we can express our ϕ_i as the subtraction of two vectors formed of ones and zeros. Therefore, the inner product $< x, \phi_i >$ is the subtraction of two inner products.

2. `Biomedical applications`

CS has also been applied in biomedical imaging. It has been shown that CS reduces the scanning time, especially in Magnetic Resonance Imaging (MRI) [44], which is a diagnostic tool.

In MRI, a constant magnetic field is applied to the patient. It interacts with hydrogen atoms in water molecules of the body polarising their spin into parallel or anti-parallel positions with respect to the field applied. These nuclei also rotate around the axis of the field with an angular frequency called precessional frequency.

With an additional radio frequency pulse the nuclei are excited, causing some of them to absorb the energy of the pulse and move to a higher energy state, i.e., they move from parallel to anti-parallel. In this stage, the nuclei are said to be at resonance. When the pulse is turned off, nuclei go back to their low energy state, releasing energy which is detected by a sensor. As the rate of the energy release depends on the tissue's chemical properties, an image of the body can be obtained.

Data acquisition is done in the Fourier-transform domain called K-space, i.e., in the frequency domain. However, the time needed to produce an image of certain quality is large, which limits medical applications of MRI. For this reason, the use of CS during the acquisition process has been explored. The fewer the samples needed, the less the time required. Algorithms applying CS have been developed specifically for this field. In [25], a review of convex-optimization-based algorithms for MRI is provided.

3. `Compressive radar`

A radar consists of an antenna that emits an electromagnetic wave. This wave is reflected in surrounding objects and then reaches a receiver. Using the time delay between the emission and the reception of the signal, the position of the object can be estimated. Speeds can also be measured with a radar.

At the receiver, there are two possible approaches: i) a pulse compression through a matched filter that consists in correlating the received signal with a template, followed by an A/D converter; and ii) an A/D converter followed by a pulse compression. These procedures require expensive equipment.

Using CS in radar systems results in a simplification of the hardware design. According to [5], the matched filter to compress the pulse can be eliminated, thereby removing the requirement of high-rate A/D conversion at the receiver, which relies on the Nyquist rate.

4. `Communication systems`

CS can be found in different areas of this application field. For example, it has been proposed for wireless sensor networks (WSN) [43].

Suppose N sensors densely deployed in a prescribed area of interest; d_j denotes the reading of the j-th sensor. The usual transmission procedure consists of a chain of transmission in which a sensor s_j sends the readings of the first j sensors to s_{j+1}. This procedure continues until the last sensor sends all the readings to the sink. The disadvantage of this system is that the last sensors consume more energy.

In the proposed compressed data gathering approach, M linear combinations of the readings are sent to the sink. In the i-th sum, each sensor multiplies its reading by a weight $\phi_{i,j}$ and adds it to the sum of the weighted reading of the previous sensors. Afterwards, this calculation is passed to the next sensor, i.e., the j-th sensor sends $\sum_{l=1}^{j} \phi_{i,l} d_l$. At the end of the process, the sink receives $\sum_{l=1}^{N} \phi_{i,l} d_l$ for all $i \in \{1, 2, ..., M\}$.

Chapter 3

CS-based inference: Classical approach

What is statistical inference? Suppose a population of which an attribute, represented by a random variable x, has an unknown distribution. For example, a group of people and their height. Samples of the value of x, represented as x_1, x_2, ..., x_n, are taken for some individuals of the population. From these results, we want to extract certain properties of the random variable. For instance, we could ask ourselves about its expected value or about the probability of having $x \geq \beta$ for a certain constant β.

This chapter explains some of the most important concepts of statistical inference [62, 67] and the theoretical framework proposed in [22] to solve inferential problems upon compressed measurements. In particular, the problems addressed are detection, classification, estimation and filtering.

3.1 Concepts of statistical inference

Definition 11 (Simple random sample). *It is said that $(x_1,\ x_2,\ ...,\ x_n)$ is a simple random sample of size n of a random variable x with distribution function F if they are random variables independent and identically distributed to x with common distribution F.*

Definition 12 (Statistic). *Let x be a random variable with distribution function F and let $\boldsymbol{x} = (x_1,\ x_2,..,x_n)$ be a simple random sample. It is said that $T : \mathbb{R}^N \rightarrow \mathbb{R}^k$ with $T = T(\boldsymbol{x})$ is a statistic if it is a measurable function.*

Some examples of statistics are:

1. Expected value: $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2. Variance: $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2$

For further theoretical development, we need to impose certain characteristics to the distribution function of the random variable we are working with. We assume the parametric hypothesis, which means that the distribution function F depends on a parameter. This is, $F \in \{F_\theta : \theta \in \Theta \subset \mathbb{R}^k\}$ where Θ is called the parametric space and it contains all the possible values of θ. Our aim, in parametric inference problems, is to know the value of a function $h : \Theta \to \mathbb{R}$.

Definition 13 (Estimator). *Let $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ be a simple random sample. $T = T(\boldsymbol{x})$ is said to be an estimator of $h(\theta)$ if T is a statistic that does not depend on unknown parameters and as far as possible $T(\boldsymbol{x}) \in h(\Theta)$ $\forall \boldsymbol{x}$.*

Example 3. Suppose a group of people waiting for judgement. We want to know the probability of being innocent.

The random variable takes two values, $x = 1$ if a person is innocent and $x = 0$ if the person is guilty. Our variable has a Bernoulli distribution of unknown parameter p, $x \sim Be(p)$. Therefore, we can take p as our parameter θ with $p \in (0, 1) = \Theta$.

Given that we want to know $P[x = 1] = p$, our target function is $h(p) = p$. A possible estimator is $T = \bar{\mathbf{x}}$. To check this we need to prove that $T(\mathbf{x}) \in h(\Theta) = (0, 1)$ for almost every $\mathbf{x}$.

The possible values for $\mathbf{x}$ are the different tuples of n elements that can be formed with zeros and ones. T applied to them gives us $0, 1/n, 2/n, ..., 1 \in [0, 1]$. Therefore, T is an estimator of $h(p)$.

Statistics are functions that summarize the information of the simple random sample studied and therefore, they collect information about θ. It is natural to think that some statistics are better than others in the sense that they do not lose relevant information. Following the example, suppose $n = 50$ and $\mathbf{x} = (1, 0, ..., 0, 1)$, which only has two ones. With the statistic we defined above, $T(\mathbf{x}) = \bar{\mathbf{x}} = 2/n = 0.04$. Hence, we can infer that the value of p is small. However, we could propose another statistic. For example, $T_2(\mathbf{x}) = \prod_{i=1}^{n} x_i$. In our case $T_2(\mathbf{x}) = 0$, which implies there is at least a zero in the simple random sample, but it does not provide information about p. The best way to collect significant information about the parameter θ from a random sample is using a sufficient statistic.

Definition 14 (Sufficient statistic). *Let $(x_1, x_2, ..., x_n)$ be a simple random sample and $T = T(x_1, x_2, ..., x_n)$ a statistic. It is said to be a sufficient statistic for θ if the conditional probability distributions of $(x_1, ..., x_n)$ given $T = t_0$ do not depend on θ for all t_0.*

A simple way of characterizing sufficient statistics is using the Fisher-Neyman factorization theorem.

Theorem 3 (Fisher-Neyman Factorization Theorem). *Let $(x_1, x_2, ..., x_n)$ be a simple random sample independently identically distributed to x, whose distribution function is f, and let $T = T(x_1, x_2, ..., x_n)$ be a statistic. T is a sufficient statistic if and only if there exist g and h such that*

$$f(\boldsymbol{x}, \theta) = g\left[T(\boldsymbol{x}), \theta\right] h(\boldsymbol{x}) \tag{3.1}$$

Now let us suppose we are interested in knowing whether a certain statement of our unknown distribution is true or not. Following the previous example, if the police does a good work, the majority of the people who go to trial are guilty. Therefore, we could assume that the probability of being innocent is small. Expressing this mathematically, our hypothesis could be written as:

$$H_0 : p \leq 1/100$$

This hypothesis is matched to an alternative hypothesis that considers a scenario in which H_0 is not true. For example:

$$H_1 : p > 1/100$$

Is there a way to determine which of them is the correct one? The part of statistical inference that deals with this type of problems is called hypothesis testing.

Definition 15 (Null and alternative hypothesis). *Let $x_1, x_2, ..., x_n$ be a simple sample of a random variable x whose distribution function satisfies the parametric hypothesis, $F \in \{F_\theta : \theta \in \Theta \subset \mathbb{R}^k\}$. Let $\Theta = \Theta_0 \cup \Theta_1$ be formed by two disjoint subsets, $\Theta_0 \cap \Theta_1 = \emptyset$. Suppose we want to test $H_0 : \theta \in \Theta_0$ vs. $H_1 : \theta \in \Theta_1$. It is said that H_0 is the null hypothesis and H_1 is the alternative hypothesis.*

The idea of a test is whether to reject or not the null hypothesis according to the experimental evidence. This is, for some values of $\mathbf{x} = (x_1, x_2, ..., x_n)$ we accept H_0 and for others we do not. This creates a division in the space containing all possible values of $\mathbf{x}$ which is denoted by Ω.

The critical region $R \subset \Omega$ is the set of $\mathbf{x} \in \Omega$ such that H_0 is rejected. In the same way, we can define the acceptance region $\Omega \setminus R$ in which H_0 is accepted. To distinguish between both regions a test statistic is used. According to its value, it is decided whether the null hypothesis is accepted or not.

Definition 16 (Critical region)**.** *Given a hypothesis test, the critical region is defined as:*

$$R = \{\boldsymbol{x} \in \Omega \mid s(\boldsymbol{x}) > c\} \tag{3.2}$$

where s is the test statistic and c is the critical value.

When taking a decision, errors are always possible. In fact, there are two types of errors in hypothesis testing. First, the null hypothesis could be rejected when it is true. This is called Type-I error. Second, the Type-II error consists in accepting the null hypothesis when it is false. Usually, in practice, the Type-I error is the one that is minimized.

The probability of Type-I error is:

$$P_I = P[\mathbf{x} \in R \mid \theta \in \Theta_0] \tag{3.3}$$

The probability of Type II-error is:

$$P_{II} = P[\mathbf{x} \in \Omega \setminus R \mid \theta \in \Theta_1] \tag{3.4}$$

where $P[\cdot]$ denotes the probability of something to occur.

Inference theory and many statistic concepts can be applied to signal processing [57]. Furthermore, the idea of hypothesis testing can be applied when working with compressed measurements. In the next sections, we analyse different inference problems following the guidelines reported in [22].

3.2 Detection

Suppose we have measurements of a signal $x \in \mathbb{R}^N$ taken according to CS principles and we want to distinguish whether it is pure noise or it contains a known signal of interest with some noise. We can model noise with a Gaussian distribution of null mean and σ^2

variance, i.e., $n \sim \mathcal{N}(0, \sigma^2 I_N)$. The signal to detect is s. The hypothesis we have can be written as:

$$\begin{cases} H_0 : & y = \Phi n \\ H_1 : & y = \Phi(s+n) \end{cases} \tag{3.5}$$

Another perspective for the detection problem statement is given in [29]. Suppose that a signal x is sparse in Ψ and $x = \sum_i^N \alpha_i \Psi_i$. Let Γ be a set of indices with $\Gamma \subset \{1, 2, .., N\}$. The objective is to know whether there are any non-zero coefficients α_i associated to the indices in Γ or not. Thus, the detection of the elements forming Ψ can be done because we could know whether x is formed by them. If $s = \Psi_\Gamma \alpha_\Gamma$, we can express x as follows:

$$x = \begin{pmatrix} \Psi_\Gamma & \Psi_I \end{pmatrix} \begin{bmatrix} \alpha_\Gamma \\ \alpha_I \end{bmatrix} + n = s + I + n \tag{3.6}$$

where n again denotes Gaussian noise and I makes reference to the interference in the signal x. We want to know if $\alpha_\Gamma = 0$ or $\alpha_\Gamma \neq 0$. The hypothesis testing associated to our problem would be

$$\begin{cases} H_0 : & x = I + n \\ H_1 : & x = s + I + n \end{cases} \tag{3.7}$$

However, for the detection problem, instead of using the complete signal x, we only take incoherent measurements $y = \Phi x$. See Definition 9. Therefore, the hypothesis testing can be reformulated as

$$\begin{cases} H_0 : & y = \Phi(I + n) \\ H_1 : & y = \Phi(s + I + n) \end{cases} \tag{3.8}$$

In [29], a greedy algorithm based on the Matching Pursuit reconstruction algorithm is proposed for this detection problem. See Point 2 in Section 2.3 for further information about greedy algorithms.

