Five Lectures on Sparse and Redundant Representations Modelling of Images

Michael Elad

IAS/Park City Mathematics Series
Volume 19, 2010

Five Lectures on Sparse and Redundant Representations Modelling of Images

Michael Elad

Preface

The field of sparse and redundant representations has evolved tremendously over the last decade or so. In this short series of lectures, we intend to cover this progress through its various aspects: theoretical claims, numerical problems (and solutions), and lastly, applications, focusing on tasks in image processing.

Our story begins with a search for a proper model, and of ways to use it. We are surrounded by huge masses of data, coming from various sources, such as voice signals, still images, radar images, CT images, traffic data and heart signals just to name a few. Despite their diversity, these sources of data all have something in common: each and everyone of these signals has some kind of internal structure, which we wish to exploit for processing it better.

Consider the simplest problem of removing noise from data. Given a set of noisy samples (e.g., value as a function of time), we wish to recover the true (noise-free) samples with as high accuracy as possible. Suppose that we know the exact statistical characteristics of the noise. Could we perform denoising of the given data? The answer is negative: Without knowing anything about the properties of the signal in mind, this is an impossible task. However, if we could assume, for example, that the signal is piecewise-linear, the noise removal task becomes feasible. The same can be said for almost any data processing problem, such as de-blurring, super-resolution, inpainting, compression, anomaly detection, detection, recognition, and more - relying on a good model is in the base of any useful solution.

When approaching a problem in image processing, there is a wide range of models to choose from, such as Principal Component Analysis (PCA), Anisotropic diffusions, Markov Random Fields, Total-Variation, Besov spaces and many more. There are two important (and sometime somewhat contradictory) requirements for a model to be useful. On one hand, it should be reliable, in the sense that it should be able to represent the data well. On the other hand, it has to be simple enough to be applied in practice to the various tasks at hand.

A simple example of a model used in a common task can be found in the JPEG algorithm. At its core, this algorithm relies on the claim that when transforming patches of 8×8 pixels in a typical image using the Discrete Cosine Transform (DCT), only the top-left (low frequency) components are dominant, while the rest are close

The Computer Science Department, the Technion, Haifa 32000, Israel
E-mail address: elad@cs.technion.ac.il
The author wishes to thank Dr. Matan Protter for the extensive help in writing these lecture-notes.

to zero. This model is used for compression by discarding the non-dominant components, thus using much fewer numbers for representing the image. While simple and relatively effective, one immediately sees potential failures of this model, and ways to further improve it.

We therefore turn our attention to the model of "Sparse and Redundant Representations" and example-based methods, which will be the focus of this booklet (and corresponding course). This model has many "fathers", drawing from various areas of research, such as signal processing (Wavelet theory, multi-scale analysis, signal transforms), machine learning, Mathematics (approximation theory, linear algebra, optimization theory) and more. We will see that it can be applied to various applications, such as blind source separation, compression, denoising, inpainting, demosaicing, super-resolution and more.

What is this model all about? Let us focus on the task of representing signals, which are 8×8 patches of images, as done above. We shall assume that a *dictionary* of such image patches is given to us, containing 256 *atom-images*, each of which is also 8×8. The *Sparseland* model assumption is that *every* image patch from the family of interest could be constructed as a linear combination of *few* atoms from the dictionary.

To represent a signal of length 64, this model uses 256 numbers, and therefore it is redundant. However, of these 256 all but few are zero, and therefore the model is sparse. Effectively, if there are 3 atoms involved in the found combination, only 6 numbers are needed to accurately represent the signal using this model – the three locations of the non-zeros, and their values.

While this model sounds near-trivial, it raises several very difficult questions. Given an image patch, can we find what atoms it was built from? Given a set of patches, how should we get the dictionary for them? Is this model applicable to various signals? The first question is problematic, as crunching the numbers indicates that a search through every possible combination is unlikely to finish any time soon, even for low-dimensional problems. However, several approximation algorithms have been proposed, with surprising guarantees on their performance.

The first two lectures will focus on such algorithms and their theoretical analysis. The third lecture will discuss how this model and theoretical foundations are brought to the realm of image processing. The second question can be addressed by learning the dictionary from a large collection of signals. Such algorithms have been developed in the past several years, and will be discussed in the fourth lecture. The last question is yet to be answered from a theoretical point of view, but empirically, this model is extremely effective in representing different sources of signals and in various problems, leading to state of the art results, which will be demonstrated in the fifth and final lecture.

Having answered the above three important questions positively, we are now ready to dive into the wonderful world of sparse and redundant representation modelling of images, or *Sparseland*. For extensive coverage of this area, the reader is encouraged to turn to my book:

> M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, New-York, 2010, ISBN: 978-1-4419-7010-7 (http://www.springer.com/978-1-4419-7010-7).

I would like to take this opportunity and thank Springer for their permission to publish these lecture-notes with PCMI. It should be noted that the copyright of some of the material here (e.g. the figures in Chapters 4 and 5) belongs to Springer. Finally, the reader will find it useful to read these lecture-notes while looking at the slides: http://www.cs.technion.ac.il/~elad/talks/2010/Summer-School-2010-PCMI.rar.

LECTURE 1

Introduction to sparse approximations - algorithms

1. Motivation and the sparse-coding problem

We start our journey with a pure linear algebra point of view, and specifically by considering a linear under-determined set of equations,

$$\mathbf{D}_{n\times k}\alpha = \mathbf{x}\ ,\ k > n. \tag{1.1}$$

In order to simplify the following developments, we will assume that the columns of $\mathbf{D}$ are (ℓ_2) normalized.

Now, as for terminology used: We shall refer hereafter to $\mathbf{x}$ as a *signal* to be processed, and α will stand for its *representation*. The matrix $\mathbf{D}$ will be referred to as the *dictionary*, and its columns will be called *atoms*.

Assuming $\mathbf{D}$ is full rank, there is an infinite number of solutions to this equation. In engineering, such a situation is unacceptable, since we aim to get a single solution to our problems. In order to be able to choose the "best" single solution, we require some quality measure $J(\alpha)$ that basically ranks the solutions based on some measure, this way choosing the solution that solves the problem

$$\hat{\alpha} = \arg\min_{\alpha} J(\alpha)\ s.t.\ \mathbf{D}\alpha = \mathbf{x}. \tag{1.2}$$

Of course, the natural question is how to select J properly. In engineering, the most common answer is some quadratic term $J(\alpha) = \|\mathbf{B}\alpha\|_2^2$, as it is easy to minimize, having a closed form solution. When chosen as a non-quadratic term, J is preferred to be convex in order to ensure the existence of a unique global minimum to the above problem. Of course, in many cases such choices are not aligned with our needs, as we shall see next.

In these lectures we are interested in cases when α is sparse. In order to gain a mathematical intuition, we consider the general ℓ_p-norm, $J(\alpha) = \|\alpha\|_p^p$. This expression sums the absolute entries of the vector α after being raised to a power p. Note that when p drops below 1 it is no longer a norm, but we shall disregard this technicality.

When p goes below 1 and approaches zero, the function $|\alpha_i|^p$ resembles the indicator function, being 1 for every $\alpha_i \neq 0$, and 0 for $\alpha_i = 0$. Effectively, we define the ℓ_0-norm, as a function counting the number of non-zeros in α and denote it by $\|\alpha\|_0^0$. While ℓ_0 is definitely not a norm, we will abuse this notation for simplicity. We also define the term *Support* as the locations of non-zeros in α. We therefore write the minimization problem (P_0):

$$(P_0)\ \ \hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0\ s.t.\ \mathbf{D}\alpha = \mathbf{x}. \tag{1.3}$$

In many cases our system might contain noise, and we cannot expect a perfect solution to the linear system $\mathbf{D}\alpha = \mathbf{x}$. Instead, we require some proximity between $\mathbf{D}\hat{\alpha}$ and $\mathbf{x}$, arriving at the following alternative problem:

$$(P_0^\epsilon) \;\; \hat{\alpha} = \arg\min_\alpha \|\alpha\|_0^0 \; s.t. \; ||\mathbf{D}\alpha - \mathbf{x}||_2 \leq \epsilon, \tag{1.4}$$

where ϵ is closely related to the properties of the noise. We will be seeing this minimization problem many times during the course.

In order to understand the complexity of the above defined problem, let us propose a method to perform this minimization in a direct way. Set $L = 1$, and consider all supports $\{S_i\}_{i=1}^I$ of cardinality L (there are $I = \binom{k}{L}$ such supports). For each support, we attempt to represent $\mathbf{x}$ using only columns from $\mathbf{D}$ in the support. This can be done by solving a simple Least-Squares (LS) problem,

$$\min_\alpha ||\mathbf{D}\alpha - \mathbf{x}||_2 \; s.t. \; Sup(\alpha) = S_j.$$

If the result LS error is below the threshold ϵ, than we have found a solution and can terminate. If none of the supports yields a solution with an error below the threshold, L should be incremented and the process should be repeated.

Taking a nominal case where $k = 2000$, and a known support size $L = 15$ (so searching the value of L is not needed), and requiring a mere 1 nano-second for each LS operation, solving just one problem will take about $7.5 \cdot 10^{20}$ years. This is due to the exponential nature of the problem. Indeed, the problem posed in Equation (1.4) is known to be NP-Hard [**41**]. This limitation leads us to seek approximation algorithms for these two problems, (P_0) and (P_0^ϵ).

There are two main approaches to approximate the solution to the problem posed in Equation (1.4), which we will explore in this lecture. The first approach is the greedy family of methods, where we attempt to build the solution one non-zero at a time. The second approach is the relaxation methods, which attempt to solve the problem by smoothing the ℓ_0-norm and using continuous optimization techniques. All these techniques that aim to approximate the solution of the problem posed in Equation (1.4) are commonly referred to as pursuit algorithms.

While this lecture will be devoted to the various approximation algorithms, the next lecture will discuss in depth the quality of the estimations they provide. The second lecture will also discuss conditions that should be met in order to ensure that (P_0) has a unique solution, and (P_0^ϵ) has a stable solution (as we shall see, we will not be able to require uniqueness due to the noise).

2. Greedy algorithms

Our first journey into approximation algorithms will deal with the family of greedy algorithms. These algorithms attempt to build the support of α one non-zero element at a time. There is a variety of greedy techniques – see [**22, 3, 9, 38, 40, 42**] for more details.

The most basic of algorithm is the Matching Pursuit (MP). This is an iterative algorithm that starts by finding the *one* atom that best describes the input signal, i.e., finding the index $\hat{i} = \arg\min_i \min_c ||\mathbf{x} - c \cdot \mathbf{d_i}||_2^2$. Once found, we compute the signal residual as $\mathbf{x} - c \cdot \mathbf{d_i}$ with the optimally found constant c and the atom $\mathbf{d}_i$.

Then, at each iteration, the *one* atom that best describes the residual is added to the support, with the appropriate coefficient contributing to reducing the residual the most. This process is repeated until the norm of the threshold drops below the

given threshold. Note that the same atom may be added to the support several times – in such cases, its corresponding coefficient is aggregated.

The Orthogonal Matching Pursuit (OMP) is an improvement of the MP. The OMP re-computes the set of coefficients for the entire support each time an atom is added to the support. This stage speeds up the convergence rate, as well as insures that once an atom has been added to the support, it will not be selected again. This is a result of the LS stage, making the residual orthogonal to all atoms in the chosen support.

A further improvement to the OMP observes that the atom selection stage does not completely correspond to the objective of reducing the residual (because of the LS stage). Therefore, this version suggests to select the atom that will best reduce the residual after the LS stage. This method is accordingly known as LS-OMP. While LS-OMP seems to require more computations, there are some numerical short-cuts to employ, which bring the efficiency of this method very close to that of the OMP.

While improving the results of the MP, both the OMP and the LS-OMP require more complex implementations and more calculations. The other direction, of simplifying the MP, has been attempted too, with the goal of reducing the amount of computations needed at each stage. The Weak-MP is born from the observation that in the MP, it is probably not necessary to find the atom that most reduces the residual, but an atom that can come close to this performance might do as well. As a result, the atoms are scanned in order, and as soon as an atom is found to reduce the residual by some threshold (or beyond), it is added to the support and the iteration ends.

A further simplification can be obtained by avoiding the search at each stage completely, and instead determine the order in which atoms are to be added in advance. Such an algorithm is called the Thresholding algorithm. In this method, the inner product between the signal and all the atoms are computed, and then sorted in descending order of magnitude. This order determines the order in which atoms will be added to the support. At each iteration, the next atom is added to the support, and LS over the support is carried out to best represent the signal (this can be done using recursive least-squares, which avoids a direct matrix inversion). As soon as the norm of the residual drops below the given threshold, the process terminates.

3. Relaxation algorithms

In the previous section we reviewed the greedy algorithms, which attempt to construct the support one atom at a time. Now it is time to turn to relaxation methods, that smooth the ℓ_0-norm and solve the alternative obtained problem using classical methods from continuous optimization [**22, 3**].

One natural way to do this is to replace the $\|\alpha\|_0^0$ penalty with the expression $\sum_{i=1}^{K}\left[1-\exp\left(-\beta\alpha_i^2\right)\right]$, and observe that as $\beta\to\infty$, this term better approximates the ℓ_0 norm. Therefore, it is possible to approximate the ℓ_0 solution by optimizing for a small value of β, and increase the value of β every few iterations.

An alternative is to "convexise" the ℓ_0-norm, which brings us to the Basis Pursuit (BP) algorithm [**7**]. In this algorithm, the ℓ_0-norm is simply replaced by

the ℓ_1-norm, leading to the optimization problem

$$\min_{\alpha} \|\alpha\|_1 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{x}\|_2 \leq \epsilon, \tag{1.5}$$

with the hope that the solution of this optimization problem is close to the solution of the original one. This problem can be formulated as linear programming for the exact case ($\epsilon = 0$), and quadratic programming for the general case. For this task, several types of efficient solvers can be employed, such as Interior Point methods [**33**], Least-Angle-Regression (LARS) [**20**] and Iterative Shrinkage [**8, 21, 24, 25, 26, 27**].

Another interesting option for handling the problem formed above (either (1.4) or (1.5)), is the Iterative Reweighted Least-Squares (IRLS). Before explaining this algorithm, we slightly change the formulation of the problem. Instead of enforcing the constraint $\|\mathbf{D}\alpha - \mathbf{x}\|_2 \leq \epsilon$, we now insert it into the optimization problem:

$$\min_{\alpha} \ \lambda \cdot \|\alpha\|_p^p + \|\mathbf{D}\alpha - \mathbf{x}\|_2^2, \tag{1.6}$$

with λ balancing the effects of both terms. A proper choice of λ makes the two formulations equivalent. Notice that our formulation uses the ℓ_p-norm, thus covering both cases posed in Equations (1.4) or (1.5).

Now, let us re-write the first term:

$$\|\alpha\|_p^p = \sum_{i=1}^{k} |\alpha_i|^p = \sum_{i=1}^{k} |\alpha_i|^{p-2} \cdot |\alpha_i| = \alpha^T \mathbf{A}(\alpha)\, \alpha. \tag{1.7}$$

Based on this, we can suggest an iterative optimization scheme in which we freeze $\mathbf{A}(\alpha)$, and minimize for α:

$$\hat{\alpha}_{j+1} = \arg\min_{\alpha} \lambda \alpha^T \mathbf{A}(\alpha_j)\, \alpha + \frac{1}{2}\|\mathbf{D}\alpha - \mathbf{x}\|_2^2. \tag{1.8}$$

This is a quadratic term, and as such, it is easy to minimize. In the following step, we freeze α_{j+1} and compute $\mathbf{A}(\alpha_{j+1})$, and so on, until the solution converges. More on the IRLS (sometimes called FOCUSS) can be found in [**28**].

