Dictionary learning from phaseless measurements
Andreas Tillmann, Yonina Eldar, Julien Mairal

To cite this version:
Andreas Tillmann, Yonina Eldar, Julien Mairal. Dictionary learning from phaseless measurements. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Mar 2016, Shanghai, China. pp.4702-4706, 10.1109/ICASSP.2016.7472569. hal-01387416

HAL Id: hal-01387416
https://inria.hal.science/hal-01387416
Submitted on 25 Oct 2016
DICTIONARY LEARNING FROM PHASELESS MEASUREMENTS

Andreas M. Tillmann*  Yonina C. Eldar†  Julien Mairal○

* TU Darmstadt, Research Group Optimization, Dolivostr. 15, 64293 Darmstadt, Germany
† Department of Electrical Engineering, Technion, Haifa 32000, Israel
○ Inria, Lear Team, 655 Avenue de l’Europe, 38330 Montbonnot, France

ABSTRACT

We propose a new algorithm to learn a dictionary along with sparse representations from signal measurements without phase. Specifically, we consider the task of reconstructing a two-dimensional image from squared-magnitude measurements of a complex-valued linear transformation of the original image. Several recent phase retrieval algorithms exploit underlying sparsity of the unknown signal in order to improve recovery performance. In this work, we consider sparse phase retrieval when the sparsifying dictionary is not known in advance, and we learn a dictionary such that each patch of the reconstructed image can be sparsely represented. Our numerical experiments demonstrate that our proposed scheme can obtain significantly better reconstructions for noisy phase retrieval problems than methods that cannot exploit such “hidden” sparsity.

Index Terms— Phase Retrieval, Dictionary Learning

1. INTRODUCTION

Phase retrieval has been an active research topic for decades [1, 2]. In mathematical terms, it can be formulated as

\[ \text{find } x \in X \subseteq \mathbb{C}^N \text{ s.t. } |f_i(x)|^2 = y_i \forall i = 1, \ldots, M, \]  

where the functions \( f_i : \mathbb{C}^N \rightarrow \mathbb{C} \) are linear operators and the scalars \( y_i = |f_i(\hat{x})|^2 \) are nonlinear measurements of the original signal \( \hat{x} \) in \( X \), obtained by removing the phase information. A traditional approach is to consider cases in which the solution of (1) is unique, up to global phase, and to devise algorithms to recover \( \hat{x} \). Usually, recovering \( \hat{x} \) involves overampling, i.e., taking \( M > N \) measurements. The most popular techniques for phase retrieval are based on alternating projections [3, 4, 5]. These methods generally require precise prior information on the signal (such as knowledge of the support set) and often converge to erroneous results. More recent approaches include semidefinite programming [6, 7, 8, 9, 10] and gradient-based methods such as Wirtinger Flow [11, 12].

In recent years, new techniques for (1) were developed when \( \hat{x} \) is sparse, i.e., is composed of only few atoms from a known dictionary [7, 13, 12, 14], leading to algorithms with improved recovery performance. The main idea is akin to compressed sensing, where one works with fewer linear measurements than signal components [15, 16]. An important observation that boosted the applicability of sparse recovery techniques is that many classes of signals admit a sparse approximation in some basis or overcomplete dictionary [17, 18, 19]. While sometimes such dictionaries are known explicitly, better results can often be achieved by adapting the dictionary to the data [18]. Numerous algorithms have been developed for this task (e.g., [20, 21]) when the signal measurements are linear.

In this work, we propose a dictionary learning scheme for simultaneously solving the signal reconstruction and sparse representation problems given nonlinear, phaseless and possibly noisy measurements. We demonstrate its ability to achieve significantly improved reconstructions (especially when the oversampling ratio is low and the noise level high) compared to Wirtinger Flow (WF) [11], which cannot exploit sparsity if the dictionary is unknown. Our algorithm, referred to as phase-retrieval dictionary-learning (PRDL), is based on alternating minimization where we iterate between best fitting the data and sparsely representing the recovered image. PRDL combines projected gradient descent like in Wirtinger Flow to update the image, iterative shrinkage to sparsify the image [22] and block-coordinate descent for the dictionary update [21].