Considering the case $\Gamma = \{1, 2, \ldots, N\}$, this formulation leads us back to the first one given at the start of this subsection. We focus on this case from now on. This is, we aim at solving Eq. (3.5). We continue by defining two probabilities that are used in our theoretical framework.

Definition 17 (Detection rate and False alarm)**.** *Given the hypothesis testing, the detection rate is defined as the probability of choosing H_1 when it is true. This is, the probability* *of doing the detection:*

$$P_D = P[\ \textit{Chose}\ H_1\ \textit{when}\ H_1\ \textit{is true}\ ] \tag{3.9}$$

The false alarm rate is defined as

$$P_F = P[\ \textit{Chose}\ H_1\ \textit{when}\ H_0\ \textit{is true}\ ] \tag{3.10}$$

By comparing definitions, we have that the false alarm rate is just the Type-I error of the hypothesis test. Given both definitions, we want to maximize P_D while controlling the value of P_F. We could do this by bounding the error, $P_F \leq \gamma$, in which case the decision rule that satisfies this constrained problem is the Neyman-Pearson test [57]. Let $f_0(y)$ denote the density function associated to H_0; likewise for $f_1(y)$ to H_1. The likelihood ratio is defined as

$$\Lambda(y) = \frac{f_1(y)}{f_0(y)} \tag{3.11}$$

Comparing this ratio to a threshold η, the election of hypothesis is done. According to the Neyman-Pearson criterion, $\Lambda > \eta$ for H_1 and $\Lambda < \eta$ for H_0. The value of η is calculated by setting $P_F = \gamma$.

Example 4. To illustrate how to obtain the Neyman-Pearson test we consider a case with a one-dimensional variable distributed according to a Gaussian distribution with known σ, and we set a hypothesis test for the mean [55]:

$$\begin{cases} H_0: & x \sim \mathcal{N}(0, \sigma^2) \\ H_1: & x \sim \mathcal{N}(1, \sigma^2) \end{cases} \tag{3.12}$$

The expressions for the density function are known for both cases.

$$f_0(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \tag{3.13}$$

$$f_1(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-1)^2}{2\sigma^2}\right) \tag{3.14}$$

For example, we can take $\sigma = 1$, and consequently we have

$$\Lambda(x) = \frac{f_1(x)}{f_0(x)} = \exp\left(x - \frac{1}{2}\right) \tag{3.15}$$

This leads to the condition:

$$\begin{cases} \exp\left(x-\frac{1}{2}\right) < \eta & \text{for } H_0 \\ \exp\left(x-\frac{1}{2}\right) > \eta & \text{for } H_1 \end{cases} \tag{3.16}$$

For simplification, we can apply a monotonic function to both sides of the inequalities. We use logarithm, and taking $\eta^* = \ln \eta + 1/2$ we obtain the following decision rule: [1]

$$\begin{cases} x < \eta^* & \text{for } H_0 \\ x > \eta^* & \text{for } H_1 \end{cases} \tag{3.17}$$

To calculate η^* we impose $P_F = \gamma$. The Gaussian distribution function is $\phi(x)$ and we define $Q(x) = 1 - \phi(x)$:

$$P_F = \int_{x>\eta^*} f_0(x)dx = \int_{\eta^*}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx = 1 - \phi(\eta^*) = Q(\eta^*) = \gamma \tag{3.18}$$

We conclude that $\eta^* = Q^{-1}(\gamma)$. Also, P_D can be calculated as

$$P_D = \int_{\eta^*}^{\infty} f_1(x)dx = Q(\eta^* - 1) \tag{3.19}$$

Finally, for our decision rule, the detection rate is

$$P_D = Q(Q^{-1}(P_F) - 1) \tag{3.20}$$

Our aim in this section is to apply this same procedure to the hypothesis testing in Eq. (3.5), as reported in [22]. The principal difference between this case and the example is that now we have more than one dimension. In particular, y is $M-$dimensional.

Assuming $\Phi\Phi^T$ is invertible, the probability density functions are:

$$f_0(y) = \frac{\exp\left(-\frac{1}{2}y^T(\sigma^2\Phi\Phi^T)^{-1}y\right)}{|\sigma^2\Phi\Phi^T|^{1/2}(2\pi)^{M/2}} \tag{3.21}$$

$$f_1(y) = \frac{\exp\left(-\frac{1}{2}(y-\Phi s)^T(\sigma^2\Phi\Phi^T)^{-1}(y-\Phi s)\right)}{|\sigma^2\Phi\Phi^T|^{1/2}(2\pi)^{M/2}} \tag{3.22}$$

Now the Neyman-Pearson test is set. We take logarithm and obtain a simplified decision rule. The compressive detector is defined as

$$t := y^T(\Phi\Phi^T)^{-1}\Phi s \tag{3.23}$$

[1] In inference, the usual notation for η^* is γ and the constrain of the false alarm rate is $P_F = \alpha$. However, in this thesis α is reserved to denote sparse signals. Therefore, the notation has been changed.

and taking

$$\eta^* = \sigma^2 \ln \eta + 1/2 s^T \Phi^T (\Phi\Phi^T)^{-1} \Phi s \tag{3.24}$$

the resulting test can be expressed as:

$$\begin{cases} t < \eta^* & \text{for } H_0 \\ t > \eta^* & \text{for } H_1 \end{cases} \tag{3.25}$$

Now, we have to calculate η^*. To simplify the notation, we write $P_{\Phi^T} = \Phi^T(\Phi\Phi^T)^{-1}\Phi$. This new operator verifies $P_{\Phi^T} = P_{\Phi^T}^T$ and $P_{\Phi^T} = P_{\Phi^T}^2$. These are the properties of an orthogonal projection operator. In fact, we have that P_{Φ^T} projects into the row space of Φ. Furthermore, we have $s^T\Phi^T(\Phi\Phi^T)^{-1}\Phi s = \|P_{\Phi^T} s\|_2^2$.

With this new notation, we have that the distribution of t in both hypothesis is:

$$\begin{cases} t \sim \mathcal{N}(0, \sigma^2 \|P_{\Phi^T} s\|_2^2) & \text{for } H_0 \\ t \sim \mathcal{N}(\|P_{\Phi^T} s\|_2^2, \sigma^2 \|P_{\Phi^T} s\|_2^2) & \text{for } H_1 \end{cases} \tag{3.26}$$

The probabilities P_F and P_D are calculated as in the example, i.e., $P_F = Q(\frac{\eta^*}{\sigma\|P_{\Phi^T} s\|_2})$ and $P_D = Q(\frac{\eta^* - \|P_{\Phi^T} s\|_2^2}{\sigma\|P_{\Phi^T} s\|_2})$. By doing $P_F = \gamma$, the threshold value is $\eta^* = \sigma\|P_{\Phi^T} s\|_2 Q^{-1}(\gamma)$, and we obtain

$$P_D(\gamma) = Q(Q^{-1}(\gamma) - \|P_{\Phi^T} s\|_2/\sigma) \tag{3.27}$$

Given that Q is a monotonic decreasing function, the smaller the argument it takes, the larger the value it provides. Therefore, the larger the value of $\|P_{\Phi^T} s\|_2/\sigma$, the greater the detection rate. From now on, we aim to bound the value of P_D to characterize the performance of the detection.

Let us first introduce a new property.

Definition 18 ($\delta-$stable embedding)**.** *Let $A, B \subset \mathbb{R}^N$ be given and $\delta \in (0,1)$. It is said that Φ is a $\delta-$stable embedding of (A, B) if $\forall a \in A$ and $\forall b \in B$ it is satisfied that*

$$(1-\delta)\|a-b\|_2^2 \leq \|\Phi a - \Phi b\|_2^2 \leq (1+\delta)\|a-b\|_2^2 \tag{3.28}$$

Theorem 4. *Suppose $\sqrt{N/M}P_{\Phi^T}$ is a $\delta-$stable embedding of $(\mathcal{S}, \{0\})$ for $\mathcal{S} \subset \mathbb{R}^N$. Then P_D is bounded for every $s \in \mathcal{S}$.*

$$P_D(\gamma) \leq Q(Q^{-1}(\gamma) - \sqrt{1+\delta}\sqrt{M/N}\sqrt{SNR}) \tag{3.29}$$

$$P_D(\gamma) \geq Q(Q^{-1}(\gamma) - \sqrt{1-\delta}\sqrt{M/N}\sqrt{SNR}) \tag{3.30}$$

where the signal to noise ratio is defined as $SNR = \|s\|_2^2/\sigma^2$.

The proof is provided in [22]. However, it just needs to consider the definition of $\delta-$stable embedding. The article states also that for typical values of δ,

$$P_D(\gamma) \approx Q\left[Q^{-1}(\gamma) - \sqrt{M/N}\sqrt{SNR}\right]. \tag{3.31}$$

With the following corollary, [22] establishes a lower bound for the detection rate of s:

Corollary 1. *Suppose* $\sqrt{N/M}P_{\Phi^T}$ *is a* $\delta-$*stable embedding of* $(\mathcal{S}, \{0\})$ *for* $\mathcal{S} \subset \mathbb{R}^N$. *Then for* $s \in \mathcal{S}$ *the next bound is satisfied.*

$$P_D(\gamma) \geq 1 - C_2 \exp(-C_1 M/N) \tag{3.32}$$

where C_1 *and* C_2 *are constants depending on* γ, δ *and SNR.*

If all the parameters are fixed except M, we have that, for growing values of M, the exponential function decreases and P_D is lower bounded by a value near 1. This implies that the probability of detecting s is high.

To finish this section, we reproduce graphics presented in [22]. We represent the approximation of P_D in Eq. (3.31) by varying parameters to see the response of the detection probability. For this purpose, we have to implement the function $Q(x)$ that appears in Eq. (3.31). Using the properties of the Gaussian distribution we have that

$$Q(x) = 1 - P[\mathcal{N}(0,1) \leq x] = 1 - \phi(x) = \phi(-x) \tag{3.33}$$

To define $Q^{-1}(x)$ we have to consider the percentile of $-x$. In Python, these functions can be constructed as:

```
from scipy.stats import norm

Q(x)= norm.cdf(-x)
Qinv(x) = -norm.ppf(x)
```

In Fig. 3.1, P_D is represented against γ, which is the value of the false alarm rate, for different values of $r = M/N$ with a fixed value for SNR of 20 dB. However, the value of SNR introduced in the equation is not in dB. In this case, it would be $SNR = 100$. This type of curve is called Receiver Operating Characteristic (ROC). It is used to study the

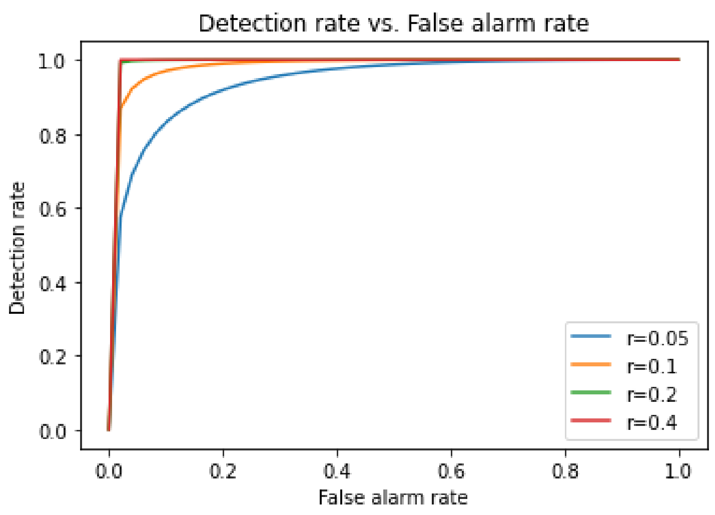

Figure 3.1: Detection rate vs. False alarm rate.

performance of a decision rule.

According to Fig. 3.1, for large values of the false alarm rate, the performance of the detector is identical for all curves. However, if the focus is on small values of P_F, we can see that for $M/N = 0.4$ or $M/N = 0.2$ the detection rate is almost 1, whereas for smaller sets of measurements the difference in performance with these last values is small. Therefore, we conclude that with M measurements, detection could be achieved successfully. However, the difference between curves with high and low values of M could imply that very low values of M are still not appropriate for detection.