In the previously described algorithm, our only demand of the residual is that its norm is bounded. However, in the general case, we expect the residual not only to be bounded, but we also expect it to look like noise. In other words, we expect the residual to show low correlation with any of the atoms in the dictionary. This observation leads to the formulation of the Dantzig Selector (DS) [**6**]:

$$\min_{\alpha} \|\alpha\|_1 \ s.t. \ \|\mathbf{D}^T(\mathbf{D}\alpha - \mathbf{x})\|_\infty \leq T. \tag{1.9}$$

Interestingly, this is a linear programming task even in the noisy case (unlike the BP, which leads to a quadratic programming problem in the noisy case), and therefore a wide range of tools can be used for solving it.

Simulations comparing the performance of the DS with those of the BP (see [**22**]) show that these two algorithm perform (on average) at the same level. It is well known that in the unitary case, the two are perfectly equivalent.

4. A closer look at the unitary case

What happens in the special case where the dictionary $\mathbf{D}$ is unitary, i.e., $\mathbf{D}^T\mathbf{D} = \mathbf{I}$ (the dictionary is of course not redundant in such a case)? Starting from the problems posed in Equations (1.4) and (1.5), let us write a more general optimization

goal,

$$f(\alpha) = \frac{1}{2}\|\mathbf{D}\alpha - \mathbf{x}\|_2^2 + \lambda \cdot \rho(\alpha) \tag{1.10}$$

where we use a general function $\rho(\alpha) = \sum_{j=1}^{k} \rho(\alpha_j)$ measuring the "quality" of the solution. This could be the ℓ_0 pseudo-norm, the ℓ_1-norm or any other separable choice of penalty. Exploiting the fact that $\mathbf{D}$ is unitary, we get

$$\begin{aligned} f(\alpha) &= \frac{1}{2}\|\mathbf{D}\alpha - \mathbf{x}\|_2^2 + \lambda \cdot \rho(\alpha) = \frac{1}{2}\|\mathbf{D}\alpha - \mathbf{D}\mathbf{D}^T\mathbf{x}\|_2^2 + \lambda \cdot \rho(\alpha) \\ &= \frac{1}{2}\|\mathbf{D}(\alpha - \beta)\|_2^2 + \lambda \cdot \rho(\alpha) = \frac{1}{2}\|\alpha - \beta\|_2^2 + \lambda \cdot \rho(\alpha). \end{aligned} \tag{1.11}$$

The first transition employed the unitary property, $\mathbf{D}\mathbf{D}^T = \mathbf{I}$. The second is done by denoting $\beta = \mathbf{D}^T\mathbf{x}$. The last transition is due to the fact that the ℓ_2-norm is invariant to a unitary transformation. Since $\rho()$ works on each entry of α independently, we can finally write

$$f(\alpha) = \sum_{j=1}^{k} \left[\frac{1}{2}(\alpha_j - \beta_j) + \lambda \cdot \rho(\alpha_j) \right], \tag{1.12}$$

which indicates that this problem is separated into k easy to solve scalar optimization problems.

As the problem becomes a set of independent 1-D problems, it is possible to compute the output $\hat{\alpha}$ for any input value of β, effectively forming a lookup table. Plotting this graph generally looks like a shrinkage curve[1] that nulls the small entries and the large ones intact. Of course, such a graph can be created for any penalty function ρ and value λ.

4.1. ℓ_0 and greedy algorithms for unitary dictionaries

When working with the ℓ_0, the function ρ is the indicator function, $\rho(\alpha_j) = |\alpha_j|^0$. Therefore, the independent optimization problem to solve is

$$\alpha_j^{opt} = \arg\min_{\alpha_j} \left[\frac{1}{2}(\alpha_j - \beta_j)^2 + \lambda \cdot |\alpha_j|^0 \right]. \tag{1.13}$$

This is a very simple task - there are only two logical options for the value of α_j. Either $\alpha_j = \beta_i$, paying λ for the second term, or $\alpha_j = 0$, and paying $\frac{1}{2}\beta_j^2$ in the first term. Any other value will result in a necessarily larger penalty.

Therefore, the shrinkage curve for this case is very simple. For $|\beta| < \sqrt{2}\lambda$, we choose $\alpha = 0$, while for $|\beta| \geq \sqrt{2}\lambda$, we choose $\alpha = \beta$. Therefore, this method is known as Hard-Thresholding, as any coefficient below the threshold $\sqrt{2} \cdot \lambda$ is set to zero, while every coefficient above this threshold remains untouched. This solution is the *global* minimizer of the problem we tackled.

It is possible to show that when the dictionary is unitary, all the greedy algorithms we have seen (thresholding, OMP, LS-OMP, weak-OMP, MP) obtain the exact solution, which is the one obtained by the hard-thresholding algorithm described here. Just to illustrate this, the thresholding algorithm will compute $\mathbf{D}^T\mathbf{x}$, and work on the coefficients in descending order of magnitude. The amount of atoms it will choose is exactly those that the hard-thresholding will leave intact,

[1] The name comes from the fact that the resulting value of $\hat{\alpha}$ is always smaller or equal to the input β.

i.e., the amount of atoms above the threshold. The OMP does the same, simply choosing one atom at a time. Since the dictionary is unitary, adding another atom to the support does not effect the inner products of the residual with the remaining atoms.

4.2. Relaxation algorithms for unitary dictionaries

The basis pursuit uses the ℓ_1 -norm as a penalty on α. In the unitary case, this leads to separate optimization problem of the form

$$\alpha_j^{opt} = \arg\min_{\alpha_j} \left[\frac{1}{2} (\alpha_j - \beta_j)^2 + \lambda \cdot |\alpha_j| \right]. \tag{1.14}$$

Analyzing this problem, it turns out that the solution becomes

$$\alpha_j^{opt} = \begin{cases} \beta_j - \lambda & \beta_j > \lambda \\ 0 & |\beta_j| \leq \lambda \\ \beta_j + \lambda & \beta_j < -\lambda \end{cases} \tag{1.15}$$

The curve this solution implies is known as soft-thresholding, as when the coefficient is above the threshold it does not remain the same (as in the hard-thresholding), but is made smaller instead.

The Dantzig Selector tries to solve an alternative problem (1.9). When we use the unitary property, it becomes

$$\begin{aligned} \hat{\alpha} &= \min_{\alpha} \|\alpha\|_1 \ s.t. \ \|\mathbf{D}^T (\mathbf{D}\alpha - \mathbf{x}) \|_\infty = \min_{\alpha} \|\alpha\|_1 \ s.t. \ \|\alpha - \mathbf{D}^T\mathbf{x}\|_\infty, \\ &= \min_{\alpha} \|\alpha\|_1 \ s.t. \ \|\alpha - \beta\|_\infty, \end{aligned} \tag{1.16}$$

where in the first transition we used $\mathbf{D}^T\mathbf{D} = \mathbf{I}$, which is the unitarity property, and in the second transition simply re-introduced the notation of β. The last step results in K independent optimization problems of the form:

$$\arg\min_{\alpha_i} |\alpha_j| \ s.t. \ |\alpha_j - \beta_j| \leq T \tag{1.17}$$

It is not difficult to see that the solution of this optimization problem is exactly identical to the solution offered for the BP algorithm in (1.15). The conclusion is that BP and DS are exactly identical when the dictionary is unitary, and both of them can be solved using a closed-form formula for each of the coefficients (such that they do not require a complex optimization procedure).

4.3. Unitary dictionaries - summary

We have seen that when the dictionary is unitary, both families of algorithms enjoy a simple closed-form solution. Given the signal $\mathbf{x}$, multiply it by the dictionary to obtain a vector of coefficients $\beta = \mathbf{D}^T\mathbf{x}$. On each of the coefficients, apply independently a Look-Up Table operation with a shape that depends on ρ and λ, and this results in the vector of coefficients $\hat{\alpha}$. This simple procedure yields the *global* optimum of $f(\alpha)$, even if f itself is not convex. This leads to the obvious question – can this be used to gain an insight to the general case, when the dictionary is non-unitary? Can we use it to construct simpler, more effective algorithms for such cases? In the 4th lecture we will see a family of algorithms doing exactly that.

LECTURE 2

Introduction to sparse approximations - theory

This lecture deals with the theory of sparse approximations. We will start by defining several properties of dictionaries - the Spark, the Mutual-Coherence, and the Restricted Isometry Property (RIP). We will then show theoretical guarantees for the uniqueness of the solution for (P_0), and show the equivalence of MP and BP for the exact case. We will then continue with the discussion of (P_0^ϵ), and show theoretical guarantees on the stability of the solution. Finally, we will mention briefly results related to near-oracle performance of pursuit techniques in the noisy case.

1. Dictionary properties

The purpose of this section is finding ways to characterize the dictionary $\mathbf{D}$, in order for us to be able to make claims about the conditions under which pursuit algorithms are guaranteed to succeed. Ideally, we would like these bounds to be tight, and the characterizations easy to compute. As we will see, this is not necessarily the case.

1.1. The matrix spark

The Spark was defined by Donoho & Elad [**14**] (similar definitions have been made in tensor decomposition by Kruskal [**34, 35**] as well as in coding theory):

Definition 2.1 (Spark)**.** Given a matrix $\mathbf{D}$, $\sigma = Spark\left(\mathbf{D}\right)$ is the smallest number of atoms from $\mathbf{D}$ that are linearly dependent.

While this definition is likely to remind the reader of the definition of the rank of a matrix, it is very different. There are two differences between the definitions - while the rank is the *largest* number of *independent* atoms, the Spark is the *smallest* number of *dependent* atoms. For an example, consider the matrix

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The rank of this matrix is 4, the maximal number of independent columns (e.g., the first four columns). The Spark of this matrix is 3: one column could only be linearly dependent if it is the zero vector; Any two columns in this dictionary are linearly independent; There is a group of three columns - the first, second and last - that are linearly dependent. Therefore, $\sigma\left(\mathbf{D}\right) = 3$.

It is easy to observe that for a dictionary $\mathbf{D}_{n\times k}$, the Spark satisfies $1 \leq \sigma \leq n+1$. Also, if for some non-trivial vector $\mathbf{v}$ it exists that $\mathbf{D}\mathbf{v} = \mathbf{0}$, than is is certain that $\|\mathbf{v}\|_0^0 \geq \sigma$, since the support of $\mathbf{v}$ constitutes a linearly dependent set of atoms. Since the smallest possible such set is σ, the support of $\mathbf{v}$ also contains at least σ atoms.

1.2. The mutual coherence

Definition 2.2 (Mutual Coherence)**.** For a matrix $\mathbf{D}$, the Mutual-Coherence M is the largest (in magnitude) off-diagonal element of the Gram matrix of $\mathbf{D}$, i.e., $\mathbf{G} = \mathbf{D}^T\mathbf{D}$.

For this definition, we assume that the columns of $\mathbf{D}$ are normalized. The Mutual-Coherence measures the maximal similarity (or anti-similarity) between two columns on the dictionary. Therefore, its value ranges between 0 (an orthonormal basis) and 1 (when there are two identical atoms). As we will see later, a large mutual coherence is bad news for our algorithms, as the algorithms are likely to confuse between the two atoms.

1.3. The relationship between the spark and the mutual coherence

Our next task is to understand the relationship between the Spark and the Mutual Coherence measures. In order to do that, we need the following well-known theorem:

Theorem 2.1 (Gershgorin's Disk Theorem)**.** *Consider a general square matrix* $\mathbf{A}$*, and define the* k'*th disk as* $disk_k = \left\{x \middle| |x - a_{kk}| \leq \sum_{j\neq k} |a_{jk}|\right\}$*. Then, the eigenvalues of* $\mathbf{A}$ *all lie in the union of all disks,* $\{\lambda_1, \lambda_2, .., \lambda_n\} \subset \cup_{k=1..n} disk_k$*.*

Alternatively, we can write $\lambda_{min} \geq \arg\min_{1\leq k\leq n} \left\{a_{kk} - \sum_{j\neq k} |a_{jk}|\right\}$ and $\lambda_{max} \leq \arg\max_{1\leq k\leq n} \left\{a_{kk} + \sum_{j\neq k} |a_{jk}|\right\}$. One of the implications of the theorem is that if $\mathbf{A}$ is diagonally dominant (assuming real entries), than $\mathbf{A}$ is necessarily positive definite. This is because the origin is not in any of the disks, and therefore all eigenvalues are positive. Using this theorem, we are now ready to make the following claim:

Theorem 2.2 (Relation between Spark and Mutual Coherence)**.** $\sigma \geq 1 + \frac{1}{M}$

Proof. Let us consider a group of columns S from $\mathbf{D}$, and compute its sub-Gram matrix. We know that these columns are independent, if this matrix is positive definite. From the Gershgorin's disk theorem, we know that the matrix will be positive definite if it is diagonally dominant.

The Mutual-Coherence bounds any off-diagonal element by M, while the diagonal elements are all 1. Therefore, the matrix is diagonally dominant if $(S-1)\,M < 1$. As long as $S < 1 + 1/M$, any group of S columns are independent. Therefore, the minimal number of columns that could be dependent is $S = 1 + 1/M$, and this is exactly the claimed result. □

1.3.1. *Grassmanian Frames.* We now turn to deal with the following question: For a general matrix $\mathbf{D}_{n\times k}$, with $k > n$, what is the best (smallest) Mutual-Coherence we can hope for? Obviously, it can't be zero, as that would mean that there are k columns that are orthogonal, and that is not possible when their length is n. It turns out that

$$M_{min} = \sqrt{\frac{k-n}{n\,(k-1)}} = O\left(\frac{1}{\sqrt{n}}\right), \tag{2.1}$$

with the second step using the substitution $k = \rho n$, and the approximation is true for small values of ρ (e.g., 2,3). We will omit the proof, and only state that this question is closely related to questions about packing vectors in the n-dimensional

space in the most spacious ways. As an example, for a matrix with $n = 50$ rows and $k = 100$ columns, the minimal possible coherence is roughly 0.1.

Can matrices achieving this bound be found in practice? The answer is yes, and this family of matrices is known as Grassmanian Frames [**49**]. The prevailing property of these frames is that the off-diagonal elements in their Gram-matrices all have the same magnitude. They are difficult to built, and a numerical algorithm by Tropp has been developed for their construction [**52**]. It is important to note, however, that for some pairs of n and k such frames do not exist.

1.4. The restricted isometry property

Definition 2.3 (Restricted Isometry Property)**.** Given a matrix $\mathbf{D}$, consider all s-sparse vectors α, and find the smallest value δ_s that satisfies

$$(1 - \delta_s) \|\alpha\|_2^2 \leq \|\mathbf{D}\alpha\|_2^2 \leq (1 + \delta_s) \|\alpha\|_2^2.$$

$\mathbf{D}$ is then said to have an s-RIP with a constant δ_s.

What is the meaning of this definition? $\mathbf{D}\alpha$ combines linearly atoms from $\mathbf{D}$. The RIP suggests that every such group of s atoms behaves like an isometry, i.e., does not considerably change the length of the vector it operates on.

As an example, when $\mathbf{D}$ is unitary, the length does not change for any s. Therefore, $\delta_s(\mathbf{D}) = 0$, for every s. In this sense, the RIP asks how close is $\mathbf{D}$ to a unitary matrix, when considering multiplications by s-sparse vectors.