2. PHASE-RETRIEVAL DICTIONARY-LEARNING

Our focus is on the application to image reconstruction. Therefore, we work in the 2D setting directly; however, all expressions and algorithms could easily be formulated for 1D signals, as in (1).

We wish to recover an image \( \hat{X} \) in \([0, 1]^{N_1 \times N_2} \) from noise-corrupted phaseless nonlinear measurements

\[ Y := |F(\hat{X})|^2 + N, \]  

where \( F : \mathbb{C}^{N_1 \times N_2} \rightarrow \mathbb{C}^{M_1 \times M_2} \) is a linear operator, \( N \) denotes noise, and the complex modulus and squares are taken component-wise. Signal sparsity is known to aid in phase retrieval, but a sparsifying transform is not always known a priori. This motivates learning a dictionary \( D \) in \( \mathbb{R}^{s \times n} \) such that each \( s_1 \times s_2 \) patch \( \hat{x}' \) of \( \hat{X} \), represented as a vector of size \( s = s_1s_2 \), can be approximated by \( \hat{x}' \approx Da' \) with a sparse vector \( a' \) in \( \mathbb{R}^n \). Here, \( n \) is chosen a priori and the number of patches depends on whether the patches are overlapping or not. In general, \( D \) is chosen such that \( n \geq s \).

Before presenting our approach for tackling (2), we define the following notation. We consider the linear operator \( \hat{F} : \mathbb{C}^{N_1 \times N_2} \rightarrow \mathbb{C}^{s \times p} \) that extracts the \( p \) patches \( x' \) (which may overlap or not) from an image \( X \) and forms the matrix \( \hat{F}(X) = (x_1', \ldots, x_p') \). Similarly, we define the linear operator \( \hat{R} : \mathbb{C}^{s \times p} \rightarrow \mathbb{C}^{N_1 \times N_2} \) that reverses this process, i.e., builds an image from a matrix containing vectorized patches as its columns. Thus, in particular, we have \( \hat{R}(\hat{F}(X)) = X \). Further, let \( A := (a_1', \ldots, a_p') \) in \( \mathbb{R}^{s \times p} \) be the matrix containing the patch representation coefficient vectors as columns. Then, our desired sparse-approximation relation \( x' \approx Da' \) can be expressed as \( \hat{F}(X) \approx DA \).

With this notation in hand, we now introduce our method, called phase-retrieval dictionary-learning (PRDL), which aims at solving

\[
\min_{X \in \mathcal{D} \times D} \frac{1}{N} \|Y - |F(X)|^2\|^2_2 + \frac{1}{2} \|\hat{F}(X) - DA\|^2_F + \lambda \sum_{i=1}^p \|a_i\|_1
\]

s.t. \( X \in [0, 1]^{N_1 \times N_2} \), \( D \in \mathcal{D} \). (3)
Learned dictionary calls for decreasing the objective\(A\) counter \(\ell\) thresholding operator product of two matrices. The algorithm also involves the classical soft-
the conjugate transpose),\(\ast\) denotes the Hadamard (element-wise) product of two matrices. The algorithm also involves the classical soft-

\[\ell\] sparsity of the image patches. To that effect, we use the objective (3) provide a way to control the trade-off between the data
the Euclidean norm to matrices. The parameters \(\mu, \lambda > 0\) in the objective (3) provide a way to control the trade-off between the data

\[\|X\|_F\] denotes the Frobenius matrix-norm, which generalizes the Euclidean norm to matrices. The parameters \(\mu, \lambda > 0\)

\[\ell\] convergence to a stationary point of (3) can be obtained under mild assumptions, using results from [25]; we cannot give details here due to space restrictions.)