In Fig. 3.2, P_D is represented as a function of M/N for different values of SNR with $\gamma = 0.1$. Note how the SNR value affects P_D. For higher SNR the performance improves. Note also that as M increases, the detection rate approaches higher values exponentially.

3.3 Classification

Let $\mathcal{S} = \{s_i\} \subset \mathbb{R}^N$ with $i \in \{1, 2.., R\}$ be a collection of known signals and $n \sim \mathcal{N}(0, \sigma^2)$ a noisy signal. The goal is to determine which s_i is present in the noise when the information available is a measurement vector with M random samples of the received signal that are taken before knowing $s_1, ..s_R$.

For the classification problem, we can set a hypothesis testing of R hypothesis:

$$H_i : y = \Phi(s_i + n) \text{ for } i \in \{1, 2, .., R\} \tag{3.34}$$

Suppose $R = 2$. This case is called binary classification and it is equivalent to the

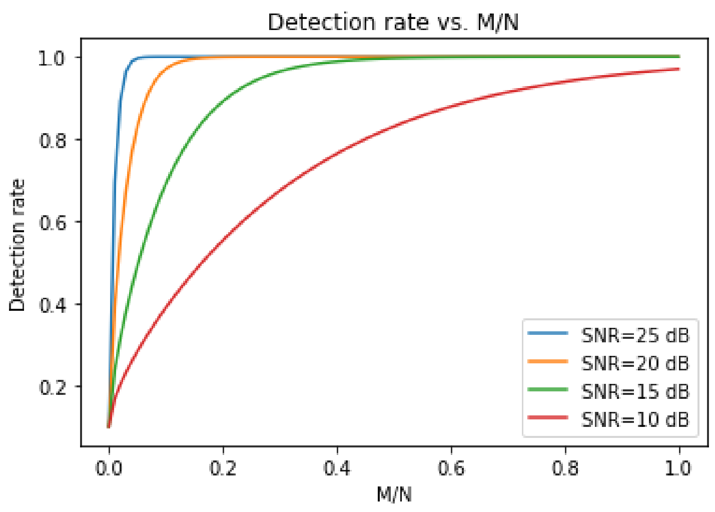

Figure 3.2: Detection rate vs. M/N.

detection problem set in the previous section. The hypothesis would be

$$\begin{cases} H_0: & y = \Phi(s_1 + n) \\ H_1: & y = \Phi(s_2 + n) \end{cases} \tag{3.35}$$

However, if $s = s_2 - s_1$, this formulation is equivalent to

$$\begin{cases} H_0: & y = \Phi(s_1 + n) - \Phi s_1 = \Phi n \\ H_1: & y = \Phi(s_2 + n) - \Phi s_1 = \Phi(s + n) \end{cases} \tag{3.36}$$

Returning to Eq. (3.34), if each hypothesis is equally likely, according to [22] the classifier with minimum probability error chooses H_i that minimizes

$$t_i := (y - \Phi s_i)^T (\Phi\Phi^T)^{-1} (y - \Phi s_i) \tag{3.37}$$

This is due to the fact that, for classification, the hypothesis selected is the one that best suits the data. Therefore, H_{i*} is chosen if the probability of y for H_{i*} is larger than for any other hypothesis.

Supposing that there exists $x \in \mathbb{R}^N$ such that $y = \Phi x$, we can write

$$\begin{aligned} t_i &= (y - \Phi s_i)^T (\Phi\Phi^T)^{-1} (y - \Phi s_i) && (3.38) \\ &= (\Phi x - \Phi s_i)^T (\Phi\Phi^T)^{-1} (\Phi x - \Phi s_i) && (3.39) \\ &= (x - s_i)^T \Phi^T (\Phi\Phi^T)^{-1} \Phi (x - s_i) && (3.40) \\ &= \|P_{\Phi^T} x - P_{\Phi^T} s_i\|_2^2 && (3.41) \end{aligned}$$

Therefore, the goal is to minimize the distance between x and s_i once they have been projected by P_{Φ^T}. With this formulation of t_i we conclude that the compressive classifier selects the nearest s_i to x in the row space of Φ.

Theorem 5. *Suppose that $\sqrt{(M/N)}P_{\Phi^T}$ is a $\delta-$stable embedding of $(\mathcal{S}, \mathcal{S})$. Let d be the minimum distance between s_i, $d = min\|s_i - s_j\|_2$. Let $y = \Phi(s_{i^*} + n)$ for $i^* \in \{1, 2, ..., R\}$. Then, the probability of correctly classifying the signal is lower bounded.*

$$P_{Classification} \geq 1 - \left(\frac{R-1}{2}\right) \exp\left(-d^2(1-\delta)M/8\sigma^2 N\right) \tag{3.42}$$

When the classification is correct, $i^* = \arg\min t_i$.

The proof is provided in [22]. From this result, we obtain that the error probability decreases exponentially with M, just as in the detection problem.

To illustrate the performance of the compressive classifier, we represent the error probability for different cases and analyse how it changes with M. We first set $N = 1000$ and $R = 3$. The signals s_1, s_2 and s_3 that match each hypothesis are generated according to a Gaussian distribution and are fixed. Then d is calculated, and for a certain value of $SNR = 10\log_{10}(d^2/\sigma^2)$, we can know the value of σ^2.

For every value of M, a Φ matrix is created. Then a loop is set with 300 iterations. In each one, the noise vector n is generated, and $x = s_1 + n$ is established, which means that the correct hypothesis is H_1. All t_i are calculated and the minimum of the three values is obtained. If the minimum was achieved in t_1, then H_1 is chosen and the classification was properly conducted. The error probability is given by the number of times the hypothesis taken was not H_1 divided by 300, which is the number of times that classification was made.

We can see in Fig. 3.3 that the lower the SNR the higher the error probability. Also, the error decreases for large values of M, as in detection. We also have that, for large values of M, the computational cost increases due to the size of Φ and the operations required to calculate t_i for $i \in \{1, 2, 3\}$.

In this section we have supposed that the probability of every hypothesis is the same, as in [22]. However, we could generalize this fact and assume that a priori probabilities of each hypothesis are known. This is, $f(H_1), ..., f(H_R)$ are known and the joint probability density function for every $i \in \{1, 2, ..., R\}$ is

$$f(y, H_i) = f_i(y) f(H_i) \tag{3.43}$$

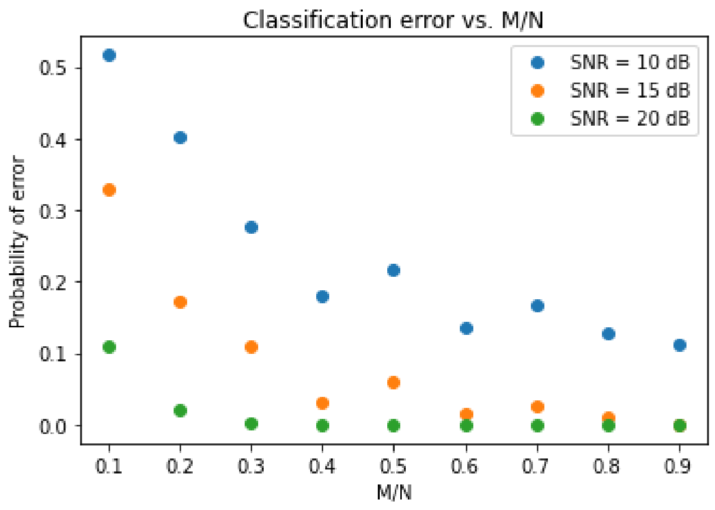

Figure 3.3: Error probability for classification vs. M/N.

where $f_i(y) = f(y|H_i)$.

In this case we are in the Bayesian Classification framework [57] and the hypothesis selected is the one with the largest a posteriori probability. This value is defined as

$$f(H_i|y) = \frac{f_i(y)f(H_i)}{f(y)} = \frac{f_i(y)f(H_i)}{\sum_{j=1}^{R} f(y, H_j)} \tag{3.44}$$

3.4 Estimation

In this section, the objective is to estimate $\langle l, x\rangle$ with $l \in \mathbb{R}^N$ and $y = \Phi x$. In [22], two estimators are presented. The first one, i.e., the orthogonalized estimator, is inspired in the compressive detector described in Section 3.2 and is given by $N/Mx^T\Phi^T(\Phi\Phi^T)^{-1}\Phi l$. The second estimator, i.e., the direct estimator, is computed by calculating the inner product of y and Φl, and is given by $\langle y, \Phi l\rangle$.

For both estimators we have a bound for the error. However, in [22] it is only stated and demonstrated for the direct estimator case. We next prove the result for the orthogonalized estimator.

Theorem 6. *Suppose that $l \in \mathcal{L} \subset \mathbb{R}^N$, $x \in \mathcal{X} \subset \mathbb{R}^N$ and Φ is a $\delta-$stable embedding of $(\mathcal{L}, \mathcal{X} \cup -\mathcal{X})$, then*

$$|\langle \Phi l, \Phi x\rangle - \langle l, x\rangle| \leq \delta \|l\|_2 \|x\|_2 \tag{3.45}$$

With the error bound for the direct estimator, we can estimate the angle between two vectors. The proof is provided in [22].

Corollary 2. *Suppose that $l \in \mathcal{L}$ and $x \in \mathcal{X}$, and Φ is a $\delta-$stable embedding of $(\mathcal{L} \cup \{0\}, \mathcal{X} \cup -\mathcal{X} \cup \{0\})$. Then*

$$\left|\cos\left(\widehat{\Phi l, \Phi x}\right) - \cos\left(\widehat{l, x}\right)\right| \leq 2\delta \tag{3.46}$$

where $\cos\left(\widehat{a, b}\right)$ represents the cosine of the angle between vectors a and b.

Now, we continue with the same result as for the estimation error but particularised to the case of the orthogonalised estimator. For the proof of the theorem, we follow the proof presented in [22] for the direct estimator. However, in this case, for the final reasoning, we have applied that P_{Φ^T} is an orthogonal projector.

Theorem 7. *Suppose that $l \in \mathcal{L} \subset \mathbb{R}^N$, $x \in \mathcal{X} \subset \mathbb{R}^N$ and $\sqrt{N/M}P_{\Phi^T}$ is a $\delta-$stable embedding of $(\mathcal{L}, \mathcal{X} \cup -\mathcal{X})$, then*

$$\left|\frac{N}{M}x^T P_{\Phi^T} l - \langle l, x\rangle\right| \leq \delta \|l\|_2 \|x\|_2 \tag{3.47}$$

Proof. We work under the assumption $\|l\|_2 = \|x\|_2 = 1$ and later generalize for vectors of arbitrary norm.

First, we have that $\sqrt{N/M}P_{\Phi^T}$ is a $\delta-$stable embedding of $(\mathcal{L}, \mathcal{X})$ and of $(\mathcal{L}, -\mathcal{X})$. Therefore,

$$(1-\delta)\|l \pm x\|_2^2 \leq N/M \|P_{\Phi^T} l - P_{\Phi^T} x\|_2^2 \leq (1+\delta)\|l \pm x\|_2^2 \tag{3.48}$$

We can expand $\|l \pm x\|_2^2$ as:

$$\|l \pm x\|_2^2 = \|l\|_2^2 + \|x\|_2^2 \pm 2\langle l, x\rangle = 2 \pm 2\langle l, x\rangle \tag{3.49}$$

Thus,

$$(1-\delta)(2 \pm 2\langle l, x\rangle) \leq N/M \|P_{\Phi^T} l - P_{\Phi^T} x\|_2^2 \leq (1+\delta)(2 \pm 2\langle l, x\rangle) \tag{3.50}$$

The parallelogram identity implies

$$\langle x, y\rangle = \frac{\|x+y\|_2^2 - \|x-y\|_2^2}{4} \tag{3.51}$$

Applying this equation we obtain

$$\begin{aligned}
\langle P_{\Phi^T} l, P_{\Phi^T} x\rangle &= \frac{\|P_{\Phi^T} l + P_{\Phi^T} x\|_2^2 - \|P_{\Phi^T} l - P_{\Phi^T} x\|_2^2}{4} && (3.52)\\
&\leq \frac{M}{2N}\left((1+\delta)(1+\langle l, x\rangle) - (1-\delta)(1-\langle l, x\rangle)\right) && (3.53)\\
&= \frac{M}{N}\left(\langle l, x\rangle + \delta\right) && (3.54)
\end{aligned}$$

Analogously, a lower bound can be obtained:

$$-\delta \leq \frac{N}{M}\langle P_{\Phi^T} l, P_{\Phi^T} x\rangle - \langle l, x\rangle \leq \delta \tag{3.55}$$

To conclude the result for $\|l\|_2 = \|x\|_2 = 1$ we have to take into account the properties of P_{Φ^T} as an orthogonal projector. First, we use that it is a self-adjoint operator, $P_{\Phi^T} = P_{\Phi^T}^T$, and then, given that it is a projector, $P_{\Phi^T}^2 = P_{\Phi^T}$. Thus,

$$\langle P_{\Phi^T} l, P_{\Phi^T} x\rangle = \langle P_{\Phi^T}^2 l, x\rangle = \langle P_{\Phi^T} l, x\rangle = x^T\Phi^T(\Phi\Phi^T)^{-1}\Phi l \tag{3.56}$$

With this we obtain the result when $\|l\|_2 = \|x\|_2 = 1$. To extended to vectors with arbitrary norm we just have to consider the theorem for $\hat{l} = \frac{l}{\|l\|_2}$ and $\hat{x} = \frac{x}{\|x\|_2}$.