1.5. The RIP and the mutual coherence

Let us denote $\mathbf{D}_S$ as the sub-matrix of $\mathbf{D}$ containing only the columns in the support S, and similarly, α_S as containing only the non-zeros in α, so $\mathbf{D}_S\alpha_S = \mathbf{D}\alpha$. Using this terminology, the RIP property states that

$$(1 - \delta_s) \|\alpha_S\|_2^2 \leq \alpha_S^T \mathbf{D}_S^T \mathbf{D}_S \alpha_S \leq (1 + \delta_s) \|\alpha_S\|_2^2.$$

On the other hand, from an eigenvalue perspective, we know that

$$\|\alpha_S\|_2^2 \cdot \lambda_{min}\left(\mathbf{D}_S^T\mathbf{D}_S\right) \leq \alpha_S^T \mathbf{D}_S^T \mathbf{D}_S \alpha_S \leq \|\alpha_S\|_2^2 \cdot \lambda_{max}\left(\mathbf{D}_S^T\mathbf{D}_S\right).$$

From the Gershgorin's theorem we already know that

$$\lambda_{min}\left(\mathbf{D}_S^T\mathbf{D}_S\right) \geq 1 - (s-1)\,M$$
$$\lambda_{max}\left(\mathbf{D}_S^T\mathbf{D}_S\right) \leq 1 + (s-1)\,M,$$

and finally, we see that

$$\delta_s \leq (s-1)\,M. \tag{2.2}$$

1.6. The RIP and the spark

Suppose that the Spark of $\mathbf{D}$ is known to be σ. Then, for $s = \sigma$, there is at least one vector α of cardinality s such that $\mathbf{D}\alpha = \mathbf{0}$, therefore indicating that $\delta_\sigma = 1$. Furthermore, for any value of $s < \sigma$, it follows that $\delta_s < 1$.

1.7. Computing the spark, the MC and the RIP

Summarizing the discussion so far, we consider the question of computing the above described properties. The Spark requires going through each and every support and checking for linear dependence. Thus, it is impossible to compute in general as it is combinatorial. There is an exception, however, as a random iid matrix has a full Spark of $(n + 1)$.

The RIP similarly has to scan all supports, and is therefore also impossible to compute in practice. There is also an exception here as well, in the case of a random Gaussian matrix, the RIP can be bounded (with high probability) by $\delta_s \leq C \cdot \sqrt{\frac{s}{n} \cdot \log\left(\frac{k}{s}\right)}$. The C is a coefficient controlling the probability – as the required confidence grows, so does the constant C, making the bound less tight.

Fortunately, the Mutual Coherence is very easy to compute, as we have seen - it only requires computing the Gram matrix, and scanning all elements in it. Unfortunately, as this measure only quantifies pairs of atoms, it is also the weakest of the three measures.

Lastly, it is important to note that all of the measures introduced are of worst-case nature, and as such lead to worse-case performance analysis of algorithms. For example, it is enough that the dictionary contains a duplicate atom to set the Mutual Coherence to 1. The rest of the dictionary can behave perfectly, but that will be unknown to the coherence. The Spark and the RIP will also suffer in such a case in a similar fashion. Different types of measures are needed for average-case performance analysis, but such measures are still very much missing.

2. Theoretical guarantees - uniqueness for P_0

After the introduction of some tools to be used, we now turn to the first problem of interest. Suppose Alice holds a dictionary $\mathbf{D}$, and generates some sparse vector of coefficients α. Multiplying the two, Alice generates the signal $\mathbf{x} = \mathbf{D}\alpha$. Then, Alice contacts her friend, Bob, and gives him the dictionary $\mathbf{D}$ and the vector $\mathbf{x}$, and asks him to find the vector α she had used.

Bob knows that α is sparse, and so he sets out to seek the sparsest possible solution of the system $\mathbf{D}\alpha = \mathbf{x}$, namely, solve the optimization problem

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \mathbf{D}\alpha = \mathbf{x}.$$

For the time being, we shall assume that Bob can actually solve this problem exactly (we have seen in the first lecture that this is generally NP-hard, and therefore impossible in practice). The question we want to ask is if (or when) can we expect that $\hat{\alpha} = \alpha$. It might turn our that $\hat{\alpha}$ is even sparser than the original α. This is essentially the question of uniqueness – what properties must α and the dictionary $\mathbf{D}$ have so that α is the sparsest vector to create $\mathbf{x}$? The answer turns out to be quite simple [**14**]:

Theorem 2.3 (Uniqueness Requirements of P_0)**.** *The following two claims define the uniqueness condition and implication:*

1. *Suppose we have found a sparse solution $\hat{\alpha}$ that satisfies $\mathbf{D}\hat{\alpha} = \mathbf{x}$. If this solution satisfies $\|\hat{\alpha}\|_0^0 < \sigma/2$, then this this solution is unique (i.e., it is the sparsest solution possible).*
2. *If Alice generated a signal with $\|\alpha\|_0^0 < \sigma/2$, than Bob is able to recover α exactly: $\hat{\alpha} = \alpha$.*

Proof. Suppose we have found two candidate solutions α_1 and α_2 to the linear system of equations $\mathbf{D}\alpha = \mathbf{x}$, leading to $\mathbf{D}\alpha_1 = \mathbf{D}\alpha_2$ and $\mathbf{D}(\alpha_1 - \alpha_2) = 0$. From the Spark property, the cardinality of any vector $\mathbf{v}$ that satisfied $\mathbf{D}\mathbf{v} = 0$ is at least the Spark, indicating that

$$\|\alpha_1 - \alpha_2\|_0^0 \geq \sigma. \tag{2.3}$$

We can also observe that

$$\|\alpha_1\|_0^0 + \|\alpha_2\|_0^0 \geq \|\alpha_1 - \alpha_2\|_0^0. \tag{2.4}$$

This property can be understood by analyzing the supports of both vectors. If there is no overlap in their supports, then the equality holds. If there is an overlap, and two terms in the overlap happen to cancel each other exactly, then the resulting support is smaller than the sum of sizes of the two supports. We can refer to this property as some sort of triangle inequality of the ℓ_0, even though it is not actually a norm.

Combining the two inequalities, it turns out that the sum of cardinalities of both vectors is at least the Spark,

$$\|\alpha_1\|_0^0 + \|\alpha_2\|_0^0 \geq \sigma. \tag{2.5}$$

This can be viewed as an uncertainty principle – two solutions cannot be jointly very sparse. If one solution is indeed sparse, the other must be non-sparse. The uncertainty principle implies that if we have found a solution α_1 with cardinality less than half the Spark, i.e., $\alpha_1 < \sigma/2$, than *every* other solution α_2 must have a cardinality of more than half the Spark, in order to fulfill the requirement that $\|\alpha_1\|_0^0 + \|\alpha_2\|_0^0 \geq \sigma$. Therefore, α is sparser than any other solution, and indeed, the sparsest solution.

It directly follows that if Alice chose α such that $\|\alpha\|_0^0 < \sigma/2$, than it is the sparsest solution to $\mathbf{D}\alpha = \mathbf{x}$, and therefore Bob will recover it exactly. □

3. Equivalence of the MP and BP for the exact case

Now we turn to walk in Bob's shoes. We have seen that Bob's task is impossible to solve directly, and therefore he has to turn to approximation algorithms, such as Orthogonal Matching Pursuit (OMP)—, Basis Pursuit (BP), etc. The question we ask now is: Is there any hope that any of these algorithms will actually recover the exact solution? Surprisingly, we will see that under some conditions, these algorithms are guaranteed to succeed.

3.1. Analysis of the OMP

Recall the OMP algorithm, introduced in lecture 1. To make our life easier, we assume that the s non-zero locations in α are in the first s locations. Furthermore, we assume that these coefficients are sorted in decreasing order of magnitude, with $|\alpha_1| \geq |\alpha_2| \geq .. \geq |\alpha_s| > |\alpha_{s+1}| = 0$. Thus, our signal can be written as $\mathbf{x} = \sum_{i=1}^{s} \alpha_i \mathbf{d}_i$. This assumption does not effect the generality of our claims, as we can achieve this structure by reordering the columns of $\mathbf{D}$ and the locations of the non-zeros in α. The re-structured problem remains equivalent to the original one.

We now ask ourselves what conditions are needed for the first OMP step to succeed, i.e., that the first atom added to the support is an atom in the true support (one of the first s atoms)? Since the OMP selects the atom with the largest magnitude inner-product, the first step succeeds if

$$\left|\mathbf{x}^T \mathbf{d}_1\right| > \max_{j>s} \left|\mathbf{x}^T \mathbf{d}_j\right|. \tag{2.6}$$

This condition guarantees that the atom selection will prefer the first atom to any atom atom outside the support, and thus guarantees that an atom in the true support is chosen. It may happen that another atom in the true support is chosen

instead of the first, but this is also considered as a success. Therefore, the question we want to answer is when does the condition (2.6) hold.

In order to find out under what conditions this inequality is satisfied, we lower bound the left-hand-side (LHS), and upper bound the right-hand-side(RHS). We will then find the conditions for the bound on the LHS still being greater than the upper bound of the RHS, thus fulfilling the original inequality as well.

We start with an upper-bound of the Right-Hand-Side (RHS):

$$\max_{j>s} \left|\mathbf{x}^T\mathbf{d_j}\right| = \max_{j>s} \left|\sum_{i=1}^{s} \alpha_i \mathbf{d}_i^T \mathbf{d}_j\right| \le \max_{j>s} \sum_{i=1}^{s} |\alpha_i| \cdot \left|\mathbf{d}_i^T\mathbf{d}_j\right| \le |\alpha_1| \cdot s \cdot M. \tag{2.7}$$

The first transition assigns the expression $\mathbf{x} = \sum_{i=1}^{s} \alpha_i \mathbf{d}_i$, the second uses the property $|a+b| \le |a| + |b|$, and the third uses the definition of the Mutual-Coherence as an upper bound on $\left|\mathbf{d}_i^T\mathbf{d}_j\right|$, as well as the monotonicity of the coefficients, implying that $|\alpha_k| \le |\alpha_1|$ for $k = 2, .., s$. We now turn to lower-bound the Left-Hand-Side(LHS) in inequality (2.6):

$$\begin{aligned} \left|\mathbf{x}^T\mathbf{d}_1\right| &= \left|\sum_{i=1}^{s} \alpha_i \mathbf{d}_i^T\mathbf{d}_1\right| = \left|\alpha_1 + \sum_{i=2}^{s} \alpha_i \mathbf{d}_i^T\mathbf{d}_1\right| \ge |\alpha_1| - \left|\sum_{i=2}^{s} \alpha_i \mathbf{d}_i^T\mathbf{d}_1\right| \\ &\ge |\alpha_1| - \sum_{i=2}^{s} |\alpha_i| \cdot \left|\mathbf{d}_i^T\mathbf{d}_1\right| \ge |\alpha_1| \cdot [1 - (s-1)\cdot M]\,. \end{aligned} \tag{2.8}$$

The transitions in the first line are made by first substituting $\mathbf{x} = \sum_{i=1}^{s} \alpha_i \mathbf{d}_i$ and then using the fact that the atoms are normalized, i.e., $\mathbf{d}_1^T\mathbf{d}_1 = 1$. Then, the bounding starts by the reversed triangle inequality $|a+b| \ge |a| - |b|$, and then repeating the steps performed in bounding the RHS (note that since the second term is with a negative sign, increasing its magnitude reduces the magnitude of the overall expression).

Using the bounds in (2.7) and (2.8) and plugging them into (2.6), we get that a condition that guarantees that the first OMP step succeeds is

$$\left|\mathbf{x}^T\mathbf{d}_1\right| \ge |\alpha_1| \cdot [1 - (s-1)\cdot M] > |\alpha_1| \cdot s \cdot M \ge \max_{j>s} \left|\mathbf{x}^T\mathbf{d_j}\right|,$$

which leads to the following bound on the cardinality of the representation,

$$s < \frac{1}{2}\left[1 + \frac{1}{M}\right]. \tag{2.9}$$

Assuming this condition is met, the first atom chosen is indeed an atom from the true support, with index i^*. Then, a LS step is performed, and a coefficient value α_1^* is assigned to this atom. Now, we look at the residual as our new signal, being

$$\mathbf{r}^1 = \mathbf{x} - \alpha_1^*\mathbf{d}_{i^*} = \sum_{i=1}^{s} \alpha_i \mathbf{d}_i - \alpha_1^*\mathbf{d}_{i^*}.$$

This signal has the same structure we the one we started with – a linear combination of the s first atoms. This means that the exact same condition is needed for the second step to succeed, and in fact, for all other steps to succeed as well.

This means, that if $s < \frac{1}{2}\left[1 + \frac{1}{M}\right]$ the OMP is guaranteed to find the correct sparse representation. This is done in exactly s iterations, as in each iteration one atom in the support is chosen, and due to the LS stage, the residual is always orthogonal to the atoms already chosen.

Using this result, and the bound we have found in the previous section regarding the Spark, we come to the following conclusion:

Theorem 2.4 (OMP and MP Equivalence)**.** *Given a vector* $\mathbf{x}$ *with a representation* $\mathbf{D}\alpha = \mathbf{x}$*, and assuming that* $\|\alpha\|_0^0 < 0.5\,(1 + 1/M)$*, then (i)* α *is the sparsest possible solution to the system* $\mathbf{D}\alpha = \mathbf{x}$*, and (ii) OMP (and MP) are guaranteed to find it exactly* [**50, 16**].

3.2. Analysis of the BP

We now turn to perform the same analysis for the BP, checking under which conditions are we guaranteed to succeed in recovering the sparsest solution. We remind ourselves that the BP aims to approximate the solution of the problem $\hat{\alpha} = \arg\min_\alpha \|\alpha\|_0^0$ *s.t.* $\mathbf{D}\alpha = \mathbf{x}$, by solving instead $\hat{\alpha} = \arg\min_\alpha \|\alpha\|_1$ *s.t.* $\mathbf{D}\alpha = \mathbf{x}$. Therefore, our aim is now to show under which condition is the sparse vector α used to generate the signal $\mathbf{x}$ also the shortest one with respect to the ℓ_1 norm.

The strategy for this analysis starts by first defining the set of all solutions for the linear set of equations $\mathbf{D}\beta = \mathbf{x}$ (excluding the one we started with), such that their ℓ_1 length is shorter (or equal) to the length of the original vector α,

$$C = \{\beta \neq \alpha \,|\mathbf{x} = \mathbf{D}\alpha = \mathbf{D}\beta \text{ and } \|\beta\|_1 \leq \|\alpha\|_1\}\,. \tag{2.10}$$

We will gradually inflate this set, while simplifying it, eventually showing that it is still empty. When done, this will imply that the initial set is also empty, and therefore α is the result of the ℓ_1 minimization, leading to α being the vector with the shortest ℓ_1 norm, therefore recovered by the BP.

Our first step starts with defining the vector $\mathbf{e} = \beta - \alpha$, or in other words, $\mathbf{e} + \alpha = \beta$. Our set remains unchanged, and is simply rewritten as

$$C_{\mathbf{e}} = \{\mathbf{e} \neq \mathbf{0} \,|\mathbf{0} = \mathbf{D}\mathbf{e} \text{ and } \|\alpha + \mathbf{e}\|_1 - \|\alpha\|_1 \leq 0\}\,. \tag{2.11}$$

Next, we simplify the linear constraint $\mathbf{0} = \mathbf{D}\mathbf{e}$,

$$\begin{aligned}
\{\mathbf{e}\,|\mathbf{0} = \mathbf{D}\mathbf{e}\} &= \left\{\mathbf{e}\,\middle|\mathbf{0} = \mathbf{D}^T\mathbf{D}\mathbf{e}\right\} = \left\{\mathbf{e}\,\middle|-\mathbf{e} = \left(\mathbf{D}^T\mathbf{D} - \mathbf{I}\right)\mathbf{e}\right\} \\
&\subseteq \left\{\mathbf{e}\,\middle||\mathbf{e}| = \left|\left(\mathbf{D}^T\mathbf{D} - \mathbf{I}\right)\mathbf{e}\right|\right\} \\
&\subseteq \left\{\mathbf{e}\,\middle||\mathbf{e}| \leq \left|\mathbf{D}^T\mathbf{D} - \mathbf{I}\right| \cdot |\mathbf{e}|\right\} \\
&\subseteq \left\{\mathbf{e}\,\middle||\mathbf{e}| \leq M \cdot \left(\mathbf{1}\mathbf{1}^T - \mathbf{I}\right) \cdot |\mathbf{e}|\right\} \\
&= \{\mathbf{e}\,|(1 + M) \cdot |\mathbf{e}| \leq M\|\mathbf{e}\|_1\}
\end{aligned}$$

The transition from the first row to the second is done by taking absolute value of both sides (an absolute value of a vector is done element-wise). Obviously, some new $\mathbf{e}$'s have been added to the set, as we are no longer forcing the sign equality, implying that the set becomes larger. The next step stems from the triangle inequality. The transition between the third and fourth lines is slightly more complex. Observe that the matrix $\mathbf{D}^T\mathbf{D}$ is effectively the Gram Matrix, and all off-diagonal elements of it are bounded by M. Furthermore, the matrix $\left(\mathbf{D}^T\mathbf{D} - \mathbf{I}\right)$ has zeros along the main diagonal, and all remaining elements are still bounded by M. Observing that $\mathbf{1}\mathbf{1}^T$ is a matrix of the proper size, all containing ones, than $M\left(\mathbf{1}\mathbf{1}^T - \mathbf{I}\right)$ is a matrix containing zeros on the diagonal, and M on the off-diagonal terms. Thus $\left|\mathbf{D}^T\mathbf{D} - \mathbf{I}\right| \leq M \cdot \left(\mathbf{1}\mathbf{1}^T - \mathbf{I}\right)$ element-wise. This way, we no longer need $\mathbf{D}$ directly, instead characterizing it using only its Mutual-Coherence.

The last step re-orders the elements, and uses $\mathbf{1}^T|\mathbf{e}| = \|\mathbf{e}\|_1$. Replacing the linear constraint in (2.11) with the one derived above, we obtain an inflated set

$$C_{\mathbf{e}} \subseteq \left\{ \mathbf{e} \neq \mathbf{0} \,\middle|\, |\mathbf{e}| \leq \frac{M}{M+1} \cdot \|\mathbf{e}\|_1 \text{ and } \|\alpha + \mathbf{e}\|_1 - \|\alpha\|_1 \leq 0 \right\} \tag{2.12}$$

Now we turn to handle the second constraint, $\|\alpha + \mathbf{e}\|_1 - \|\alpha\|_1 \leq 0$,

$$\begin{aligned}
\{\mathbf{e} \,|\, \|\alpha + \mathbf{e}\|_1 - \|\alpha\|_1 \leq 0\} &= \left\{ \mathbf{e} \,\middle|\, \sum_{i=1}^{K} (|\alpha_i + e_i| - |\alpha_i|) \leq 0 \right\} \\
&= \left\{ \mathbf{e} \,\middle|\, \sum_{i=1}^{s} (|\alpha_i + e_i| - |\alpha_i|) + \sum_{i>s} |e_i| \leq 0 \right\} \\
&\subseteq \left\{ \mathbf{e} \,\middle|\, \sum_{i>s} |e_i| - \sum_{i=1}^{s} |e_i| \leq 0 \right\} \\
&= \left\{ \mathbf{e} \,\middle|\, \|\mathbf{e}\|_1 - 2 \cdot \mathbf{1}_s^T |\mathbf{e}| \leq 0 \right\}.
\end{aligned}$$

The first step simply writes the ℓ_1 norm as a sum of elements, and the second step uses the knowledge that only the first s coefficients in α are non-zeros (as we have assumed earlier). The third term uses a version of the triangle inequality, $|x+y| - |x| \geq -|y|$. Using this replacement, each term in the sum is made smaller, thus inserting more possible $\mathbf{e}$'s into the set. The last step merely rewrites the expression, since $\sum_{i>s} |e_i| - \sum_{i=1}^{s} |e_i| = \sum_{i=1}^{K} |e_i| - 2 \cdot \sum_{i=1}^{s} |e_i| = \|\mathbf{e}\|_1 - 2 \cdot \mathbf{1}_s^T |\mathbf{e}|$, with $\mathbf{1}_s$ denoting a vector of length k with the first s elements being 1-es and the remaining elements being 0-es.

Plugging this inflated group into our definition of the group C_e in (2.12), we arrive at

$$C_{\mathbf{e}} \subseteq \left\{ \mathbf{e} \neq \mathbf{0} \,\middle|\, |\mathbf{e}| \leq \frac{M}{M+1} \cdot \|\mathbf{e}\|_1 \text{ and } \|\mathbf{e}\|_1 - 2 \cdot \mathbf{1}_s^T |\mathbf{e}| \leq 0 \right\}. \tag{2.13}$$

The attractive part about this set is that it no longer depends directly on $\mathbf{D}$, but only on its Mutual-Coherence, and furthermore, it does not depend on α any more, allowing us to obtain a general bound applicable to all vectors.

Our next simplification uses the observation that if we have found any vector $\mathbf{e} \in C_e$, then for every real number z, $z \cdot \mathbf{e}$ is also in C_e. Since we are only interested in the question weather C_e is empty or not, we choose to intersect this set with the ℓ_1-ball, i.e., all the vectors $\mathbf{e}$ that have an ℓ_1-norm of 1,

$$C_{\mathbf{e}} \subseteq \left\{ \mathbf{e} \neq \mathbf{0} \,\middle|\, |\mathbf{e}| \leq \frac{M}{M+1} \text{ and } 1 - 2 \cdot \mathbf{1}_s^T |\mathbf{e}| \leq 0 \text{ and } \|\mathbf{e}\|_1 = 1 \right\}. \tag{2.14}$$

Our purpose now is to derive conditions for this set to be empty. The first constraint is that every element of $\mathbf{e}$ is smaller than $M/(1+M)$. In order to satisfy the second constraint, the energy of the vector $\mathbf{e}$ should be concentrated in the elements in the support. Using the bound on the size of each element, for the second constraint to be satisfied (and the set being not empty) it is required that

$$1 - 2 \cdot s \cdot \frac{M}{1+M} \leq 0 \Rightarrow s \geq \frac{M+1}{2M}.$$

This means that as long as $s < \frac{M+1}{2M}$, the set is empty, and therefore C_e is also empty. This implies that under this condition, a vector β different from α, satisfying

$\mathbf{D}\beta = \mathbf{x}$ and with $\|\beta\|_1 \leq \|\alpha\|_1$ does not exist. This leads to the BP actually arriving at α as the result of its minimization, which means that when $s < \frac{M+1}{2M}$, the basis pursuit is guaranteed to recover α exactly.

Stating this formally, we obtain the equivalence theorem for the Basis Pursuit:

Theorem 2.5 (BP Equivalence)**.** *Given a vector* $\mathbf{x}$ *with a representation* $\mathbf{D}\alpha = \mathbf{x}$*, and assuming that* $\|\alpha\|_0^0 < 0.5\,(1 + 1/M)$*, then BP is guaranteed to find the* sparsest *solution* [**14, 29**].

Obviously, this claim is exactly identical to the equivalence claim for the OMP, meaning that the bound to guarantee success of the two algorithms is identical. However, it does not mean that the two algorithms are perform the same, as they are different in general (specifically, when this bound is not met, or in the noisy case). It is hard to tell which is better.

Furthermore, the bounds presented here are worst-case performance guarantees. When analyzing actual performance, these bounds are usually too pessimistic. Analysis of average performance has also been done. In a series of papers [**5, 12, 13, 18, 19, 53**], bounds are developed from a probabilistic point of view, showing that up to some bounds, the actual vector may be recovered with a probability close to 1. These bounds are proportional to $O(1/M^2)$, instead of $O(1/M)$.

4. Theoretical guarantees - stability for (P_0^ϵ)

4.1. Stability versus uniqueness

Consider the following problem, which is very similar to the one we dealt with above: Alice chooses a sparse vector α, multiplies it by $\mathbf{D}$, and obtains $\mathbf{x} = \mathbf{D}\alpha$. Alice wants to send the signal $\mathbf{x}$ to Bob, but unfortunately, unlike before, the transmission line is imperfect. Therefore, noise is added to the signal, and Bob receives the signal $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where we know that the norm of $\mathbf{v}$ is bounded, $\|\mathbf{v}\|_2 \leq \epsilon$.

Bob would like to recover α. Since $\mathbf{y}$ is noisy, there is no sense in solving the exact equation $\mathbf{y} = \mathbf{D}\alpha$, and thus Bob would attempt to solve

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{x}\|_2 \leq \epsilon.$$

Due to the noise being added, there might be somewhere in the sphere around $\mathbf{x}$ a vector with a sparser representation than α. If not, α itself will be chosen. Therefore, we know that $\|\hat{\alpha}\|_0 \leq \|\alpha\|_0$. Can we hope that $\hat{\alpha} = \alpha$? If not, can we at least hope that they are similar or close-by?

It turns out (we will not show it hear, but the slides present it through a simple and illustrative example) that exact equality in the recovery, $\hat{\alpha} = \alpha$, is impossible to expect in general, and thus we will turn to discuss *stability*, which is a claim about the proximity between Alice's $\mathbf{a}$, and Bob's proposed solution $\hat{\alpha}$.

Suppose we created the true signal with an s-sparse vector of coefficients α_1, and the solution we got is an s-sparse (or sparser) vector α_2. These vectors satisfy $\|\mathbf{D}\alpha_1 - \mathbf{y}\|_2 \leq \epsilon$ and $\|\mathbf{D}\alpha_2 - \mathbf{y}\|_2 \leq \epsilon$, which means that both of these solutions are inside a sphere around $\mathbf{y}$ with radius ϵ. As these two points are inside the sphere, the maximal distance between the two is twice the radius, and therefore

$$\|\mathbf{D}\,(\alpha_1 - \alpha_2)\,\|_2 \leq 2\epsilon. \tag{2.15}$$

On the other hand, we can use the RIP property. The vector $\alpha_1 - \alpha_2$ is $2s$-sparse at most (if the vectors have different supports), and we have the RIP property

$$(1-\delta_{2s})\,\|\alpha_1-\alpha_2\|_2^2 \le \|\mathbf{D}\,(\alpha_1-\alpha_2)\,\|_2^2. \tag{2.16}$$

Combining (2.15) (by squaring both sides) with (2.16) leads to the relation

$$(1-\delta_{2s})\,\|\alpha_1-\alpha_2\|_2^2 \le 4\epsilon^2 \tag{2.17}$$

Rearranging the terms leads to the following bound on the distance between the two solutions

$$\|\alpha_1-\alpha_2\|_2^2 \le \frac{4\epsilon^2}{1-\delta_{2s}} \le \frac{4\epsilon^2}{1-(2s-1)\,M}, \tag{2.18}$$

where in the last step we have used the relationship between the RIP and the Mutual-Coherence $\delta_s \le (s-1)\,M$. We have obtained a bound on the distance between the two representation vectors, and we can see that this bound is in the order of the noise. It is slightly higher, since the denominator is smaller than 1, but it is not far-off. This is what we strived to show – stability.

The above analysis assumes that the power of the noise is bounded, $\|\mathbf{v}\|_2 \le \epsilon$. However, there was no assumption made as to the shape of the noise, and the noise might have taken any shape to work against our analysis and algorithm. For this reason, the bounds we obtain is weak, showing no effect of noise attenuation.

An alternative approach could be to model the noise as a random, i.i.d Gaussian vector, the elements of which are randomly drawn as $v_i \sim N\big(0,\sigma^2\big)$. When such a model is assumed, we can expect to get much better stability bound: The norm of the error is expected to be close to (but not exactly) $s\cdot\sigma^2$ instead of $n\cdot\sigma^2$, which is a much better result. However, this claim will no longer be deterministic, but only probabilistic, with a probability very close to 1.

5. Near-oracle performance in the noisy case

So far we discussed the theoretical solution of (P_0^ϵ). However, we know that this task is impossible, and approximation algorithms are used, such as the OMP and the BP. Very similar analysis exists that shows stability of these algorithms – a successful recovery of the representation vector. We will not show this here, and we refer the readers to [**15, 16, 51**]

As above, there is a distinction between adversive noise, which leads to worst-case performance bounds similar to the one given in Equation (2.18). When random noise is assumed, these bounds improve dramatically. In that respect, it is customary to compare the performance of pursuit techniques to the one obtained by an oracle - a pursuit technique that *knows the exact support*. The oracle's error is typically $O(s\sigma^2)$ (as described above), and practical pursuit methods show bounds that are this expression multiplied by a constant (larger than 1) and a *logk* factor [**1, 6**].

LECTURE 3

Sparse and redundant representation modelling

1. Modelling data with sparse and redundant representations

1.1. Inherent low-dimensionality of images

Let us think of the following virtual experiment: Consider images $\mathbf{x}$ of size $\sqrt{n}\times\sqrt{n}$ (e.g., $n = 400$ for 20×20 images). Each such image can be thought of as a point in the $\mathbb{R}^n$ space. Once we accumulate many such images, we will have many such points in $\mathbb{R}^n$. However, no matter how many images we accumulate, the space $\mathbb{R}^n$ is not going to be completely filled, but rather many empty areas will remain. Furthermore, the local density of the images from one place to another is expected to vary considerably.

The bottom line from these two observations is that we believe that the true dimension of the images is not n. Rather, we believe that the images form a low-dimensional manifold in $\mathbb{R}^n$, with spatially varying densities. It is our task to model this manifold reliably.

The tool with which we represent the manifold is the Probability Density Function (PDF) $Pr(\mathbf{x})$. This function tell us for each $\mathbf{x} \in \mathbb{R}^n$ how probable it is to be an image. Obviously, this function should be non-negative and when summed over the entire space it should sum to 1. In order to fulfill our assumption of the images forming a low-dimensional manifold, the PDF $Pr(\mathbf{x})$ should be zero for most of the cube $[0, 255]^n$.

The function $Pr(\mathbf{x})$ is called the *Prior* as it reflects our prior knowledge about what images should look like. Unfortunately, characterizing this function explicitly is probably an impossible task, which leads us to look for ways to simplify it.

1.2. Who needs a prior?

Let us consider one simple example in which we need a prior. Suppose we would like to obtain an image $\mathbf{x}$ (lying on the low-dimensional manifold). Unfortunately, we are only able to obtain its noisy version, $\mathbf{z} = \mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is a random noise vector with a limited magnitude $\|\mathbf{v}\|_2 \leq \epsilon$. Using the prior over the image $\mathbf{x}$, we can now use it to try and recover $\mathbf{x}$, by solving the problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} Pr\left(\mathbf{x}\right) \;\; s.t. \; \|\mathbf{x} - \mathbf{z}\|_2 \leq \epsilon.$$

In effect, we are looking for the most probable vector $\mathbf{x}$ (according to our prior), which is also close enough to $\mathbf{z}$ to explain it. This approach is a very known one – it is called the Maximum A'posteriori Probability (MAP) approach.

This simple example is just one of many that require a good model (prior). All inverse problems, such as denoising, deblurring, super-resolution, inpainting, demosaicing, tomographic reconstruction, single image scale-up and more, require a good image prior, in order to be able to differentiate between the true image content and the degradation effects. Compression algorithms require a model in order to

be able to find the most compact representation of the image. Anomaly detection requires a model of the image, in order to pick up on anything not conforming to the model. The number of problems requiring a good image model is ever growing, which leads us to the natural question - what is this prior $Pr(\mathbf{x})$? Where can we get it from?

1.3. Evolution of priors

The topic of a proper choice of $Pr(\mathbf{x})$ for images has been at of the foundation of the research in image processing since its early days, and many models have been suggested. Since it is not possible to characterize $Pr(\mathbf{x})$ explicitly, some form of computing it is suggested. Usually, instead of suggesting $Pr(\mathbf{x})$ directly, its minus log is used, $G(\mathbf{x}) = -\log(Pr(\mathbf{x}))$, thereby giving a penalty of 0 to the "perfect" image, and some positive penalty to any other, a larger value as the probability of the image decreases.