□

We next estimate the mean value of a vector x. Fig. 3.4 depicts the estimation error vs. M/N. The first step is to set $N = 1000$. Then, the vector x is generated with a Gaussian distribution of mean and variance 1 and we take $l = 1/N[1, .., 1]$. Therefore, $\langle l, x\rangle = \sum_{i=1}^{N} x_i/N = \bar{x}$. For different values of M we generate a random matrix Φ 300 times and compute

$$\left|N/M x^T\Phi^T(\Phi\Phi^T)^{-1}\Phi l - \langle l, x\rangle\right| / (\|l\|_2\|x\|_2) \tag{3.57}$$

The point represented in Fig. 3.4 for a certain value of M is the mean estimation error over the 300 iterations.

Note that for large values of M the estimation error decreases rapidly. However, in this range, the size of Φ is high. This increases the computational cost, especially in the case of the orthogonalized estimator. On the other hand, the direct estimator presents a higher estimation error. When choosing an estimator for a specific application, one would have to decide between reducing computational costs or the estimation error.

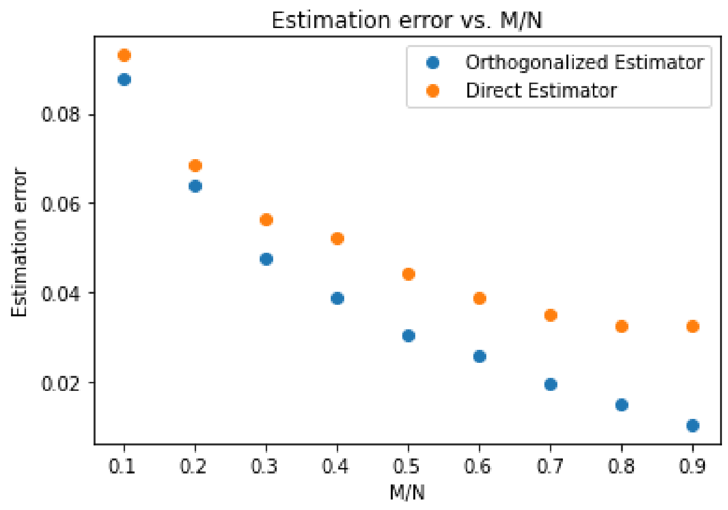

Figure 3.4: Estimation error vs. M/N.

A generalization of this theory to estimate vectors can be done easily. We just need to consider a matrix $L \in \mathcal{M}_{Z\times N}(\mathbb{R})$ and estimate $Lx \in \mathbb{R}^Z$ component by component. Therefore, Z estimations are made.

3.5 Filtering

Suppose we have transmitted a signal and it has suffered interference along the way. The signal x that reaches the receiver is the signal of interest $x_s \in \mathcal{S}_s$ plus the interference $x_I \in \mathcal{S}_I$. Before processing the signal x we have to identify the part of interest. This is, the goal is to filter the interference. However, in this thesis, we assume that the information available at the receiver is a measurement vector $y = \Phi(x_s + s_I)$. Therefore, we filter compressed measurements.

Theorem 8. *Let $\mathcal{S}_I$ be a K_I-dimensional subspace of $\mathbb{R}^N$ and assume that an orthonormal basis is placed in the columns of Ψ_I, which is a $N \times K_I$ matrix. Let $\Omega = \Phi\Psi_I$ and its pseudoinverse $\Omega^\dagger = (\Omega^T\Omega)^{-1}\Omega^T$. Let $P_\Omega = \Omega\Omega^\dagger$ be a projector onto the column space of Ω. Then $P_{\Omega^\perp} = I - \Omega\Omega^\dagger$ filters x_I in the sense that*

$$P_{\Omega^\perp}\Phi x_I = 0 \quad \forall x_I \in \mathcal{S}_\mathcal{I} \tag{3.58}$$

Proof. Let $Col(\Omega)$ and $R(\Omega)$ be the column and row spaces of Ω respectively. We have that P_Ω is an orthogonal projector that projects onto $Col(\Omega)$. This is equivalent to say that it projects onto the row space of Ω^T.

We have that $P_{\Omega^\perp}$ projects to the complementary space. Therefore, the null space of $P_{\Omega^\perp}$ is the space of the columns of Ω. This is, $Ker(P_{\Omega^\perp}) = Col(\Omega)$. Given that $\Phi x_I \in Col(\Omega)$, we have that $P_{\Omega^\perp}\Phi x_I = 0$ for all $x_I \in \mathcal{S}_\mathcal{I}$.

$\square$

With $P_{\Omega^\perp}$ we can eliminate interference. The problem is that we do not know how $\mathcal{S}_s$ is transformed by this operator. However, if $\mathcal{S}_s$ were orthogonal to $\mathcal{S}_I$, then its structure would be preserved under certain hypothesis. The following theorem incorporates this idea:

Theorem 9. *Let $\mathcal{S}_I$ be a K_I-dimensional subspace of $\mathbb{R}^N$ with orthonormal basis Ψ_I. Suppose that Φ is a $\delta-$stable embedding of $(\hat{\mathcal{S}}_s \cup \{0\}, \mathcal{S}_I)$. Let P_Ω and $P_{\Omega^\perp}$ be defined as in Theorem 8. Let $\hat{\mathcal{S}}_s = P_{\mathcal{S}_I^\perp}\mathcal{S}_s$ be the projection of $\mathcal{S}_s$ onto $\mathcal{S}_I^\perp$. Then, for all $x = \hat{x}_s + x_I$ with $\hat{x}_s \in \hat{\mathcal{S}}_s$ and $x_I \in \mathcal{S}_I$, the following properties are satisfied:*

$$P_{\Omega^\perp}\Phi x = P_{\Omega^\perp}\Phi \hat{x}_s \tag{3.59}$$

$$P_{\Omega}\Phi x = P_{\Omega}\Phi \hat{x}_s + \Phi x_I \tag{3.60}$$

$$1 - \frac{\delta}{1-\delta} \le \frac{\|P_{\Omega^\perp}\Phi \hat{x}_s\|_2^2}{\|\hat{x}_s\|_2^2} \le 1 + \delta \tag{3.61}$$

$$\frac{\|P_{\Omega}\Phi \hat{x}_s\|_2^2}{\|\hat{x}_s\|_2^2} \le \delta^2 \frac{1+\delta}{(1-\delta)^2} \tag{3.62}$$

According to Theorem 8, $P_{\Omega^\perp}\Phi x_I = 0$ for all $x_I \in \mathcal{S}_I$. We also have that $P_\Omega \Phi x_I = \Phi x_I$ because $\Phi x_I \in Col(\Omega)$. Moreover, taking into account the bounds presented in Theorem 9, we have that $P_{\Omega^\perp}$ preserves the structure of $\hat{\mathcal{S}}_s$, and P_Ω nearly cancels signals from $\hat{\mathcal{S}}_s$. The proof of these bounds is provided in [22].

From Theorem 9 we have two corollaries. With the first one we obtain properties of P_Ω and $P_{\Omega^\perp}$ that can be useful for performing other inference problems once the filtering has been done.

Corollary 3. *Let $\mathcal{S}_I$ be a K_I-dimensional subspace of $\mathbb{R}^N$ with orthonormal basis Ψ_I. Let Ω, P_Ω and $P_{\Omega^\perp}$ be defined as in Theorem 8. Suppose that Φ is a $\delta-$stable embedding of $(\hat{\mathcal{S}}_s \cup \{0\}, \mathcal{S}_I)$. Then $P_{\Omega^\perp}\Phi$ is a $\delta/(1-\delta)-$stable embedding of $(\hat{\mathcal{S}}_s, \{0\})$ and $P_\Omega\Phi$ is a $\delta-$stable embedding of $(\mathcal{S}_I, \{0\})$.*

The following corollary implies that the signal x_s with sparsity k_s can be recovered under the CS framework once the interference has been filtered.

Corollary 4. *Suppose that Ψ is an orthonormal basis of $\mathbb{R}^N$ and Φ is a $\delta-$stable embedding of $(\Psi(\Sigma_{2k_s}), Col(\Psi_I))$ where Ψ_I is an $N \times K_I$ submatrix of Ψ. Let P_Ω and $P_{\Omega^\perp}$ be defined as in Theorem 8. Then $P_{\Omega^\perp}\Phi$ is a $\frac{\delta}{1-\delta}-$stable embedding of $(P_{\Omega(\Psi_I)^\perp}\Psi(\Sigma_{2k_s}), \{0\})$.*

By comparing the RIP property (Definition 7) and a δ-stable embedding (Definition 18) we have that a sensing matrix Φ satisfies RIP of order $2k$ if and only if Φ is a $\delta-$stable embedding of $(\Sigma_{2k}, \{0\})$, and $\Phi\Psi$ satisfies RIP of order $2k$ if and only if Φ is a $\delta-$stable embedding of $(\Psi(\Sigma_{2k}), \{0\})$ with $\Psi(\Sigma_{2k}) = \{\Psi\alpha : \alpha \in \Sigma_k\}$. With these equivalences in mind and knowing that $P_{\Omega(\Psi_I)^\perp}\Psi(\Sigma_{2k_s})$ is the original set of sparse signals but with zeros in the positions indexed by Ψ_I, Corollary 4 implies that reconstruction is possible. In other words, the hypothesis of Theorem 2 are satisfied and the sensing matrix that we now consider is $P_{\Omega^\perp}\Phi$.

Now let us see how to apply these ideas of filtering and reconstruction in practical terms. Suppose that we have the compression of a signal x that is composed by a signal of interest x_s that we want to reconstruct, the interference x_I, and Gaussian noise. Therefore, $x = x_s + x_I + n$, and we want to recover just x_s. Using filtering on compressed measurements, [22] proposes a reconstruction approach called *cancel then recover*. It consists in filtering x_I directly on the measurements and then recover the signal with CoSaMP, as described in Point 2 of Section 2.3 and in Subsection 2.4.1.

To implement this idea, we take the sparsity of x_s as $k_s = 10$ and the sparsity of x_I as $k_I = 20$. We take separated supports for x_s and x_I and generate the coefficients according to a standard Gaussian distribution. Finally we re-scale x_I so that the signal-to-interference ratio is 0 dB. The signal-to-interference ratio is defined analogously to the SNR in Definition 10:

Definition 19 (Signal-to-Interference Ratio)**.** *Suppose that a signal $x \in \mathbb{R}^N$ containing a signal of interest x_s and interference x_I, i.e., $x = x_s + x_I$. The Signal-to-Interference Ratio is defined as the ratio between the root mean square of x_s and x_I.*

$$SIR = 20\log_{10}\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_s)_i^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_I)_i^2}} \tag{3.63}$$

Therefore, if we need $SIR = 0$ dB, then the norm of both signals has to be the same. We can obtain this by scaling x_I by the factor $\|x_s\|_2/\|x_I\|_2$. The next step is to generate the Gaussian noise; as in [22], we take the value of the variance so that SNR=15 dB, which is the ratio between the root mean square of the signal and the noise in decibels. If we impose that the mean of the noise is zero, then its root mean square is its variance, σ^2. We have:

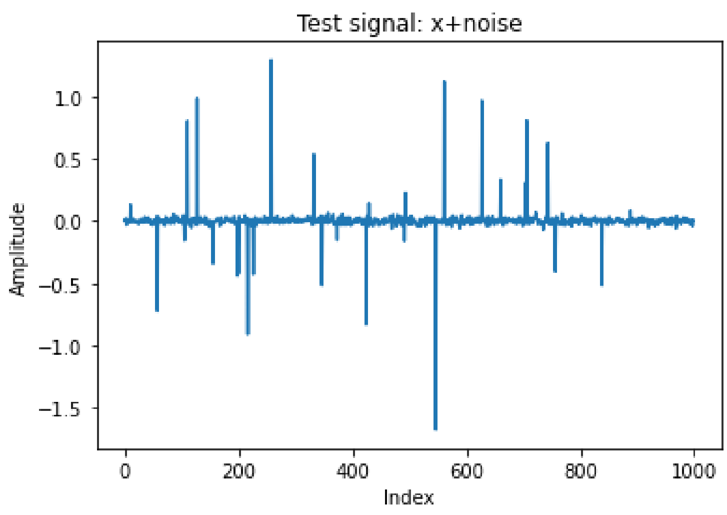

Figure 3.5: Test signal.

$$SNR = 15\text{dB} = 10\log_{10}\frac{\frac{1}{N}\sum_{i=1}^{N}x_i^2}{\sigma^2} \tag{3.64}$$

The final test signal obtained is represented in Fig. 3.5.