There has been an evolution of choices for $G(\mathbf{x})$ through the years. It started with the ℓ_2-norm, and the very simple energy term, $G(\mathbf{x}) = \lambda\|\mathbf{x}\|_2^2$. It then progressed to smoothness terms, demanding low energy of some derivatives, $G(\mathbf{x}) = \lambda\|\mathbf{L}\mathbf{x}\|_2^2$, with $\mathbf{L}$ being the Laplacian operator. A further step was made by adaptively demanding this low energy, while allowing pixels detected as edges to contain high energy in the derivative, $G(\mathbf{x}) = \lambda\|\mathbf{L}\mathbf{x}\|_{\mathbf{W}}$, where $\mathbf{W}$ is a pre-computed diagonal matrix, assigning different weights to different pixels.

The next major step was the abandonment of the convenient ℓ_2 norm and the progression to robust statistics, $G(\mathbf{x}) = \lambda\rho\{\mathbf{L}\mathbf{x}\}$. One of the most known priors from this family is the total variation, $G(\mathbf{x}) = \lambda\| \, |\nabla\mathbf{x}| \, \|_1$, suggested in the early 1990's. [**44**].

Around the same time, the prior of Wavelet sparsity $G(\mathbf{x}) = \lambda\|\mathbf{W}\mathbf{x}\|_1$ started evolving in the approximation theory and the harmonic analysis communities [**11, 17, 38**]. The idea behind this prior is that looking at the wavelet coefficients of an image, they should be sparse - most of them should be zeros or very close to zero.

The prevailing model in the last decade, which is an evolution of all models described so far, is the one involving sparse and redundant representations. Similarly to the wavelets sparsity, this model assumes sparse coefficients, however, it no longer relies on fixed transforms. In the reminder of these course, we will focus on this model and show its applicability to image processing. Of course, this model is not the last in the evolution, and in the coming years new models will be suggested, and they will perform even better (hopefully).

2. The *Sparseland* prior

Our model (and any other model) represents a belief - how we believe an image should look like, or what are the rules it obeys. The sparse and redundant representation model believes that all images are generated by a "machine" that is able to create $\sqrt{n}\times\sqrt{n}$ images. Inside this machine there exists a dictionary $\mathbf{D}_{n\times k}$, and each of the columns of this dictionary is a prototype signal – an atom.

Each time this machine is required to generate a signal, a vector α is (randomly) chosen. Not any vector can be selected, however, as this vector must be sparse - it may only contain a small number L of non-zeros, at random locations and with random values, which we will assume are Gaussian. Then, the result signal is created by multiplying the dictionary by the sparse vector of coefficients: $\mathbf{x} = \mathbf{D}\alpha$.

We call this machine *Sparseland* and we assume that all our signals in the family of interest are created this way.

The *Sparseland* model has some interesting properties. It is simple, as any signal generated is a linear combination of only a small number of atoms. Instead of describing the signal using n numbers to represent the signal, we only need $2L$ values – the locations of the non-zeros and their values. These signals are thus very compressible.

On the other hand, this source is also very rich. We have many (combinatorial) ways to choose subsets of L atoms, and can therefore create a wealth of signals using this model. As a matter of fact, the signals generated are a union of many low-dimensional Gaussians.

The last reason, which is also the actual reason this model is indeed being used, is that it is in fact familiar and has been used before, in other contexts and perhaps in a simplified way, like wavelets and JPEG. This progress have been observed in several trends recently. For example, the transition from JPEG to JPEG2000 is essentially a move from the ℓ_2-norm and taking the leading coefficients, to the notion of sparsity of wavelet coefficients. The progress from Wiener filtering (Fourier) to robust estimation also promotes sparsity. Similar trend exists in the evolution of wavelets, from unitary transforms to over-complete transforms with concepts and tools such as frames, shift-invariance, steerable wavelets, contourlets, curvelets and more, all point to the direction of redundancy in the representation [**39, 10, 46, 45**]. The ICA (independent component analysis) transform was also shown to be tied to sparsity and independence between the coefficients.

Another field progressing towards an extensive usage of sparse and redundant representation modeling is that of machine learning. Methods such as LDA, PCA, SVM and feature-selection methods have all been extended or improved recently by the introduction of sparse and redundant representations.

All of these directions clearly point to *Sparseland*. It is here, and we are going to use it. In order to do that, however, we must define it clearly, analyze it, understand it, and find out how it should be applied to the various tasks at hand.

One more thing we must take into account in our model is the model noise. While we expect our signals to look like they have been created as $\mathbf{D}\alpha$ with a sparse α, we may want to allow for some perturbations from this exact model, with the understanding that, just like any other model, this model is not perfect or exact. For this purpose, we assume that signals are created as $\mathbf{y} = \mathbf{D}\alpha + \mathbf{e}$, with $\mathbf{e}$ the model noise with bounded energy $\|\mathbf{e}\|_2 \leq \epsilon_0$.

3. Processing *Sparseland* signals

The model we will be considering in this section is the exact model, i.e. $\mathbf{x} = \mathbf{D}\alpha$. We will assume from now on that there is no model noise, i.e., $\epsilon_0 = 0$. This does not mean that we work only on clean signals; It only means that true signals can indeed be created exactly by multiplying the dictionary by a sparse vector of coefficients. Now, we ask – How do we process such signals?

3.1. Transform

Suppose we have a *Sparseland* signal $\mathbf{y}$, and we want to design a transform for it, similarly to the wavelet transform or the Fourier transform. What would we like such a transform to do?

First, we would like it to be invertible - we do not want to lose information by applying the transformation. We would also like the transformation to be compact, i.e., for its energy to be concentrated in a small number of coefficients. Lastly, we would also want the coefficients to be independent of each other.

We can propose a very simple transformation that follows all these requirements – let us transform $\mathbf{y}$ into the sparsest vector α such that $\mathbf{D}\alpha = \mathbf{y}$,

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \mathbf{D}\alpha = \mathbf{y}.$$

We know that such an α indeed exists, making this transform possible, since it is our model's assumption that $\mathbf{y}$ was generated by a sparse vector multiplying the dictionary.

As the dictionaries used are redundant, α is usually longer than $\mathbf{y}$, making the transform suggested also redundant. It is also invertible, as multiplying $\mathbf{D}$ by α brings us back to our original signal. Lastly, we observe that the energy of the transform is concentrated in only a few non-zero coefficients, as α is sparse, and those non-zeros are independent.

3.2. Compression

Suppose we now try to compress the *Sparseland* signal $\mathbf{y}$. In order to allow compression, we allow some error ϵ between the compressed signal and the original one. Changing this value will create different rates of compression, thus creating the rate-distortion curve for the compression scheme. The compression can be done by solving

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{y}\|_2^2 \leq \epsilon^2,$$

which means searching for the smallest number of coefficients that can explain the signal $\mathbf{y}$ with an error of less than ϵ. Note that the final rate-distortion curve is obtained from taking into account the further error introduced by quantizing the non-zero coefficients in α.

3.3. Denoising

Given a noisy *Sparseland* signal $\mathbf{z} = \mathbf{y} + \mathbf{v}$, created by the addition of additive noise $\mathbf{v}$ (with known power $\|\mathbf{v}\|_2 = \epsilon$) to our original signal, we would like to recover $\mathbf{y}$ as accurately as possible.

Since we know the signal $\mathbf{y}$ is sparse over the dictionary $\mathbf{D}$ and we expect the noise to not be sparse over the same dictionary (since it is not a "real" image), the following task can be proposed:

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2,$$

with the final solution being $\hat{\mathbf{y}} = \mathbf{D}\hat{\alpha}$. This denoising method works quite well in practice, as we will see later on in the lectures.

There is a subtle point to address regarding denoising. Our model assumes that the signals we operate on can be indeed represented exactly using a sparse representation over the dictionary. However, for real-world signals, this is not exactly the case, and there are some model mismatches. The denoising algorithm will remove these true details from the signal, along with the noise. This implies that if the noise is very weak, such that its power is in the order of the model mismatch, the denoising result might actually be worse than not denoising at all. This is because the price of removing the model mismatch may be greater than the price of not removing the noise. Usually, when the standard deviation of the noise

is 5 grey levels or more (when the image is in the range $[0, 255]$), it is considered to be stronger than the model mismatches, and denoising the image indeed improves its quality.

3.4. General inverse problems

As a generalization of the denoising problem, we now assume that the *Sparseland* signal is given to us after a degradation operator $\mathbf{H}$ and an additive noise, $\mathbf{z} = \mathbf{H}\mathbf{y} + \mathbf{v}$. The operator $\mathbf{H}$ can be any linear operator that can be written as a matrix, representing blur, projection, down-scaling, masking and more.

We can try to recover $\mathbf{y}$ by looking for the sparsest α that when multiplied by the dictionary and then by $\mathbf{H}$ is close enough to the input signal $\mathbf{z}$. This vector is found by solving the minimization problem:

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{H}\mathbf{D}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2,$$

and again, the final solution is obtained by $\hat{\mathbf{y}} = \mathbf{D}\hat{\alpha}$. The minimization is carried out in practice by defining the equivalent dictionary $\tilde{\mathbf{D}} = \mathbf{H}\mathbf{D}$, and solving the same minimization problem we have already encountered:

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\tilde{\mathbf{D}}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2,$$

There is a wide range of problems that can be solved using this mechanism: deblurring, inpainting, demosaicing, super-resolution, tomographic reconstruction, image-fusion and many more.

3.5. Compressed-sensing

Suppose that we want to sample the *Sparseland* signal $\mathbf{y}$, such that we use much less than n samples. There may be various reasons for this desire, such as the signal being too long, having limited time to sample it, each sample being expensive, and so on. Instead, we propose to sample noisy projections of it, using a known projection matrix $\mathbf{P}$, making our signal $\mathbf{z} = \mathbf{P}\mathbf{y} + \mathbf{v}$, with $\|\mathbf{v}\|_2 = \epsilon$ representing the sampling noise. Again, we suggest to recover the original signal by solving

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{P}\mathbf{D}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2,$$

and multiplying the found vector of coefficients by the dictionary, $\hat{\mathbf{y}} = \mathbf{D}\hat{\alpha}$. This problem again is of the same form seen above, considering the effective dictionary as $\tilde{\mathbf{D}} = \mathbf{P}\mathbf{D}$. The Compressed Sensing field deals with questions like the minimal number of rows in $\mathbf{P}$ (the minimal number of projections) needed to recover the original signal $\mathbf{y}$ with sufficient accuracy, and what kind of $\mathbf{P}$ should be chosen.

3.6. Morphological-component-analysis

The MCA task [**47, 48**] deals with the case where a given signal $\mathbf{z} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{v}$ is a mixture of two *Sparseland* signals $\mathbf{y}_1$ and $\mathbf{y}_2$, each created from a different dictionary $\mathbf{D}_1$ and $\mathbf{D}_2$ respectively, with noise also being added. The goal is to recover each of the original signals $\mathbf{y}_1$ and $\mathbf{y}_2$, assuming their dictionaries $\mathbf{D}_1$ and $\mathbf{D}_2$ are known.

Since both signals should have a sparse representations, it is suggested to solve the minimization problem

$$\hat{\alpha_1}, \hat{\alpha_2} = \arg\min_{\alpha_1, \alpha_1} \|\alpha_1\|_0^0 + \|\alpha_2\|_0^0 \ s.t. \ \|\mathbf{D}_1\alpha_1 + \mathbf{D}_2\alpha_2 - \mathbf{z}\|_2^2 \leq \epsilon^2,$$

and recover the original signals using $\hat{\mathbf{y}_1} = \mathbf{D}_1\hat{\alpha_1}$ and $\hat{\mathbf{y}_2} = \mathbf{D}_2\hat{\alpha_2}$.

By concatenating the vectors α_1 and α_2 vertically into a single vector α, and concatenating the two dictionaries by putting them side by side $\mathbf{D} = [\mathbf{D}_1\mathbf{D}_2]$, it turns out that we have already seen this task:

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2.$$

3.7. Applications: Summary

We have gone over several applications that can be solved using the *Sparseland* model. There are many more that we can envision using this model – encryption, watermarking, scrambling, target detection, recognition, feature extraction and more. What is common to all these problems is the need to solve a variant of the problem (P_0^ϵ),

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0^0 \ s.t. \ \|\mathbf{D}\alpha - \mathbf{z}\|_2^2 \leq \epsilon^2.$$

This is a problem we have already seen and discussed - we know it is NP-hard, but we have seen several algorithms to approximate its solution and some guarantees on the quality of their solutions. The conclusion is that signal processing for *Sparseland* signals is reasonable and practical.

LECTURE 4

First steps in image processing

In this lecture we will start working on image processing applications that rely on the *Sparseland* model. We will consider image deblurring, image denoising, and image inpainting. Several other, more advanced applications, using the tools shown in the lecture, will be given in the following (and last) lecture. More on these applications can be found in [**22**].

1. Image deblurring via iterative-shrinkage algorithms

1.1. Defining the problem

The deblurring problem is one of the most fundamental problems in image processing: A high-quality image $\mathbf{x}$ undergoes blurring, be it due to atmospheric blur, camera blur, or any other possible source. We assume this blur is known (could be space variant). We measure a noisy (white iid Gaussian) version of the blurred image. The noise $\mathbf{v}$ has a known standard deviation. Thus, the image we actually obtain $\mathbf{y}$ is related to the ideal image $\mathbf{x}$ through

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}, \tag{4.1}$$

and our task is to recover $\mathbf{x}$ given $\mathbf{y}$. Since we are dealing with the *Sparseland* model, we assume that the image $\mathbf{x}$ has a sparse representation over a carefully chosen dictionary, and therefore we will try to solve

$$\hat{\alpha} = \arg\min_{\alpha} \frac{1}{2}\|\mathbf{H}\mathbf{D}\alpha - \mathbf{y}\|_2^2 + \lambda \cdot \rho\left(\alpha\right). \tag{4.2}$$

Here, the function ρ is sparsity promoting penalty, such as the ℓ_1-norm, or something similar (see the end of lecture 1). Since we are not interested in $\hat{\alpha}$ in itself but in the recovered image, we finish the processing with $\hat{\mathbf{x}} = \mathbf{D}\hat{\alpha}$.

Let us define the ingredients in Equation (4.2). The vector $\mathbf{y}$ is the given degraded image, and $\mathbf{H}$ is the known blur operator. The dictionary $\mathbf{D}$ should be chosen by us. In the experiments we report here, we follow the work by Figueiredo and Nowak [**26, 27**] and use the 2D un-decimated Haar wavelet transform with 2 resolution levels. This is a global dictionary with a redundancy factor of $7:1$. As for the function ρ, we use an approximation of the ℓ_1 norm, with a slight smoothing,

$$\rho\left(\alpha\right) = |\alpha| - S \cdot \log\left(1 + \frac{\alpha}{S}\right). \tag{4.3}$$

The function ρ is a scalar function, operating on each entry of α individually, and summing the result over all entries. The benefit of using this function rather than the ℓ_1 norm directly is that it is smooth and derivable in the origin (and everywhere else) and is therefore much more convenient to work with.

Lastly, the value of λ is set manually to get the best possible result. There are algorithms that can find its value for optimal performance automatically. However, we will not discuss those here and proceed with the manually chosen value.

There is a wide knowledge in the area of continuous optimization, and it seems like once we have written down our penalty function, we can use a wide variety of methods or even general purpose optimization software packages (e.g., Matlab, MOSEK) for its solution. However, such general purpose solvers typically perform very poorly for this specific optimization task. One possible reason for this poor performance is the fact that these solvers disregard the knowledge about the sparsity of the desired solution. A second possible reason can be understood from look at the gradient of our penalty function,

$$\bigtriangledown f(\alpha) = \mathbf{D}^T(\mathbf{D}\alpha - \mathbf{y}) + \lambda \cdot \rho'(\alpha)$$

and the Hessian

$$\bigtriangledown^2 f(\alpha) = \mathbf{D}^T\mathbf{D} + \lambda \cdot \rho''(\alpha).$$

We must remember that α will contain many zero elements, and few non-zero elements in *unknown locations*. In the locations of the zeros, the value of ρ'' will be very large, while for the non-zeros, these will be small numbers. Therefore, the Hessian's main diagonal will have a high contrast of values, causing this matrix to be of very high condition-number, hurting the performance of general-purpose solvers. Furthermore, as we do not know in advance the locations of these non-zeros (this is actually the goal of the optimization - recovering the support), we cannot even pre-condition the Hessian. This forces us to seek for tailored optimization methods that will take advantage of the expected sparsity in the solution.

How should we do that? In lecture 1 we have seen that when the matrix $\mathbf{D}$ is unitary, the solution is obtained by a simple scalar-wise shrinkage step $S_{\rho,\lambda}\left(\mathbf{D}^T\mathbf{y}\right)$. When $\mathbf{D}$ is not unitary, we would like to propose a similarly simple solution. This idea leads to a family of algorithms known as "Iterative Shrinkage" [**25**]. There are several such algorithms, and here we will present one of them.

1.2. The parallel coordinate descent (PCD) method

An interesting iterative shrinkage algorithm emerges from the classic coordinate descent (CD) algorithm. Put generally, the CD operates as follows: Given a function to minimize with a vector as the unknown, all entries but one are fixed, and the remaining entry is optimized for. Then, this coordinate is fixed, and a different entry is optimized. This is done serially over all entries, and in several iterations, until convergence.

In our problem, solving for one entry at a time considers only the atom $\mathbf{d}_j$ and the coefficient α_j. Taking it out of the current support by setting $\alpha_j = 0$, we get that the current error in representation is $\mathbf{e}_j = \mathbf{y} - \mathbf{D}\alpha$. We now attempt to best reduce this error using only the atom $\mathbf{d}_j$, by choosing the optimal value for α_j, i.e., minimizing

$$f(\alpha_j) = \frac{1}{2}\|\alpha_j \cdot \mathbf{d}_j - \mathbf{e}_j\|_2^2 + \lambda\rho(\alpha_j).$$

This is a scalar optimization problem that can be solved using shrinkage on the scalar value $\mathbf{d}_j^T\mathbf{e}_j$. The shrinkage is applied by

$$\alpha_j^{OPT} = S_{\rho,\lambda/\|\mathbf{d}_j\|^2}\left(\mathbf{d}_j^T\mathbf{e}_j\right),$$

where S denotes the shrinkage operation – its shape is dictated by the choice of ρ and λ/c. Notice that the shrinkage is done differently to each entry, based on the norm of the j-th atom.

For low-dimensional problems ($k \approx 100$) it makes sense to solve our problem this way exactly. However, in the image deblurring problem, the unknown contains many thousands entries, and it is impossible to use the above algorithm, as it requires to extract one column at a time from the dictionary. The solution to this problem is the Parallel-Coordinate-Descent (PCD) algorithm. The idea is the following: Compute all the k descent trajectories $\{v_j\}_{j=1}^{k}$, each considering one coordinate optimization. Since all are descent directions, their sum is also a descent direction. The PCD takes this sum and uses this for the current iteration. The problem with this direction is that it is not known how far to go along it. Therefore, a 1D line-search is needed, and the overall iteration step is written as

$$\alpha_{k+1} = \alpha_k + \mu \cdot \left[S_{\rho,\mathbf{Q}\lambda} \left(\mathbf{Q}\mathbf{D}^T \left(\mathbf{y} - \mathbf{D}\alpha_k \right) - \alpha_k \right) \right], \tag{4.4}$$

with $\mathbf{Q} = \text{diag}^{-1}\left(\mathbf{D}^T\mathbf{D}\right)$. μ is the found step-size in the line-search step.

1.3. Results

Figure 1 shows the deblurring result obtained by the PCD algorithm after 10 iterations. More details on this experiment are given in the accompanying slides.

FIGURE 1. Debluring. Left: The original image, Middle: the measured (blurred and noisy) image, and Right: the PCD deblurring result.

2. Image denoising

The image denoising problem can be viewed as a special case of the image deblurring problem discussed above. For this to be evident, all we need to do is select $\mathbf{H} = \mathbf{I}$ (i.e., no blur at all). In this section we shall present two very different ways to handle this problem.

2.1. Global shrinkage

Embarking from the above-described deblurring algorithm, we can suggest the following simplified denoising method: Instead of the iterative mechanism, we can apply the thresholding algorithm, as follows

$$\hat{\mathbf{x}} = \mathbf{D} S_T \left(\mathbf{D}^T \mathbf{y} \right). \tag{4.5}$$

Of course, since we are using the redundant Haar dictionary, we should normalize its columns (since they are not ℓ_2 normalized) in order to have a single scalar threshold for all atoms. Put differently, this leads to

$$\hat{\mathbf{x}} = \mathbf{D}\mathbf{W}S_T\left(\mathbf{W}^{-1}\mathbf{D}^T\mathbf{y}\right). \tag{4.6}$$

The only missing ingredient is determining the value of the threshold T. Experimenting with this algorithm, different values of T will have a different quality of denoising. As an example, on the well known image "barbara" with noise with standard deviation $\sigma = 20$, the optimal value of T is around 55. The denoised image using this method is more than 5 dB better in PSNR than the noisy image. This result is shown in Figure 2.

FIGURE 2. Global denoising. Left: The original image, Middle: the measured (noisy) image, and Right: the global denoising result.

2.2. From global to local processing

While the above result is good, it is definitely not the best that can be achieved. We will now turn to a much better method (and see an even better one in the next lecture).

When discussing algorithms for the solution of the (P_0^ϵ) problem, we saw that very large signals could pose a computational problem. Therefore, when we work on the image as a whole, we are forced to use only simple algorithms, such as (iterative) shrinkage. Also, when using dictionaries for an entire image, the result may be sparse when compared to the overall number of atoms, but we still require a large number of coefficients. These two observations lead us to attempt operating on small image patches instead.

When processing a large image, we claim that every $N \times N$ patch (e.g., 8×8) in it has a sparse representation over a dictionary. Furthermore, we use full overlaps between the patches. By operating this way, we in fact force shift-invariance sparsity. In order to write this formally, we define the operator $\mathbf{R}_{ij}$, which extracts an $N \times N$ patch from the image around pixel (i, j). Now, the minimization task to solve is [**30, 31, 23**]

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x},\{\alpha_{ij}\}_{ij}} \left[\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \mu\sum_{ij}\|\mathbf{R}_{ij}\mathbf{x}-\mathbf{D}\alpha_{ij}\|_2^2\right] \; s.t. \; \|\alpha_{ij}\|_0 \le L. \tag{4.7}$$

The first term requires a proximity between the recovered image and the input image, a direct consequence of the log-likelihood. The second term requires that each

patch in the reconstructed image can be represented well as a linear combination of atoms from the dictionary $\mathbf{D}$. The constraint forces all of these linear combinations to be sparse.

At this stage, the dictionary we will be using for this algorithm is the over-complete DCT dictionary with 256 atoms, each representing an 8×8 patch. This dictionary is separable, allowing it so be applied efficiently.

How can we perform the minimization in (4.7)? We will take a coordinate descent approach. First, we will fix $\mathbf{x} = \mathbf{y}$, and solve for $\{\alpha_{ij}\}_{ij}$. The first term disappears, and we are left with $M \cdot N$ different minimization tasks:

$$\hat{\alpha}_{ij} = \arg\min_{\alpha} \|\mathbf{R}_{ij}\mathbf{x} - \mathbf{D}\alpha\| \ s.t. \ \|\alpha\|_0 \leq L. \tag{4.8}$$

We have seen this problem before - this is the sparse coding problem, and can be solved by any pursuit algorithm (e.g., OMP). Once we have found $\{\alpha_{ij}\}_{ij}$, we fix them, and recover $\mathbf{x}$. This is a least-squares problem and a little bit of algebra leads to a closed-form solution

$$\hat{\mathbf{x}} = \left[\mathbf{I} + \mu \sum_{ij} \mathbf{R}_{ij}^T \mathbf{R}_i j\right]^{-1} \left[\mathbf{y} + \mu \sum_{ij} \mathbf{R}_{ij}^T \mathbf{D}\hat{\alpha}_i j\right]. \tag{4.9}$$

While this expression seems complicated, it is actually simple to implement. While the operator $\mathbf{R}_{ij}$ extracts a patch around the pixel (i, j), the operator $\mathbf{R}_{ij}^T$ returns a patch to the location (i, j). Therefore, the left matrix is in fact a diagonal one, indicating for each pixel the amount of patches it belongs in, serving as normalization.

Therefore, the entire denoising process is done by extracting each patch, performing sparse coding, and getting a denoised patch instead. The denoised patch is returned to its original position, with aggregation performed in the overlaps area. Then, each pixel is normalized by the number of patches it belongs too (which is generally N^2, apart from the borders of the image).

It turns out that this very simple method performs about 2.5dB better than the global thresholding show previously. Furthermore, this algorithm can be implemented very efficiently on parallel processors. We will see this local processing approach in several more applications. The result obtained is shown in Figure 3.

FIGURE 3. Local denoising. Left: The original image, Middle: the measured (noisy) image, and Right: the local denoising result.

3. Image inpainting

Assume that a signal has been created as $\mathbf{x} = \mathbf{D}\alpha_0$ using a very sparse vector of coefficients α_0. Unfortunately, we do not get $\mathbf{x}$ directly, but rather obtain $\tilde{\mathbf{x}}$, which is the same signal $\mathbf{x}$, but with some samples removed. We shall assume that we know which samples are missing. Each missing sample from $\mathbf{x}$ implies one lost equation in the system $\mathbf{x} = \mathbf{D}\alpha_0$. Therefore, we remove these corresponding rows from the dictionary $\mathbf{D}$, obtaining the dictionary $\tilde{\mathbf{D}}$. Now, we may try to solve

$$\hat{\alpha} = \arg\min_{\alpha} \|\alpha\|_0 \ s.t. \ \tilde{\mathbf{x}} = \tilde{\mathbf{D}}\alpha. \tag{4.10}$$

If α_0 was sparse enough, it will also be the solution of this modified problem. Then, by computing $\mathbf{D}\hat{\alpha}$ we will recover the original signal completely.

Using the above intuition, we now turn to extend the local denoising algorithm into an inpainting one. The minimization task here is very similar to the one we had for the denoising (4.7), with the needed modification to take into account the masking operator [**30, 31, 36**],

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x},\{\alpha_{ij}\}_{ij}} \left[\frac{1}{2}\|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2^2 + \mu \sum_{ij} \|\mathbf{R}_{ij}\mathbf{x} - \mathbf{D}\alpha_{ij}\|_2^2 \right] \ s.t. \ \|\alpha_{ij}\|_0 \le L, \tag{4.11}$$

where we now require that $\mathbf{x}$ is close to $\mathbf{y}$ only in the locations of un-masked pixels.

Again this minimization will be done using the coordinate descent approach. First, we compute $\mathbf{X} = \mathbf{M}^T\mathbf{y}$ (e.g., by zero filling), and do sparse coding for each patch, using the matching pursuit

$$\hat{\alpha}_{ij} = \arg\min_{\alpha} \|\mathbf{M}_{ij}\left(\mathbf{R}_{ij}\mathbf{x} - \mathbf{D}\alpha\right)\| \ s.t. \ \|\alpha\|_0 \le L, \tag{4.12}$$

with $\mathbf{M}_{ij}$ the appropriate masking for each patch. Once the sparse coding is finished, the result image is again reconstructed by aggregating the patches in the overlaps area, normalizing by the number of patches in each pixel,

$$\hat{\mathbf{x}} = \left[\mathbf{M}^T\mathbf{M} + \mu \sum_{ij} \mathbf{R}_{ij}^T\mathbf{R}_ij\right]^{-1} \left[\mathbf{M}^T\mathbf{y} + \mu \sum_{ij} \mathbf{R}_{ij}^T\mathbf{D}\hat{\alpha}_ij\right]. \tag{4.13}$$

Figure 4 presents inpainting results obtained using this algorithm.

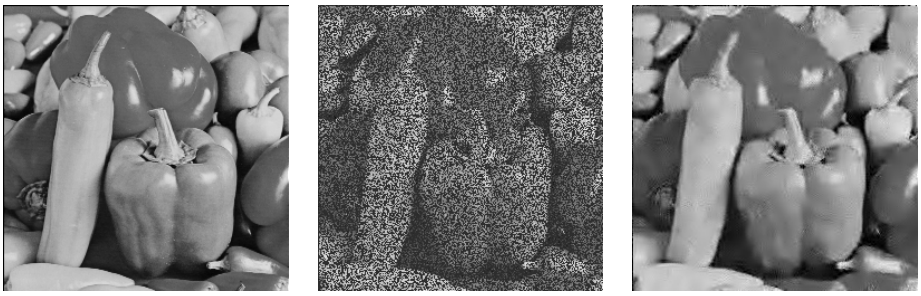

FIGURE 4. Local inpainting. Left: The original image, Middle: the measured image with 50% missing pixels, and Right: the inpainting result.

4. Dictionary learning

4.1. General dictionary learning

Our entire model and applications rely on having a suitable dictionary. A good dictionary would be one that sparsifies our data. The question we aim look at now is how to construct such dictionaries. We have already seen one possible solution – we can choose a dictionary from an existing family of transforms, such as DCT, curvelets, contourlets, wavelets and so on. This approach has its benefits, as usually these transforms can be applied rather efficiently. However, a dictionary selected this way is typically not the best fit for the actual data.

A better approach is to train a dictionary from examples. The core idea is to collect a set of representative examples of the images we work with, and train a dictionary so that the examples can be sparsely represented over it. Such a dictionary will be better suited for our data, therefore leading to better performance in various tasks.

Let us collect all the examples $\{\mathbf{x}_j\}$ into a matrix $\mathbf{X}$, where each column is one example. Then, we can write the decomposition as $\mathbf{X} \approx \mathbf{DA}$, where each column of $\mathbf{A}$ contains the coefficients for the representation of the corresponding example. A good dictionary is therefore one that is able to adequately represent each of the examples, while the representation for each example in indeed sparse, i.e., each column in $\mathbf{A}$ has only a small number of non-zeros. Formally, we would like to solve

$$\hat{\mathbf{D}}, \hat{\mathbf{A}} = \arg\min_{\mathbf{D},\mathbf{A}} \sum_{j=1}^{P} \|\mathbf{D}\alpha_j - \mathbf{x}_j\| \ s.t. \ \forall j, \ \|\alpha_j\|_0 \leq L. \tag{4.14}$$

This problem will be handled as follows: The first stage is sparse-coding, assuming $\mathbf{D}$ is fixed, and only the representation vectors are sought. In this stage we should solve for each example

$$\min_{\alpha} \|\mathbf{D}\alpha - \mathbf{x}_j\|_2^2 \ s.t. \ \|\alpha\|_0 \leq L. \tag{4.15}$$

We have seen in the first lecture several ways to perform this minimization, via pursuit algorithms. The second stage of dictionary training is the update of the dictionary, once the representation vectors $\mathbf{A}$ have been recovered. There are two main methods to do so: the Method of Optimal Directions (MOD) method [**32**] and the K-SVD method [**2**].

4.2. MOD dictionary update

One option is to try and update the dictionary at once. The goal is minimizing

$$\min_{\mathbf{D}} \|\mathbf{DA} - \mathbf{X}\|_F^2, \tag{4.16}$$

with $\mathbf{X}$ and $\mathbf{A}$ fixed. Note the use of the Frobenius norm, as both $\mathbf{DA}$ and $\mathbf{X}$ are matrices. Few simple linear algebra steps lead to

$$\mathbf{D} = \mathbf{X}\mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1} = \mathbf{X}\mathbf{A}^{\dagger}. \tag{4.17}$$

Once the dictionary is found this way, its columns need to be re-normalized.

4.3. K-SVD dictionary learning

Another option to train the dictionary is imitating the K-means algorithm in the dictionary update stage. This means that we process one atom at a time and attempt to optimally replace it.