This test signal can be sensed with a Gaussian matrix obtaining $y = \Phi test$ with $M = 200$. Now we construct the projector $P_{\Omega^\perp}$ taking Ψ_I as the standard basis. The projection of y gives us a new vector $Py = P_{\Omega^\perp}y$. Finally, we reconstruct x_s with CoSaMP. To this end, the vector of measurements that we supply to the algorithm is Py and the sensing matrix is $P_{\Omega^\perp}\Phi$. The error obtained with this procedure is $\varepsilon = 0.4\%$. The recovery can be seen in Fig. 3.6.

Finally, we can vary the value of k_I from 10 to 90 at intervals of 10 and observe how the SNR of the reconstruction varies. For every point in Fig. 3.7, we have calculated the mean of the SNR value over 50 iterations. Note that as k_I increases the performance of the reconstruction method degrades. Therefore, the sparser the interference, the better the reconstruction of the signal of interest.

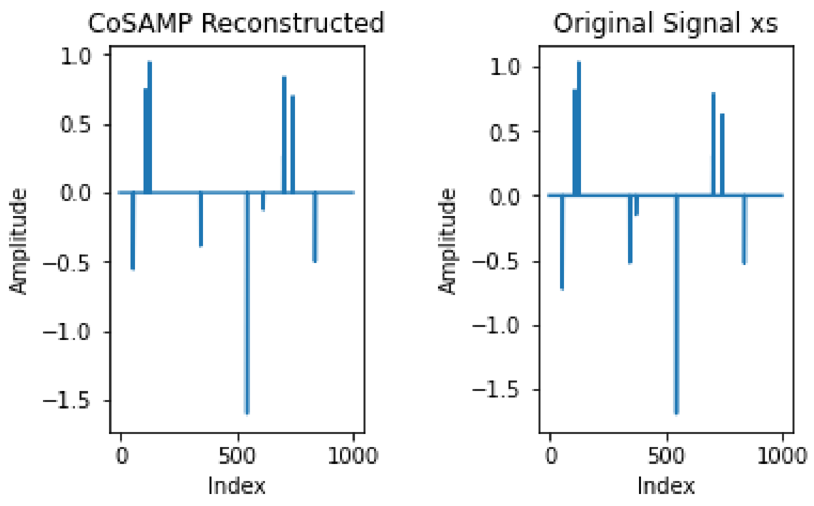

Figure 3.6: Recovery with CoSaMP using the so-called cancel-then-recover approach.

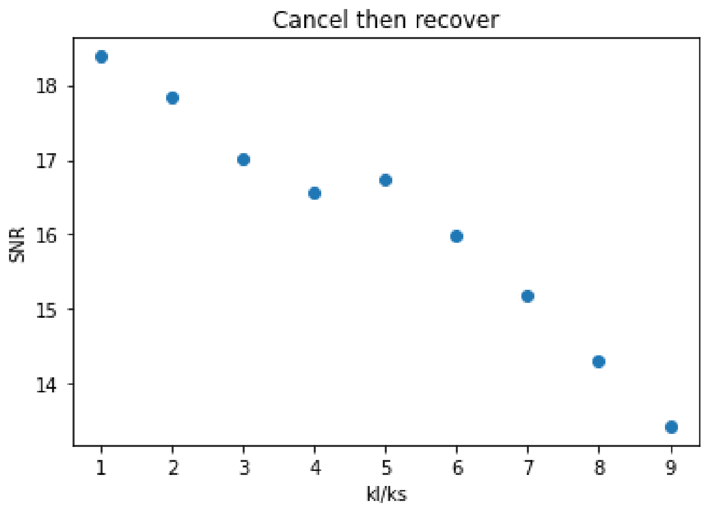

Figure 3.7: SNR vs. k_I/k_s using the so-called cancel-then-recover approach.

Chapter 4

CS-based inference: State of the art

The objective set in this thesis is to solve inference problems directly on compressed measurements. The theory developed in Chapter 3 is a classical proposal to obtain our goal. This is, through mathematics we can construct a solution to the main problems we may have to face. However, in practice, the majority of the work currently done in this field does not explicitly use mathematical background. Instead, machine learning is applied. In particular, Neural Networks (NNs) have turned out to be the fundamental tool to perform inference. If we pay attention to the dates of the related literature, we find that the statistical approach to inference on compressive measurements was developed around 2010. Just a couple of years later, the revolution of neural networks started.

4.1 Inference and Machine Learning

Machine learning is a field that aims to address problems that humans, as cognitive beings, are able to solve. The approach followed to achieve this goal is learning. In other words, programs must improve their results leveraging the experience of executing tasks repeatedly [52]. Within the field of machine learning, NNs have gained relevance in the recent years.

NNs simulate the functioning of the human brain. When input signals reach a neuron, a weighted sum of them is calculated. The output is generated only if the inputs are above a threshold. NNs are inspired by this process. They are composed of layers that contain several neurons. Each neuron generates a weighted sum of its inputs that comes from the previous layer. The output y of the j-th neuron in a layer is expressed as

$$y_j = f\left(\sum_{i=1} W_{ij}x_i + b_j\right) \tag{4.1}$$

where x_i are inputs, W_{ij} are weights, b_j is a bias, and $f(\cdot)$ is a non-linear function. Weights and biases are the flexible parameters of the network. If the outputs of all neurons in a layer are given by the weighted sum of all the inputs coming from the previous layer, then both layers are said to be fully connected (FC) layers. If this is not the case, then we have sparsely connected layers. Fig. 4.1 shows a representation of both types of connections.

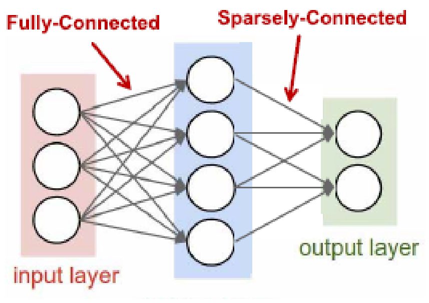

Figure 4.1: Scheme of fully and sparsely connected layers provided in [59].

If there are more than three layers in the network, then it is called a Deep Neural Network (DNN) [59].

To illustrate the use of DNNs let us suppose that we have images of various pets and we want to classify them automatically according to the type of animal. First, some images are supplied to the DNN together with the information of what animal category they belong to so that it can learn and adjust the weights for correct classification. This process is called training. Once this is done, a new image that has not been processed by the network yet is introduced. Using the knowledge previously acquired during training, the DNN will return a rank of probabilities associated with different possible categories of the animal in the new image. This process is called inference. The relationship with the problems stated in the previous chapter is direct. The DNN performs classification, which is an statistical inference problem.

One of the first achievements done by NNs was in 1989 when the LeNet network was developed for hand-written digit recognition [39]. However, it was not until 2010s that huge progress took place in this field. For example, a Microsoft's speech recognition system appeared in 2013 [26]. In fact, the progress in accuracy of DNNs in classification and detection tasks can be seen in the results of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [53]. The ImageNet project started in 2010. A publicly available database is provided to promote a yearly competition.

One of the challenges of the contest is image classification. The algorithms participating in the competition have to identify the class of a given image, that is, they must infer the image content. A total of 1.3 millions of images divided into 1000 classes and labeled with their corresponding class are provided for training. Then, a database with non-labeled images is supplied to test the accuracy of the algorithm proposed. In 2012, the so-called AlexNet DNN achieved an error reduction of 10% with respect to the previous year.

The results throughout the years are presented in Fig. 4.2. Specifically, the top-5 classification error of the best entries of the contest are shown. This top-5 error is based on whether the correct class of the test image is among the five classes that the algorithm chooses as the most probable for the processed image.

The use of DNNs involved a significant step forward in classification accuracy. AlexNet was just the beginning of the Deep-Learning era. In the following years, researchers developed deeper and more accurate networks. In 2015, a DNN called ResNet outperformed human ability for classification in the contest.

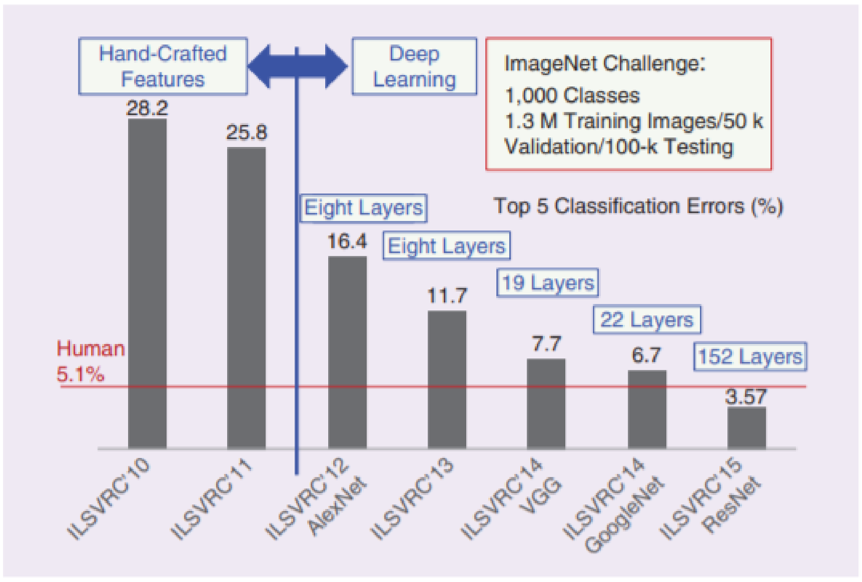

Figure 4.2: Classification errors of the best entries in ILSVRC throughout the years [63].

The conceptual mixture of CS and machine learning leads to a new framework often called Compressed Learning (CL). The original signals are said to be in the data domain. We can then perform a dimensionality reduction by sampling according to the framework of CS to enter the measurement domain. The objective of CL is to perform pattern recognition in the measurement domain.

An example of classification applying CL theory is [10], which employed soft-margin Supported Vector Machines (SVMs) (see Section 4.4.2 for further details on SVMs). In general terms, an SVM [7, 18] is a machine-learning algorithm that can be used for classification. It constructs a hyperplane as a boundary between the classes we want to distinguish. In [10], it was theoretically and experimentally demonstrated that a SVM classifier trained using compressed data performs almost as well as the best SVM classifier on the original data. Different approaches have been proposed over the years to implement this idea of performing inference directly on sampled data. However, with the DNN revolution, an increasing number of researchers have explored DNNs for CL.

4.2 Classification and recognition

Image classification and activity recognition are challenges that do not necessarily need a full-image reconstruction. Therefore, there has been an attempt to adapt conventional methods to this new scenario of inference on compressed measurements.

In [41], the authors proposed to address face recognition in the near-infrared spectrum using compressed measurements of the faces. Given that infrared cameras are expensive, alternative solutions must be found. In this study, an SPC (see application 1 in Section 2.5) was used to extract compressed images. Thus, they created their own database made up of compressed face images in the near-infrared.

Feature correlation usually refers to the dependence on different attributes of the analysed data. In [41], the proposal was based on keeping correlation using filters applied to the compressed data. They called them smashed filters. Once the feature extraction is done, a SVM performs a one-vs-all classification to identify faces. Therefore, the multi-class classification problem splits into a series of binary classifications, one per face, that are solved.