For processing the i-th atom, the first stage is collecting only those examples that use it into the matrix $\mathbf{X}^i$. Then, the dictionary $\mathbf{D}$ and coefficient matrix $\mathbf{A}$ are all fixed apart from the i-th column in $\mathbf{D}$ and the i-th row in $\mathbf{A}$. Denoting by $\mathbf{D}_i$ and $\mathbf{A}_i$ these matrices with the i-th column/row removed respectively, the following residual is computed

$$\mathbf{E}^i = \mathbf{X}^i - \mathbf{D}_i\mathbf{A}_i = \mathbf{X}^i - \mathbf{D}\mathbf{A} + \mathbf{d}_i\alpha_i.$$

This is the residual matrix for all the examples that use the i-th atom, without taking into account the i-th atom itself. The new atom is the one that is best able to reduce the mean residual, i.e., the result of the minimization problem

$$\min_{\mathbf{d}_i,\alpha_i} \|\alpha_i\mathbf{d}_i^T - \mathbf{E}^i\|_F^2. \tag{4.18}$$

The vector α_i is the optimal set of coefficients for this atom in each example. The resulting atom $\hat{\mathbf{d}}_i$ is inserted into the dictionary, and the new vector of coefficients α_k replaces the i-th row in $\mathbf{A}$ (only in the columns relevant to the examples using this atom).

The optimization task in (4.18) is in fact a rank-1 approximation, which can be solved directly using the SVD decomposition. As every atom is updated using an SVD operation, this algorithm is appropriately named K-SVD. Note, that in the MOD training stage, the matrix of coefficients $\mathbf{A}$ is held constant, and the dictionary $\mathbf{D}$ is adapted to it. In the K-SVD algorithm, on the other hand, the atom update stage also optimizes for the k'th row of $\mathbf{A}$ at the same time.

LECTURE 5

Image processing - more practice

In the previous lecture we saw how to denoise and inpaint an image based on local, patch-processing. However, in both this cases, we used a fixed dictionary – the local DCT. In this lecture we shall revisit these two applications, and incorporate into them the K-SVD, in an attempt to improve their performance. We shall also demonstrate the capabilities of the *Sparseland* model in handling image processing tasks by considering two new applications – an image scale-up process based on a pair of trained dictionaries, and a facial image compression algorithm that performs better than JPEG-2000. More details on the applications presented here can be found in [**22, 3**].

1. Image denoising with a learned dictionary

When thinking about dictionary learning, both the MOD and the K-SVD are not able to process signals with large dimensions. The reason is obvious – a high dimensional signal implies a dictionary held as a very large explicit matrix, intolerable amounts of computations, both for the training and for using this dictionary, and a huge amount of examples to learn from, in order to avoid over-fitting. Thus, learned dictionaries are naturally used for low-dimensional signals.

We have already seen in the previous lecture that denoising (and inpainting) can be done very effectively by operating on small (e.g., 8×8) patches. The processing is then done on local blocks of images, using a dictionary that handles low-dimensional signals. Formally, the denoising we have proposed is written as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x},\{\alpha_{ij}\}_{ij}} \left[\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \mu \sum_{ij} \|\mathbf{R}_{ij}\mathbf{x} - \mathbf{D}\alpha_{ij}\|_2^2 \right] \quad s.t.\ \|\alpha_{ij}\|_0 \le L. \tag{5.1}$$

The second term is the prior, expressing our belief that every patch in the image should have a sparse representation over the dictionary $\mathbf{D}$. Our objective is to introduce a learned dictionary into the above paradigm. One option is collecting a set of high-quality images, extract patches from them, and use these examples to train a dictionary. This dictionary will be a "universal dictionary", as it is trained on general content images. This option is useful, and brings reasonably good results - about 0.5-1dB below the state of the art.

An interesting alternative is to use patches from the corrupted image itself for the training process. Suppose an image of 1000×1000 is to be denoised. Such an image contains nearly 10^6 patches, which are more than enough to allow training the dictionary. Furthermore, since the dictionary training process contains some sort of averaging, the resulting dictionary will actually be (almost) noise free, and can thus be used for denoising.

This option works much better than using the "universal dictionary", and leads to state-of-the-art results [**23**]. In fact, going back to the penalty term in (5.1), we can bring about this option by minimizing over $\mathbf{D}$ as well. Again, a coordinate descent approach is used. First, the image $\mathbf{x} = \mathbf{y}$ is fixed, and the dictionary $\mathbf{D}$ is fixed on some initialization (e.g., the redundant DCT). Each patch in the image undergoes sparse coding. Freezing the representations, the dictionary is updated. All the patches then undergo sparse coding with the new dictionary, and so. Iterating several times between sparse coding and dictionary update is exactly the training procedure we have seen in the previous lecture, performed over the patches in the image. After several such training iterations, the final dictionary is used for one last sparse-coding stage. Then, the sparse representations and the dictionary are fixed, and the result image is recovered, using the same simple aggregation method shown in the previous lecture,

$$\hat{\mathbf{x}} = \left[\mathbf{I} + \mu \sum_{ij} \mathbf{R}_{ij}^T \mathbf{R}_i j\right]^{-1} \left[\mathbf{y} + \mu \sum_{ij} \mathbf{R}_{ij}^T \mathbf{D}\hat{\alpha}_i j\right]. \tag{5.2}$$

Figure 1 presents the result obtained using this algorithm, for the same test shown in Lecture 4. The obtained image has $PSNR = 31$dB, which is 1dB higher compared to the DCT-based outcome.

FIGURE 1. Local denoising with a trained dictionary. Left: The original image, Middle: the noisy image ($\sigma = 20$), and Right: the denoising result.

This denoising algorithm has been applied to many variation of the denoising problem. One is of course the denoising of gray-scale images. When published, this method led to state-of-the-art results. Later, it was extended by Mairal et. al. by using joint sparsity (i.e., sparse coding similar patches together) [**37**]. This method is currently the best performing denoising algorithm.

The denoising algorithm could also be applied to denoising of color images. Usually, there is a sensitivity in how the relationships between the color channels should be handled. In this algorithm, the solution is simple – the training is performed on $8 \times 8 \times 3$ patches, i.e., on colored patches extracted from the image. The dictionary contains color atoms, and the rest of the algorithm remains the same. This method is also the state-of-the-art in color image denoising [**36**].

Video sequences can also be handled by this method. When processing image sequences, the redundancy between frames can and should be used in order to achieve better denoising. While most video denoising algorithms require motion

estimation (which is error-prone), the suggested method can simply train on space-time patches extracted from the video. Therefore, the dictionary is also constructed of "movielets", as each atom is a very short, very small (e.g., $8 \times 8 \times 5$) space-time patch. Furthermore, since consecutive images are very similar, it is not necessary to retrain a dictionary for each image. Instead, it can be propagated from one frame to the next, only requiring fine tuning (1 K-SVD iteration) for each one. This method's performance is equivalent to the state-of-the-art [**43**].

2. Image inpainting with dictionary learning

Turning to image inpainting, we would like to apply dictionary learning here as well, for handling the local patches. The process is very similar to the one done in denoising. The algorithm starts with dictionary training using the K-SVD, by performing alternating steps between sparse coding and dictionary update. During the sparse coding stage, only the un-masked (i.e., known) pixels in each patch are used. This is similar to the sparse coding undertaken in the fixed dictionary scenario. Once the sparse coding is done, each atom is updated using the SVD operation, just like the K-SVD method.

This dictionary training is repeated for several (e.g., 10) iterations. Then, the image is reconstructed from the final sparse coding using the same formula we have seen:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x},\{\alpha_{ij}\}_{ij}} \left[\frac{1}{2}\|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2^2 + \mu \sum_{ij} \|\mathbf{R}_{ij}\mathbf{x} - \mathbf{D}\alpha_{ij}\|_2^2 \right] \quad s.t.\ \|\alpha_{ij}\|_0 \leq L, \tag{5.3}$$

Of course, just like in the denoising case, this method can be easily generalized to inpainting of color images and inpainting of video sequences using the same extensions as in the denoising case. Figure 2 presents the results obtained with this algorithm, showing the benefit of using learned dictionary.

This approach to inpainting works well when the missing pixels are in (near-random) locations, or organized in small groups. If the missing pixels form a large area to reconstruct, the local approach will not work, and an alternative, global, approach will be needed.

Another problem could arise if the missing pixels create a periodic mask, for example, when every other pixel is missing. In this case, the dictionary will not be able to learn the relationship between neighboring pixels, because it has no information about such neighbors. Such is the case, for example, in image single-image scale-up and demosaicing. The periodic nature of the mask pattern may cause the learned dictionary to absorb this pattern into its atoms. In order to prevent that, the dictionary should trained while allowing only a very small number of non-zeros, and furthermore, only few training iterations in order to avoid over-fitting. Such a solution leads to state of the art results, which can be seen in [**36**].

3. Image scale-up with a pair of dictionaries

We turn to describe an image scale-up algorithm that uses trained dictionaries. The method described here follows the work of Yang et. al. [**54**], with several important modifications that lead to a simpler and more efficient algorithm, with better results [**22**].

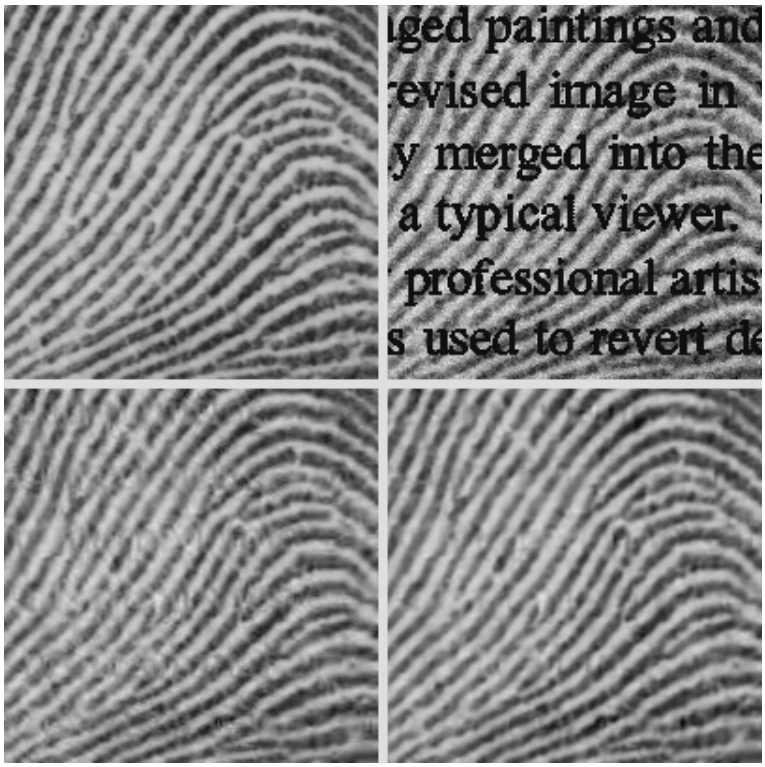

FIGURE 2. Image inpainting with the K-SVD. Top-Left: The original image, Top-Right: the measured image (masked with missing pixels), Bottom Left: DCT-based inpainting, and Bottom Right: the proposed K-SVD inpainting result.

3.1. Preliminaries

The image scale-up problem is defined as follows. A high-quality (high-resolution) image $\mathbf{y}_h$ is blurred, decimated, and contaminated by additive noise, obtaining the image $\mathbf{z}_l$,

$$\mathbf{z}_l = \mathbf{SHy}_l + \mathbf{v}, \tag{5.4}$$

with $\mathbf{H}$ a blur operation, $\mathbf{S}$ the decimation operator, and $\mathbf{v}$ the noise. We assume that the decimation $\mathbf{S}$ and blur $\mathbf{H}$ operators, and the statistical properties of the noise $\mathbf{v}$, are known.

We would like to reverse this process and recover the high quality image from the given low quality one. This is a severely ill-posed inverse problem, and therefore we shall need an image model for its regularization – we shall use the *Sparseland* model.

In order to simplify notations, we will assume the the low resolution image has undergone simple interpolation (such as bicubic) to the same size as the target image

$$\mathbf{y}_l = \mathbf{Qz}_l, \tag{5.5}$$

with $\mathbf{Q}$ being the interpolation operator. This initialization step will save us the technicalities of handling two scales in our formulation.

The algorithm we suggest relies on two core ideas. The first is operating locally, in the sense that each patch in the low-quality image will undergo resolution enhancement on its own. Then, the improved patches will be merged (using averaging in the overlap areas, just as in the previously described denoising and inpainting) in order to create the final output image.

How each patch is enhanced? This enhancement requires two dictionaries, $\mathbf{A}_l$ and $\mathbf{A}_h$ for the low and high quality patches respectively. Each atom in $\mathbf{A}_h$ should be the high-quality counterpart of the corresponding low quality atom in $\mathbf{A}_l$. In order to enhance a low-quality patch, its sparse representation over the low-quality dictionary $\mathbf{A}_l$ is found. This sparse representation is then used to reconstruct a patch from the high-quality dictionary, which is the enhancement result for the low-quality patch. We will now go into further details of this algorithm, and focus on learning these two dictionaries.

3.2. The core idea

As we have been practicing throughout this course, we assume that every $\sqrt{n}\times\sqrt{n}$ patch extracted from location k in the high-quality image through $\mathbf{p}_h^k = \mathbf{R}_k\mathbf{y_h}$, can be represented sparsely over a high-quality dictionary $\mathbf{A}_h$, i.e., for every patch $\mathbf{p}_h^k$ we assume that there exists a very sparse representation vector $\mathbf{q}^k$ such that $\mathbf{p}_h^k \cong \mathbf{A}_h\mathbf{q}^k$.

Next, we aim to find a connection between a high-quality patch in $\mathbf{y}_h$ and a patch in the same location in $\mathbf{y}_l$. We can use our knowledge of the creation process of $\mathbf{z}_l$ from $\mathbf{y}_h$ through blur, decimation and the addition of noise, with interpolation used to create $\mathbf{y}_l$. Therefore,

$$\mathbf{y}_l = \mathbf{Q}\left(\mathbf{DH}\mathbf{y}_h + \mathbf{v}\right). \tag{5.6}$$

Combining all operators into one $\mathbf{L}^{ALL} = \mathbf{QDH}$, we come to the relationship for each patch

$$\|\mathbf{L}\mathbf{p}_h^k - \mathbf{p}_l^k\|_2 \leq \epsilon, \tag{5.7}$$

with $\mathbf{L}$ a local portion of $\mathbf{L}^{ALL}$. This expression ties the low-quality patch and the high-quality patch in the same location. Using our assumption that the high quality patch can be sparsely represented over the high quality dictionary $\mathbf{p}_h^k \cong \mathbf{A}_h\mathbf{q}^k$, we obtain the relationship $\|\mathbf{L}\mathbf{A}_h\mathbf{q}^k - \mathbf{p}_l^k\|_2 \leq \epsilon$. Defining a new dictionary, $\mathbf{A}_l = \mathbf{L}\mathbf{A}_h$, we obtain

$$\|\mathbf{A}_l\mathbf{q}^k - \mathbf{p}_l^k\|_2 \leq \epsilon. \tag{5.8}$$

This observation leads to the conclusion that the sparse representation $\mathbf{q}^k$ is also the representation of the low quality patch $\mathbf{p}_l^k$ over the low quality dictionary $\mathbf{A}_l$.

3.3. Dictionary training

Since we want to use trained dictionaries, the obvious question is what do we train them on. One option is obtaining one (or more) high quality images. The high-quality image is then blurred and down-sampled it using the known (expected) operators. The low-resolution image is then interpolated back to the size of the high-quality image. Then, all the patches from each image are extracted, with patches in the same location in the original and interpolated image being paired.

Finally, each patch undergoes some pre-processing steps (different for the high-quality and low-quality patches), to form our final corpus for training. More on this pre-processing can be found in [**22**].

An alternative to using a pre-determined image for training, we can attempt training on the input image, as was done for denoising and inpainting and shown to improve results. However, we only have the low-quality image, so it seems something is missing. To our aid will come to observation that images tend to have a multi-resolution nature. If we consider an image pyramid, the characterizations of the local relationship between two consecutive levels in the pyramid is relatively the same in all levels.

From this observation comes the bootstrapping approach. The input image is blurred, decimated and then interpolated to create an even lower quality image. Now, we use the pair of low-quality and lowest-quality images as the basis for the training of the high and low quality dictionaries respectively. Once these two dictionaries are trained, the algorithm proceeds regardless of the training corpus.

Once the training patches have been collected and prepared, dictionary training can begin. The first step is training the low-quality dictionary from the set of low-quality patches. This training is done using the K-SVD algorithm. The next step is training the high-quality dictionary. We must remember that the two dictionaries should align with each other, such that a high-quality patch and its low-quality counterpart have the same representation over their dictionaries. This means that given a low quality patch $\mathbf{p}_l^k$ with a representation vector $\mathbf{q}^k$, it should also be the representation vector of the high-quality patch $\mathbf{p}_h^k$. Therefore, it makes sense to use the high-quality patches to train the high-quality dictionary such that this scale-up process gets as close as possible to its objective. Formally, we should minimize

$$\min_{\mathbf{A}_h} \sum_k \|\mathbf{p}_h^k - \mathbf{A}_l \mathbf{q}^k\|_2^2. \tag{5.9}$$

This is a simple least-squares problem to solve, and it results in a dictionary that is coupled with the low-quality dictionary that was already trained.

Once both dictionaries have been trained, we are now armed with all that we need to recover a high-resolution image given a low-resolution input. Note that this training algorithm may be done off-line on a high-quality image (which we expect to be similar to the image we would later enhance) or online, on the input image itself.

Figure 3 presents an example for the results obtained by this algorithm. The figure shows a high-resolution image and its scaled-down version. The scaling-up process aims to return to the original image – we provide two possible results, one obtained by the plain bicubic interpolation, and the second using the above-described algorithm.

4. Image compression using sparse representation

Compressing images in one of the most researched fields in image processing, because of its importance to the ability to transfer/store images easily while retaining as much of their quality as possible. Trained dictionaries that lead to the sparsest representations seem ideal for compression. However, there is a difficulty with training a global dictionary for all images, as it is expected to be less effective. The alternative, training a dictionary tailored for the specific given image, as practiced

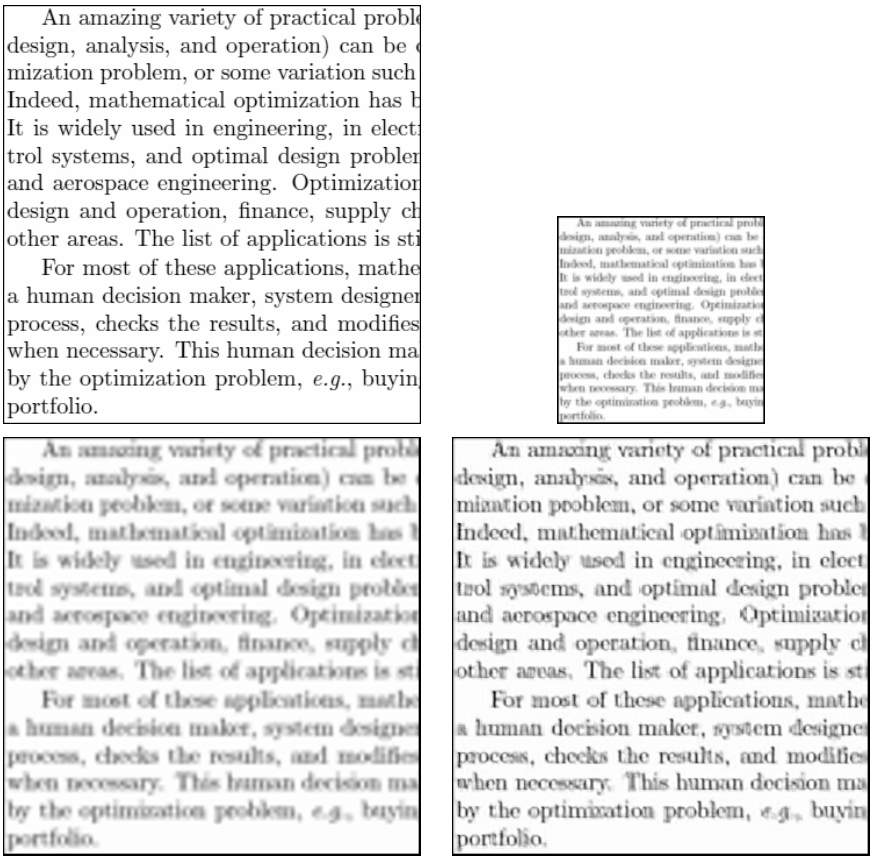

FIGURE 3. Image Scale-Up. Top-Left: The original image, Top-Right: the measured image (blurred, decimated, and noisy), Bottom Left: Bicubic scaled-up image, and Bottom Right: the proposed scale-up result.

in denoising and inpainting, is also difficult, as it requires allocated bits for the dictionary, which may compromise the success of the compression algorithm.

When turning to a specific family of images, the story changes, as it is now possible to tailor specific dictionaries to the task, while not actually needing to send them. One example for such a problem is the compression of high-quality (e.g., 500×400) facial ID images, to put on biometric identity cards or credit cards. It is possible to use general purpose compression algorithms, such as JPEG and JPEG2000 for this purpose, but a specifically trained algorithm is expected to do much better.

The proposed algorithm we describe here [**4**] has two stages – first, training appropriate dictionaries, and then, applying the trained dictionaries to compress (and decompress) the facial images.

4.1. Facial image compression algorithm - training

For training, a set of high-quality facial images are collected. This set is then used for training a dictionary for each portion of the image. The stages are as following:

(1) Key points in each image are detected (e.g., eyes, nose). These locations are used to transform each image (using a piece-wise affine transform) such that these key points are in exactly the same coordinates in all images.
(2) Each image is divided into non-overlapping 15×15 patches. Overlaps are avoided as redundancy will only worsen the compression rate.
(3) All patches in the same location are collected into a set.
(4) The mean of each set of patches is computed, stored, and removed from each patch.
(5) For each set, a dictionary is trained, either using a linear approximation (i.e., PCA) or non-linear approximation (i.e., K-SVD).
(6) Each patch location is assigned a constant number of atoms to use, and quantization levels to use, both varying from location to location. This assignment is done by minimizing the mean error on the training set while achieving the desired compression rate.

Once the training procedure ends, for each patch location, its mean patch, dictionary, number of allotted atoms and quantization levels are stored both in the encoder and decoder.

4.2. Compressing and decompressing

When a new image is to be compressed, it undergoes the following stages (similar to the training stage):

(1) The same key points in the image are detected and the image is warped.
(2) The (warped) image is divided into non-overlapping 15×15 patches.
(3) For each patch, the mean of this location is removed.
(4) Sparse coding over this location's dictionary (or projection onto the PCA) is carried out, using the number of atoms allotted to this location.
(5) The values of the coefficients selected are quantized.
(6) Entropy coding is used to further compress the sent values.

Note that if K-SVD is used, when sending the atoms' coefficients their indices are sent as well, while in PCA, only the coefficients are sent. The recovery algorithm is the exact opposite. Entropy de-coding is performed on the sent data, and each patch is then reconstructed from the send coefficients (and indices if K-SVD is used). The inverse transform is applied to the image, returning to an image that is as close as possible to the original. Figure 4 presents the obtained results for 550 Bytes per image.

5. Summary

The field of sparse and redundant representation modeling offers a new model, and this can be brought to various problems in image processing, often leading to state-of-the-art results. We have seen in these lecture-notes a partial list of these success stories, and many more have been introduced to the scientific community in the past few years. More applications, and more details in general about this field and its impact to image processing, can be found in my book:

> M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, New-York, 2010.

FIGURE 4. Facial image compression. From left to right: the original image, JPEG results, JPEG-2000 results, local-PCA, and K-SVD. The numbers represent the mean-squared-error per pixel after compression and decompression with 550 Bytes.

Bibliography

[1] Z. Ben-Haim, Y.C. Eldar, and M. Elad, Coherence-based performance guarantees for estimating a sparse vector under random noise, to appear in *IEEE Trans. on Signal Processing*.

[2] M. Aharon, M. Elad, and A.M. Bruckstein. K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*, 54(11):4311–4322, November 2006.

[3] A.M. Bruckstein, D. L. Donoho, and M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM Review*, 51(1):34-81, 2009.

[4] O. Bryt and M. Elad, Compression of facial images using the K-SVD algorithm, *Journal of Visual Communication and Image Representation*, 19(4):270–283, May 2008.

[5] E.J. Candès, J. Romberg, and T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. Inform. Theory*, 52:489-509, 2006.

[6] E.J. Candès and T. Tao, The Dantzig selector: Statistical estimation when p is much larger than n, *Annals of Statistics*, 35(6):2313–2351, June 2007.

[7] S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).

[8] I. Daubechies, M. Defrise, and C. De-Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Communications on Pure and Applied Mathematics*, LVII:1413–1457, 2004.

[9] G. Davis, S. Mallat, and M. Avellaneda, Adaptive greedy approximations, *Journal of Constructive Approximation*, 13:57–98, 1997.

[10] M.N. Do and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image on Image Processing*, 14(12):2091–2106, 2005.

[11] D.L. Donoho, De-noising by soft thresholding, *IEEE Trans. on Information Theory*, 41(3):613–627, 1995.

[12] D.L. Donoho, For most large underdetermined systems of linear equations, the minimal ℓ_1-norm solution is also the sparsest solution, *Communications On Pure And Applied Mathematics*, 59(6):797–829, June 2006.

[13] D.L. Donoho, For most large underdetermined systems of linear equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution, *Communications On Pure And Applied Mathematics*, 59(7):907–934, July 2006.

[14] D.L. Donoho and M. Elad, Optimally sparse representation in general (non-orthogonal) dictionaries via l1 minimization, *Proc. of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[15] D.L. Donoho and M. Elad, On the stability of the basis pursuit in the presence of noise, *Signal Processing*, 86(3):511–532, March 2006.

[16] D.L. Donoho, M. Elad, and V. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, *IEEE Trans. On Information Theory*, 52(1):6–18, 2006.

[17] D.L. Donoho and I.M. Johnstone, Ideal denoising in an orthonormal basis chosen from a library of bases, *Comptes Rendus del'Academie des Sciences, Series A*, 319:1317–1322, 1994.

[18] D.L. Donoho and J. Tanner, Neighborliness of randomly-projected Simplices in high dimensions, *Proceedings Of The National Academy Of Sciences*, 102(27):9452–9457, July 2005.
[19] D.L. Donoho and J. Tanner, Sparse nonnegative solutions of underdetermined linear equations by linear programming, *Proc. Natl. Acad. Sci.*, 102:9446-9451 , 2005.
[20] B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani, Least angle regression, *The Annals of Statistics*, 32(2):407–499, 2004.
[21] M. Elad, Why simple shrinkage is still relevant for redundant representations?, to appear in the *IEEE Trans. On Information Theory*.
[22] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer New-York, 2010.
[23] M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. on Image Processing* 15(12):3736–3745, December 2006.
[24] M. Elad, B. Matalon, and M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, *Applied and Computational Harmonic Analysis*, 23:346–367, November 2007.
[25] M. Elad and M. Zibulevsky, Iterative shrinkage algorithms and their acceleration for l1-l2 signal and image processing applications", *IEEE Signal Processing Magazine*, 27(3):78–88, May 2010.
[26] M.A. Figueiredo and R.D. Nowak, An EM algorithm for wavelet-based image restoration, *IEEE Trans. Image Processing*, 12(8):906–916, 2003.
[27] M.A. Figueiredo, and R.D. Nowak, A bound optimization approach to wavelet-based image deconvolution, IEEE International Conference on Image Processing - ICIP 2005, Genoa, Italy, 2:782–785, September 2005.
[28] I.F. Gorodnitsky and B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm, *IEEE Trans. On Signal Processing*, 45(3):600–616, 1997.
[29] R. Gribonval and M. Nielsen, Sparse decompositions in unions of bases, *IEEE Trans. on Information Theory*, 49(12):3320–3325, 2003.
[30] O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part I: Theory, *IEEE Trans. on Image Processing*, 15(3):539–554, 2006.
[31] O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part II: Adaptive algorithms, *IEEE Trans. on Image Processing*, 15(3):555–571, 2006.
[32] K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T-W, Lee, and T.J. Sejnowski, Dictionary learning algorithms for sparse representation, *Neural Computation*, 15(2)349–396, 2003.
[33] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, A method for large-scale ,1-regularized least squares problems with applications in signal processing and statistics, *IEEE J. Selected Topics Signal Processing*, 1(4):606-617, Dec. 2007.
[34] J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, *Linear Algebra and its Applications*, 18(2):95–138, 1977.
[35] X. Liu and N.D. Sidiropoulos, Cramer-Rao lower bounds for low-rank decomposition of multidimensional arrays, *IEEE Trans. on Signal Processing*, 49(9):2074–2086, 2001.
[36] J. Mairal, M. Elad, and G. Sapiro, Sparse representation for color image restoration, *IEEE Trans. on Image Processing*, 17(1):53–69, January 2008.
[37] J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman, Non-local sparse models for image restoration, International Conference on Computer Vision (ICCV), Tokyo, Japan, 2009.
[38] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic-Press, Second-Edition, 2009.
[39] S. Mallat and E. LePennec, Sparse geometric image representation with bandelets, *IEEE Trans. on Image Processing*, 14(4):423–438, 2005.

[40] S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.

[41] B.K. Natarajan, Sparse approximate solutions to linear systems, *SIAM Journal on Computing*, 24:227–234, 1995.

[42] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, the twenty seventh Asilomar Conference on Signals, Systems and Computers, 1:40–44, 1993.

[43] M. Protter and Michael Elad, Image sequence denoising via sparse and redundant representations, *IEEE Trans. on Image Processing*, 18(1):27–36, January 2009.

[44] L. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D*, 60:259–268, 1992

[45] E.P. Simoncelli, W.T. Freeman, E.H. Adelsom, and D.J. Heeger, Shiftable multiscale transforms, *IEEE Trans. on Information Theory*, 38(2):587–607, 1992.

[46] J.-L. Starck, E.J. Candès, and D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. on Image Processing*, 11:670–684, 2002.

[47] J.-L. Starck, M. Elad, and D.L. Donoho, Redundant multiscale transforms and their application for morphological component separation, *Advances in Imaging And Electron Physics*, 132:287–348, 2004.

[48] J.-L. Starck, M. Elad, and D.L. Donoho, Image decomposition via the combination of sparse representations and a variational approach. *IEEE Trans. On Image Processing*, 14(10):1570–1582, 2005.

[49] T. Strohmer and R. W. Heath, Grassmannian frames with applications to coding and communication, *Applied and Computational Harmonic Analysis*, 14:257–275, 2004.

[50] J.A. Tropp, Greed is good: Algorithmic results for sparse approximation, *IEEE Trans. On Information Theory*, 50(10):2231–2242, October 2004.

[51] J.A. Tropp, Just relax: Convex programming methods for subset selection and sparse approximation, *IEEE Trans. On Information Theory*, 52(3):1030–1051, March 2006.

[52] J.A. Tropp, I.S. Dhillon, R.W. Heath Jr., and T. Strohmer, Designing structured tight frames via alternating projection, *IEEE Trans. Information Theory*, 51(1):188–209, January 2005.

[53] J.A. Tropp and A.A. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, Submitted for publication, April 2005.

[54] J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution via sparse representation, to appear in P*IEEE Trans. on Image Processing*.