To improve the accuracy obtained using filters, a DNN designed to classify on compressed measurements was presented in 2015 [40]. The proposed scheme is the following:

1. Compressed measurements of a scene are obtained using an SPC. A measurement vector is obtained that can be mathematically expressed as $y = \Phi x$, where x represents the scene and Φ is the sensing matrix that corresponds to the pattern formed by the micromirrors of the camera.

2. A linear projection of the measurement vector is computed using Φ^T, thereby obtaining a pseudoimage: $\hat{x} = \Phi^T \Phi x$.

3. The pseudoimage is the input of a special type of network called Convolutional Neural Network (CNN) whose particular structure depends on the type of task addressed.

4. A last layer, called Softmax, is placed to output the probabilities of the original image belonging to each class.

CNNs are networks with layers that perform convolution. Every weighted sum is calculated using the same set of weights. Suppose that the input of the system is an image. In a convolutional layer, the set of weights are placed in a 2D array called filter or kernel whose dimension is smaller than the input, which is also a 2D array. This filter is applied to a section of the input with the same size as the filter. The result is called an output activation. The filter moves across the input producing more outputs, and the collection of all of them is called feature map (fmap). In [59], this process is illustrated as shown in Fig. 4.3.

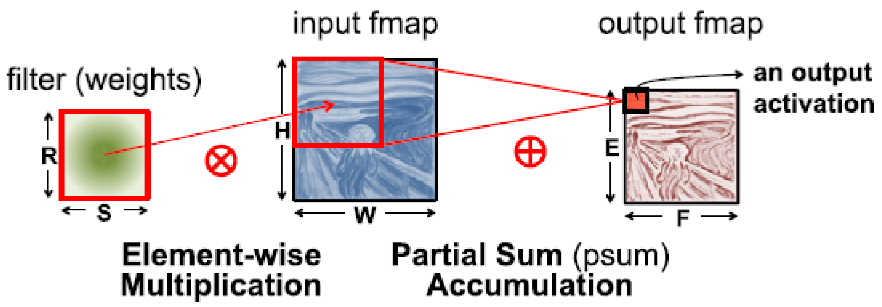

Figure 4.3: Scheme of a 2D convolution provided in [59].

If the input image has colour, then it will have different channels and the input size would be $H \times W \times C$, where H stands for height, W for width, and C for channels. In this case, we would use a stack of C 2D filters. Usually, this is called a 3D filter.

The experiments presented by [40] do not use the SPC physically but instead they simulate its mechanism, i.e., the implementation of $y = \Phi x$, on software. The results are based on two experiments that use the scheme described above.

In the first experiment, the objective was classification. The database used is MNIST, which is a collection of 28×28 images in grayscale containing hand-written digits from 0 to 9. Therefore, it has 10 different classes. MNIST comprises a total of 70000 images, among which 60000 are used for training and the rest for testing. The network developed to classify these images is based on LeNet-5 and the sensing matrix chosen is a Gaussian matrix. The network is trained and tested for different compression ratios M/N with $N = 28 \times 28 = 784$. Finally, the results are compared to the ones achieved with the smashed filters proposed in [41].

According to the experimental data reported in [40], Table 4.1 compares the results of [40] and [41]. Clearly, the DNN approach achieves better performance.

Compression ratio (M/N)	Test error Smashed Filters	Test error CNN approach
1	13,86%	0,89%
0,25	27,42%	1,63%
0,10	43,55%	2,99%
0,05	53,21%	5,18%
0,01	63,03%	41,06%

Table 4.1: Comparison of test error for MNIST classification with different approaches [40].

In the second experiment, a complex database was chosen to test the proposed framework. The authors worked with the ILSVRC 2012 dataset. The same aforementioned scheme was followed. However, in this case, the sensing matrix is a Hadamard matrix and the CNN is a modified version of AlexNet.

Based on this publication, other researchers contributed to the improvement of classification using compressed measurements. For example, [1] and [66] tested new networks capable of learning the sensing matrix before performing inference. They added a first layer, whose parameters are learned, to act as the sensing matrix Φ. A second layer learns a new set of weights and projects the input measurements as Φ^T in [40]. However, this second matrix $\hat{\Psi}$ is a different matrix.

Following these advances, [24] incorporated the learned-sensing-matrix approach and also extracted both features with a DNN and hand-crafted features using DCT coefficients. Classification was conducted on the fused features and MNIST was used to test the proposed network. The results were compared to the ones obtained using [41], [40], and [1]. A significant improvement in accuracy was achieved with respect to the smashed filter proposal and similar test errors to DNNs in [40] and [1] were obtained.

In [4], image classification based on learning the sensing matrix was also proposed. This sensing matrix is to be employed as the matrix for an SPC. There are many other proposals for image classification using DNNs. For example, [42] develops a system inspired in the functioning of the human retina to sense the image in an efficient way and classify with good results. Also, some researches have put together classification and an interactive interface. In [58], a DNN was developed that classifies compressed measurements and can be tested through an interactive interface where one can write a digit, which is compressed and classified. Words can also be written and the letters recognized are sent to a spelling corrector.

Finally, human-action recognition is addressed in [45]. A human action such as walking or running can be registered through the pertinent sensor. The aim is to distinguish between different activities. A sensing-matrix design is proposed in [45] to apply CS measurement acquisition and subsequent classification. To test their proposal, the authors used the so-called CMU Mocap database, which contains images of skeletons in motion. They selected 6 classes to classify: walk, run, walk with arms out, walk swinging, walk with wild legs, and walk on toe.

4.3 Detection and estimation

Detection is an inference problem required in a variety of applications. For example, to ensure security in roads or shopping centers, we may need to be able to detect and track people. Detection can be also useful for certain medical applications to improve the diagnostics tools available.

In [38], the authors propose to perform human tracking on compressed measurements using DNNs. Instead of taking linear measurements, they choose a special case of compressed samples. They use subsampling, which implies retaining the value of random pixels and discarding the rest. The database used to test their proposal is SENSIAC. It contains videos of people walking in a background at different distances. In particular, human tracking was tested at 500 m, 1000 m, and 1500 m. The idea is to perform tracking by detection. This is, the person is detected in the video at different times and therefore, a trajectory is formed.

Leaving aside computer-vision applications, there are other fields that can benefit from performing inference with compressed measurements. One of these areas is health care, [16, 20]. There is an increased usage of wireless body sensors to keep control of various pathologies. By detecting a change in physiological signals or factors indicating a disease, early intervention can be done. If the detection was achieved on compressed measurements, less energy would be required by the wearable device.

Moreover, if we used the traditional CS paradigm, after sampling, we would have to transmit the measurements and reconstruct the signals at the receiver point. This could mean losing too much time during the recovery and consequently missing, for instance, a disease indicator that appears during a brief period of time. This is the case of atrial fibrillation.

Detection of atrial fibrillation, which is a type of arrhythmia that can indicate thrombosis or strokes, was investigated in [16]. This fibrillation appears in electrocardiograms (ECG). With Fig. 4.4, the authors showed the difference between a normal ECG signal and one with atrial fibrillation.

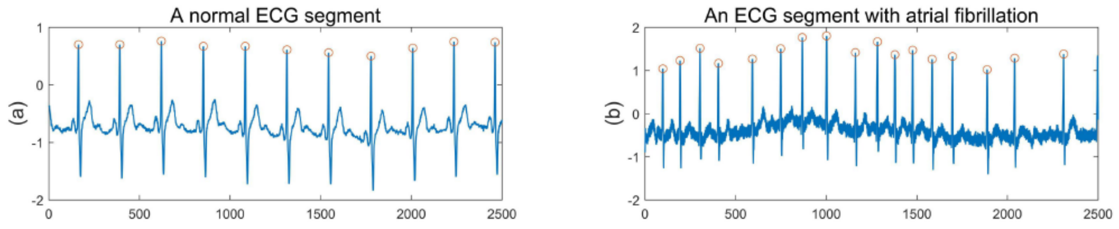

Figure 4.4: ECG without and with atrial fibrillation extracted from [16].

Their aim was to detect atrial fibrillation in compressed measurements of an electrocardiogram (ECG) using a DNN developed ad-hoc for this purpose. Similar to the implementations for classification, the first layer of the network projects the compressed measurements. However, instead of using Φ^T, they used the pseudo-inverse of the sensing matrix to initialise the weights of this layer, i.e., $\Phi^\dagger$. With training, these weights, along with the ones of the rest of the network, are optimized to maximize the detection accuracy.

Up to now, we have focused on applications using machine learning. However, it is possible to implement inference using the classical techniques seen in Chapter 3. Following with health care applications, not only detection is needed. Estimation can also be useful. In [20], the heart rate is estimated in the compressed domain using the estimators developed in Chapter 3. With an ECG, heart beats can be measured over a period of time and the heart rate is determined by certain peaks called R-peaks, as shown on the right plot in Fig. 4.4. The idea is to perform matched filtering in the compressed data and apply a threshold to localize these peaks.

The ECG signal $x(t)$ is a continuous signal in time and $\psi(t)$ is a template of an R-peak. Matched filtering consists in convolving the ECG signal, $x(t)$, with a time-reverse function of $\psi(t)$:

$$R(t) = \int_{-\infty}^{\infty} x(\tau)\psi(\tau - t)d\tau \tag{4.2}$$

With the resulting R, we will be able to detect the R-peaks that are placed where maxima appear in the output.

If we consider x as a discrete signal of length N, then matched filtering consists in calculating a vector R whose components are $R_n = \langle x, \psi_n \rangle$ with $n \in \{1, ..., N\}$; ψ_n is appropriately defined and contains the template used.

Now suppose that measurements are taken and we have $y = \Phi x \in \mathbb{R}^M$. With the estimators analysed in Chapter 3, we can estimate the value of those inner products. For example, in the case of the direct estimator, we would have $\hat{R}_n = \langle y, \Phi \psi_n \rangle$.

4.4 Implementation

In this section, we describe how we tested some implementations of inference problems using compressed measurements. We focus on detection and classification as they are the tasks typically addressed by the research community.

4.4.1 MNIST database

As mentioned above, the MNIST database consists on 70000 28×28 grayscale images of hand-written digits from 0 to 9; 60000 of these images are reserved for training and the remaining 10000 are for testing the targeted machine-learning algorithm. It is a simple and widely used database that is accessible through Tensorflow.keras.

```
import tensorflow as tf
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
```

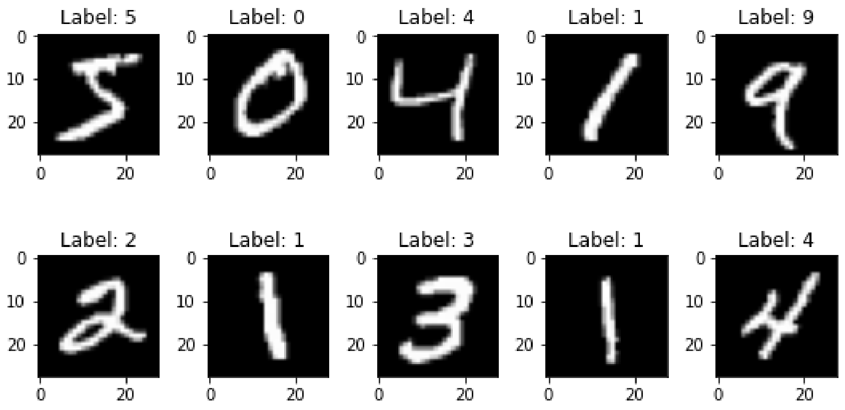

Figure 4.5: Examples of MNIST images with their labels.

Fig. 4.5 shows the first 10 images of the training section of the database. As we can see from the code above, each image comes with a label that indicates the class it belongs to. This is important for the training process, because the algorithm needs to know the class of every image to learn.

Given that each image is 28×28, when they are transformed into a vector, we will have a vector of length $N = 28 \cdot 28 = 784$. In the following subsections, we describe how to detect and classify on MNIST.

4.4.2 Detection with SVM

As previously explained in this chapter, SVMs are algorithms used for classification that rely on the construction of an hyperplane to separate classes [9]. This means that we can use SVMs to perform detection by dividing the data into classes. For binary classification, one of the classes represents the target category.

Suppose a dataset for training, $\{x_i, y_i\}_{i=1}^{l}$, where $x_i \in \mathbb{R}^N$ encodes the data and y_i represents the corresponding label indicating the class of x_i. We consider two possibilities for the labels, namely $y_i \in \{1, -1\}$. Suppose that the points $\{x_i\}_{i=1}^{l}$ are linearly separable, i.e., there exists at least one hyperplane that leaves one class of the points on a side and the other class on the other. In particular, let us suppose the hyperplane

$$\mathbf{w} \cdot x + b = 0 \tag{4.3}$$

where $\mathbf{w} \in \mathbb{R}^N$ is normal to the hyperplane and $b \in \mathbb{R}$. If this hyperplane separates both classes of data completely, we would have the followings inequalities:

$$\mathbf{w} \cdot x_i + b \geq 1 \text{ for } y_i = 1 \tag{4.4}$$

$$\mathbf{w} \cdot x_i + b \leq -1 \text{ for } y_i = -1 \tag{4.5}$$

The margins of the hyperplane, d_+ and d_-, are defined as the minimum distance between the proper hyperplane and the nearest data points on each side, called supported vectors. These are the closest points with $y = 1$ on one side of the hyperplane and the closest points with $y = -1$ on the other side. The end of these margins are given by taking equalities in Eqs. (4.4) and (4.5); the distance in between is $2/\|\mathbf{w}\|_2$. Therefore, we can maximize the margins by minimizing $\|\mathbf{w}\|_2^2$ subject to Eqs. (4.4) and (4.5). The SVM tries to find the hyperplane with maximum margins. Fig. 4.6 represents this hyperplane construction together with the supported vectors in a 2D space.

Now, let us assume that not all the points can be separated as indicated above. In this case, we could relax the model and even let the SVM misclassify some samples of the data. Thus, some points could fall into the margin regions or on the wrong side of

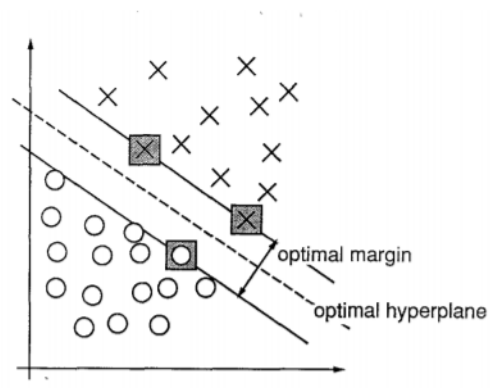

Figure 4.6: Example of linear separation of 2D data extracted from [18]. Supported vectors are marked with grey squares.

the hyperplane. To make sure the situation is controlled and the error committed is not critical, a new parameter, usually called C, is introduced to penalize points improperly placed. This algorithm is called soft-margin SVM. Finally, if the data considered are not linearly separable, then every x_i is mapped to a higher dimension where it is separable through an application called kernel. In this case, the algorithm is called non-linear classifier.

The code provided in [34] implements an SVM that detects the digit 3 in images extracted from MNIST. First, ns images are selected from the database for training, half of them are digit 3 whereas the rest are other digits. In the same way, nt images are selected for testing, again half of them are digit 3. Once this is done, a linear supported vector classifier is fed with the training set without compressing the images. The results for detection in the data domain present an accuracy of 88%.

The next step is to obtain the accuracy in the measurement domain. We set different values of measurements M. For each of them, we set a loop of 20 iterations to calculate the mean accuracy obtained over them. In each iteration, Gaussian and Bernoulli matrices of size $M \times N$ were generated, where $N = 784$ (see Section 4.4.1). The definition of these matrices is provided in Definition 8. Detection is performed with each of the matrices to compare their results.

Fig. 4.7 shows that in general the accuracy obtained is similar to the one in the data domain. However, for small values of M there is an abrupt descent, which may imply that those compression rates are not suitable for this implementation of detection in the measurement domain. Note also that the performance of the Gaussian and the Bernoulli matrices are very similar, although for small values of M there seems to occur a small drop in accuracy for the Gaussian case with respect to the Bernoulli matrix.

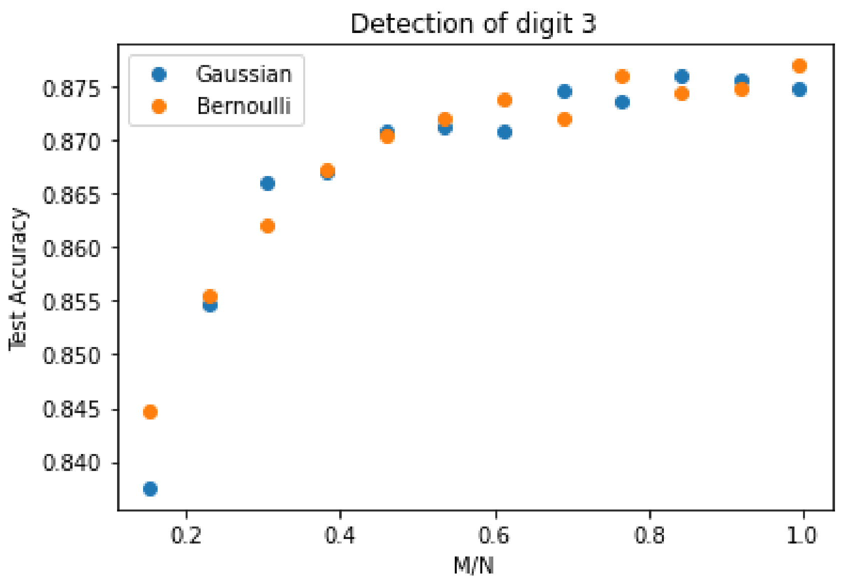

Figure 4.7: Detection accuracy vs. M/N for Gaussian and Bernoulli matrices.

4.4.3 Classification with DNN

In this subsection, we show how to construct a network to classify on MNIST using the approach reported in [40]. In this approach, as previously explained in this chapter, every image of the database is compressed with the sensing matrix Φ to simulate the measurements that an SPC would return, and then they are projected with Φ^T. Therefore, our first step is to create a function to pre-process the complete dataset that returns pseudoimages to be used as inputs of the DNN.

The DNN implemented has been created according to the one proposed in [40], which is based on the LeNet model. The scheme of the network is the following:

1. Convolutional layer: 20 output feature maps, kernels of size 5×5, and Rectified Non-linear Unit (ReLU) as the nonlinear function.

   ReLU is defined as $f(x) = \max(0, x)$.

2. Max pooling layer: size 2×2 and stride =2.

   Input windows of the size indicated are reduced or pooled by substituting it by the maximum of the values in the window. The stride indicates the number of movements in the horizontal or vertical direction to perform another pooling.

3. Convolutional layer: 50 output feature maps, kernels of size 5×5 and ReLU.

4. Max pooling layer: size 2×2 and stride $= 2$.

5. Fully connected layer: output feature vector of length 50.

6. Fully connected layer: output feature vector of length 10 and softmax.

To train our DNN, we need an optimizer. This is an algorithm that updates weights to reduce the loss and obtain better accuracy results. In this example, we have used Adam optimizer [37], training the network for 10 epochs. On each epoch, the DNN is fed with the training images and learns from them. We divided the images of the training set between training and validation images. Once the training is done, the validation images are used to calculate the accuracy of the network. For $M/N = 0.3$, we obtained an 87% of accuracy. Finally, to illustrate the performance of our network, we randomly selected 10 images from the test set and compared the correct label and the predicted class for $M/N = 0.3$, as shown in Table 4.2. We can see that only the first image was misclassified.

Correct label	6	7	0	5	7	8	1	6	8	2
Predicted class	8	7	0	5	7	8	1	6	8	2

Table 4.2: Classification results on MNIST.

If we train the network for different values of M/N, we obtain the plot depicted in Fig. 4.8. It is clear that for a wide range of values of M, the results are promising. However, for small values of M, the accuracy drops rapidly.

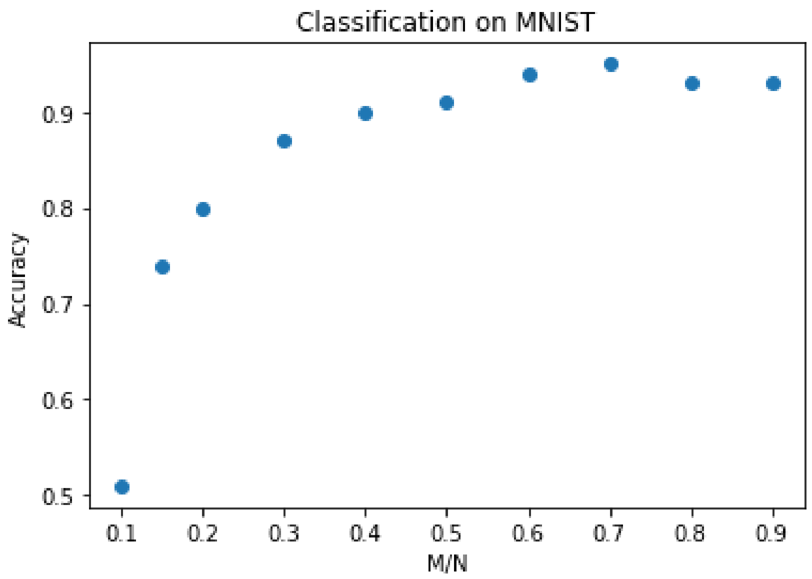

Figure 4.8: Accuracy of classification vs. M/N for Gaussian matrices.

Chapter 5

Conclusions

We began this work talking about the technological revolution that is generating large amounts of data every day. In turn, this is giving rise to the problem of efficiently transmitting and storing such vast amount of information. In this context, CS emerged as a new theoretical framework able to improve classical approaches with the characteristic of concurrent sampling and compression. This new theory was developed for sparse or compressible signals. Although this may seem a limitation of CS, signals can be transformed to a new domain in which they have the necessary properties to be used.

Initially, CS was developed for signal reconstruction. Thus, a signal of interest is sampled and compressed with a sensing matrix, and the objective is to reconstruct the original signal as faithfully as possible from the measurements taken. However, it is not always useful to reconstruct the complete signal or even try reconstruction at all. Furthermore, if we are able to recover the signal from the compressed measurements is because all the relevant information was already there. Therefore, we could perform inference directly on compressed measurements.

We have studied the mathematical theory of four inference problems: detection, classification, estimation, and filtering. Concerning the first two problems, they were addressed using hypothesis testing, which is a fundamental branch of statistical inference. The experimental results for detection rate and classification error illustrate that it is possible to perform these tasks on compressed measurements with good results. However, for small values of M/N, the error increases rapidly, meaning that high compression are not suitable. Continuing with estimation, we have studied two different estimators with similar performance. Finally, filtering an unwanted signal in the compressed domain enables reconstructing a signal of interest from a noisy signal. The *cancel-then-recover* technique implies performing filtering in the compressed domain to eliminate the interference and then reconstructing the signal of interest.

When trying to implement inference on CS measurements, researchers have explored a machine-learning perspective. The fusion of these two fields is called CL. In particular, after the NN revolution, many proposed approaches to leverage DNNs and many applications have been developed. As examples, in Chapter 4 we have reviewed some of these applications. Classification on compressed images has been widely studied and different approaches have been developed to improve previous results. Face recognition has also been addressed with CL. Detection on compressed measurements has been used for applications as different as human tracking and disease detection. However, not all inference applications use machine learning. We have also found an example that uses estimators to estimate the heart rate.

Finally, we show experimental results of detection and classification on compressed measurements as they are the principal problems addressed by researchers. We have tested a code implementing detection on compressed measurements using an SVM and two different types of random matrices. The results show that the accuracy is very similar to the one obtained for non-compressed data even for few measurements, and the choice between Gaussian or Bernoulli matrices does not make much difference in accuracy results. Concerning classification, we have conducted experiments on the basis of a code developed according to the proposal in [40]. The results show that for a wide range of compression ratios, classification on MNIST with good accuracy is achieved. However, for very small values of M, the rapid decrease in accuracy makes accuracy unacceptable for classification with this code.

In conclusion, inference on compressed measurements by means of CS as a sampling and compression strategy is possible. We have studied four inference problems and reviewed applications of this new theoretical framework mainly using machine learning. Moreover, we have shown experimental results of detection and classification obtaining good accuracy for many compression ratios. However, it has also been observed a rapid decrease in accuracy when the number of measurements is small.

Appendix

This appendix elaborates on the code used in Chapters 2, 3, and 4. The files are available in a Github repository that can be accessed in `https://github.com/MarJimCom/Inference_on_compressive_measurements`.

Based on [54], we developed the code in *Sparsity_and_reconstruction.py* that is employed in Subsection 2.4.1. The file *Non_sparse_signals.py*, based in [60, 19], corresponds to Subsection 2.4.2. To end with the code used in Chapter 2, *Image_reconstruction.py* contains the code from [60] used for image reconstruction and the calculation of our errors.

Following with Chapter 3, the plots depicted are produced by the codes in *Detection.py*, *Classification.py*, *Estimation.py* and *Filtering.py*.

Detection_with_SVM.py corresponds to the results provided in Section 4.4.2. The complete implementation can be found in [34]. We include the changes we introduced to obtain Fig. 4.7. Finally, *Classification_with_DNN.py* contains the DNN based on the proposal reported in [40] and used to create the plot in Fig. 4.8 in Section 4.4.3.

Bibliography

[1] Adler, A., Elad, M., and Zibulevsky, M. Compressed Learning: A Deep Neural Network Approach. *arXiv:1610.09615 [cs]* (Oct. 2016).

[2] Ahmed, N., Natarajan, T., and Rao, K. Discrete Cosine Transform. *IEEE Transactions on Computers C-23*, 1 (Jan. 1974), 90–93.

[3] Andrew, G., and Gao, J. Scalable training of L^1 -regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning - ICML '07* (Corvalis, Oregon, 2007), ACM Press, pp. 33–40.

[4] Bacca, J., Correa, C. V., Vargas, E., Castillo, S., and Arguello, H. Compressive Classification from Single Pixel Measurements Via Deep Learning. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)* (Pittsburgh, PA, USA, Oct. 2019), IEEE, pp. 1–6.

[5] Baraniuk, R., and Steeghs, P. Compressive Radar Imaging. In *2007 IEEE Radar Conference* (Apr. 2007), pp. 128–133. ISSN: 2375-5318.

[6] Blumensath, T., and Davies, M. E. Iterative Hard Thresholding for Compressed Sensing. *arXiv:0805.0510 [cs, math]* (May 2008).

[7] Boser, B. E., Guyon, I. M., and Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (Pittsburgh, Pennsylvania, USA, July 1992), COLT '92, Association for Computing Machinery, pp. 144–152.

[8] Boutell, T. PNG (Portable Network Graphics) Specification Version 1.0. Tech. Rep. RFC 2083, RFC Editor, Mar. 1997.

[9] Burges, C. J. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery 2*, 2 (June 1998), 121–167.

[10] Calderbank, R., and Jafarpour, S. Finding needles in compressed haystacks. In *Compressed Sensing: Theory and Applications*, G. Kutyniok and Y. C. Eldar, Eds. Cambridge University Press, Cambridge, 2012, pp. 439–484.

[11] Candes, E., Romberg, J., and Tao, T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory 52*, 2 (Feb. 2006), 489–509.

[12] Candes, E., and Wakin, M. An Introduction To Compressive Sampling. *IEEE Signal Processing Magazine 25*, 2 (Mar. 2008), 21–30.

[13] Chartrand, R. Exact Reconstruction of Sparse Signals via Nonconvex Minimization. *IEEE Signal Processing Letters 14*, 10 (Oct. 2007), 707–710.

[14] Chartrand, R., and Staneva, V. Restricted isometry properties and nonconvex compressive sensing. *Inverse Problems 24*, 3 (June 2008), 035020.

[15] Chen, S. S., Donoho, D. L., and Saunders, M. A. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing 20*, 1 (Jan. 1998), 33–61.

[16] Cheng, Y., Hu, Y., Hou, M., Pan, T., He, W., and Ye, Y. Atrial Fibrillation Detection Directly from Compressed ECG with the Prior of Measurement Matrix. *Information 11*, 9 (Sept. 2020), 436.

[17] Cormode, G., and Muthukrishnan, S. Combinatorial Algorithms for Compressed Sensing. In *2006 40th Annual Conference on Information Sciences and Systems* (Princeton, NJ, Mar. 2006), IEEE, pp. 198–201.

[18] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20*, 3 (Sept. 1995), 273–297.

[19] Cox, W. A Brief Introduction to Compressed Sensing with Scikit-Learn. http://www.gallamine.com/2014/02/a-brief-introduction-to-compressed.html, Feb. 2014.

[20] Da Poian, G., Rozell, C. J., Bernardini, R., Rinaldo, R., and Clifford, G. D. Matched Filtering for Heart Rate Estimation on Compressive Sensing ECG Measurements. *IEEE Transactions on Biomedical Engineering 65*, 6 (June 2018), 1349–1358.

[21] Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *arXiv:math/0307152* (Nov. 2003).

[22] Davenport, M., Boufounos, P., Wakin, M., and Baraniuk, R. Signal Processing With Compressive Measurements. *IEEE Journal of Selected Topics in Signal Processing 4*, 2 (Apr. 2010), 445–460.

[23] Davenport, M. A., Duarte, M. F., Eldar, Y. C., and Kutyniok, G. Introduction to compressed sensing. In *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, Cambridge, 2012, pp. 1–64.

[24] Degerli, A., Aslan, S., Yamac, M., Sankur, B., and Gabbouj, M. Compressively Sensed Image Recognition. In *2018 7th European Workshop on Visual Information Processing (EUVIP)* (Tampere, Nov. 2018), IEEE, pp. 1–6.

[25] Deka, B., and Datta, S. *Compressed sensing magnetic resonance image reconstruction algorithms.* Springer Berlin Heidelberg, New York, NY, 2019.

[26] Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., Gong, Y., and Acero, A. Recent advances in deep learning for speech research at Microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (May 2013), pp. 8604–8608. ISSN: 2379-190X.

[27] Dong, B., Mao, Y., Osher, S., and Yin, W. Fast linearized Bregman iteration for compressive sensing and sparse denoising. *Communications in Mathematical Sciences 8*, 1 (2010), 93–111.

[28] Donoho, D. Compressed sensing. *IEEE Transactions on Information Theory 52*, 4 (Apr. 2006), 1289–1306.

[29] Duarte, M., Davenport, M., Wakin, M., and Baraniuk, R. Sparse Signal Detection from Incoherent Projections. In *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings* (Toulouse, France, 2006), vol. 3, IEEE, pp. III–305–III–308.

[30] Duarte, M. F., Davenport, M. A., Takhar, D., Laska, J. N., Sun, T., Kelly, K. F., and Baraniuk, R. G. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine 25*, 2 (Mar. 2008), 83–91.

[31] Foucart, S., and Rauhut, H. *A Mathematical Introduction to Compressive Sensing.* Applied and Numerical Harmonic Analysis. Springer New York, New York, NY, 2013.

[32] Gilbert, A. C., Strauss, M. J., Tropp, J. A., and Vershynin, R. Sublinear approximation of signals. R. A. Athale and J. C. Zolper, Eds., p. 623206.

[33] Gilbert, A. C., Strauss, M. J., Tropp, J. A., and Vershynin, R. One sketch for all: fast algorithms for compressed sensing. In *Proceedings of the thirty-*

ninth annual ACM symposium on Theory of computing - STOC '07 (San Diego, California, USA, 2007), ACM Press, p. 237.

[34] Gupta, Y., and Agarwal, P. Compressed learning using svms. `https://github.com/pratyush1019/Compressed-Learning`, 2020.

[35] Janse van Rensburg, C. *Big data : a compressed sensing approach.* Dissertation, University of Pretoria, 2017.

[36] Kaur, R., and Choudhary, P. A Review of Image Compression Techniques. *International Journal of Computer Applications 142*, 1 (May 2016), 8–11.

[37] Kingma, D. P., and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Jan. 2017).

[38] Kwan, C., Gribben, D., and Tran, T. Multiple Human Objects Tracking and Classification Directly in Compressive Measurement Domain for Long Range Infrared Videos. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (New York City, NY, USA, Oct. 2019), IEEE, pp. 0469–0475.

[39] Le Cun, Y., Jackel, L., Boser, B., Denker, J., Graf, H., Guyon, I., Henderson, D., Howard, R., and Hubbard, W. Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine 27*, 11 (Nov. 1989), 41–46.

[40] Lohit, S., Kulkarni, K., and Turaga, P. Direct inference on compressive measurements using convolutional neural networks. In *2016 IEEE International Conference on Image Processing (ICIP)* (Phoenix, AZ, USA, Sept. 2016), IEEE, pp. 1913–1917.

[41] Lohit, S., Kulkarni, K., Turaga, P., Wang, J., and Sankaranarayanan, A. C. Reconstruction-free inference on compressive measurements. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (Boston, MA, USA, June 2015), IEEE, pp. 16–24.

[42] Lubana, E. S., Aggarwal, V., and Dick, R. P. Machine Foveation: An Application-Aware Compressive Sensing Framework. In *2019 Data Compression Conference (DCC)* (Snowbird, UT, USA, Mar. 2019), IEEE, pp. 478–487.

[43] Luo, C., Wu, F., Sun, J., and Chen, C. W. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09* (Beijing, China, 2009), ACM Press, p. 145.

[44] Lustig, M., Donoho, D., Santos, J., and Pauly, J. Compressed Sensing MRI. *IEEE Signal Processing Magazine 25*, 2 (Mar. 2008), 72–82.

[45] Ma, R., Liu, G., Hao, Q., and Wang, C. Design of compressive imaging masks for human activity perception based on binary convolutional neural network. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (Daegu, Nov. 2017), IEEE, pp. 260–265.

[46] Mallat, S., and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41*, 12 (Dec. 1993), 3397–3415.

[47] Needell, D., and Tropp, J. A. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *arXiv:0803.2392 [cs, math]* (Apr. 2008).

[48] Pati, Y., Rezaiifar, R., and Krishnaprasad, P. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers* (Pacific Grove, CA, USA, 1993), IEEE Comput. Soc. Press, pp. 40–44.

[49] Qaisar, S., Bilal, R. M., Iqbal, W., Naureen, M., and Lee, S. Compressive sensing: From theory to applications, a survey. *Journal of Communications and Networks 15*, 5 (Oct. 2013), 443–456.

[50] Raid, A. M., Khedr, W. M., El-dosuky, M. A., and Ahmed, W. Jpeg Image Compression Using Discrete Cosine Transform - A Survey. *arXiv:1405.6147 [cs]* (May 2014).

[51] Rani, M., Dhok, S. B., and Deshmukh, R. B. A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications. *IEEE Access 6* (2018), 4875–4894. Conference Name: IEEE Access.

[52] Ray, S. A Quick Review of Machine Learning Algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMIT-Con)* (Faridabad, India, Feb. 2019), IEEE, pp. 35–39.

[53] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision 115*, 3 (Dec. 2015), 211–252.

[54] Schnoor, E. GitHub - ekki25/SparseRecovery: Implementation of different compressive sensing sparse recovery algorithms. `https://github.com/ekki25/SparseRecovery`, 2019.

[55] Scott, C. Statistical Signal Processing. https://cnx.org/contents/68ebe763-38ea-436e-9fbd-c8abd71bae60@1.9, Dec. 2013.

[56] Shannon, C. Communication in the Presence of Noise. *Proceedings of the IRE 37*, 1 (Jan. 1949), 10–21.

[57] Spagnolini, U. *Statistical signal processing in engineering.* John Wiley & Sons, Hoboken, NJ, 2017.

[58] Stubbs, J. J., Pattichis, M. S., and Birch, G. C. Interactive image and video classification using compressively sensed images. In *2017 51st Asilomar Conference on Signals, Systems, and Computers* (Pacific Grove, CA, Oct. 2017), IEEE, pp. 2038–2041.

[59] Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE 105*, 12 (Dec. 2017), 2295–2329.

[60] Taylor, R. Compressed Sensing in Python. `http://www.pyrunner.com/weblog/B/`, 2016.

[61] Tibshirani, R. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288.

[62] Vélez Ibarrola, R., and García Pérez, A. *Principios de inferencia estadística.* Universidad Nacional de Educación a Distancia, Madrid, 2012. OCLC: 828357736.

[63] Verhelst, M., and Moons, B. Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices. *IEEE Solid-State Circuits Magazine 9*, 4 (2017), 55–65.

[64] Walden, R. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications 17*, 4 (Apr. 1999), 539–550.

[65] Xie, K., He, Z., and Cichocki, A. Convergence Analysis of the FOCUSS Algorithm. *IEEE Transactions on Neural Networks and Learning Systems 26*, 3 (Mar. 2015), 601–613.

[66] Zisselman, E., Adler, A., and Elad, M. Compressed Learning for Image Classification: A Deep Neural Network Approach. In *Handbook of Numerical Analysis*, vol. 19. Elsevier, 2018, pp. 3–17.

[67] Zuev, K. Statistical Inference. *arXiv:1603.04929 [math, stat]* (Mar. 2016).