The algorithmic framework we employ is that of alternating minimization: For each variable \(D, A\) and \(X\) in turn, we take one step towards solving (3) w.r.t. this variable alone, keeping the other two fixed. We summarize our method in Algorithm 1, where the superscript \(\ast\) denotes the adjoint operator (for a matrix \(Z\), \(Z^\ast\) is thus the conjugate transpose). \(\Re(\cdot)\) extracts the real part of a complex-valued argument, and \(\odot\) denotes the Hadamard (element-wise) product of two matrices. The algorithm also involves the classical soft-thresholding operator \(S_{\lambda/\mu}(Z) := \max\{0, |Z| - \tau\} \odot \text{sign}(Z)\)

and the Euclidean projection \(P_F(Z) := \max\{0, \min\{1, Z\}\}\) onto \(X := [0, 1]^{N_1 \times N_2}\); here, all operations are meant component-wise.

To avoid training the dictionary on potentially useless early estimates, the algorithm performs two phases—while the iteration counter \(\ell\) is smaller than \(K_1\), the dictionary is not updated. We next explain the algorithmic steps in more detail.

### 3.1. Updating the Patch Representation Vectors

Updating \(A\) (i.e., (3) with \(D\) and \(X\) fixed at their current values) calls for decreasing the objective

\[\sum_{i=1}^{p} \left( \frac{1}{2} \| D^{(i)} a_i^\ast - x_i^\ast \|^2_F + \frac{1}{\mu} \| a_i^\ast \|_1 \right).\] (4)

Since the objective here is separable, we can update all vectors \(a_i^\ast\) in parallel independently. To do so, we choose to perform one step of the algorithm ISTA (see, e.g., [22]), which has a monotonic decrease property. ISTA is a gradient-based method which performs the following update for each \(i = 1, \ldots, p;\)

\[a_i^{(\ell+1)} = S_{\lambda/\mu} \left( a_i^{(\ell)} - \frac{\mu}{\lambda} \nabla \phi(x_i) \right).\]

This update involves a gradient descent step followed by soft-thresholding. The step size parameter \(\gamma_\ell^A\) can be chosen as \(1/L_A\), where \(L_A\) is an upper bound on the Lipschitz constant of the gradient; here, \(L_A = \|D_{ij}\|_2^2\) would do, but a better strategy is to use a backtracking scheme to automatically update \(L_A\) [22].

Constructing \(A_{(\ell+1)}\) from the \(a_i^{(\ell+1)}\) as specified above is equivalent to Step 2 of Algorithm 1.

### 3.2. Updating the Image Estimate

With \(D = D^{(\ell)}\) and \(A = A^{(\ell+1)}\) fixed, updating \(X\) consists of decreasing the objective

\[\frac{1}{2} \| Y - |F(X)|^2 \|^2_F + \frac{1}{2} \| E(X) - DA \|^2_F\] (5)

This problem can be seen as a regularized version of the phase retrieval problem (with regularization parameter \(\mu\)) that encourages the patches of \(X\) to be close to the sparse approximation \(DA\) obtained during the previous iterate.

Our approach to decrease the value of the objective (5) is by a simple projected gradient descent step. In fact, for \(\mu = 0\), this reduces to the so-called Wirtinger Flow method [11], but with necessary modifications to take into account the constraints on \(X\) (real-valuedness and variable bounds [0, 1]).

The gradient (matrix) of \(\phi(X)\) := \(1/2\|Y - |F(X)|^2\|^2_F\) with respect to \(X\) can be computed as

\[\nabla \phi(X) = \Re \left( F^* (F(X) \odot (|F(X)|^2 - Y)) \right),\]

by using the chain rule (we omit a more detailed derivation due to space limitations). The gradient of \(\psi(X) := \frac{1}{2} \|E(X) - DA\|_2^2\) is given by

\[\nabla \psi(X) = \mu E^* (E(X) - DA) = \mu R \odot R (E(X) - DA),\]
where \( R \in \mathcal{X} \) has entries \( r_{ij} \) equal to the number of patches the respective pixel \( x_{ij} \) is contained in. Note that if the whole image is divided into a complete set of non-overlapping patches, \( R \) will just be the all-ones matrix; otherwise, the element-wise multiplication with \( R \) undoes the averaging of pixel values performed by \( R \) when assembling an image from overlapping patches.

Finally, the gradient w.r.t. \( X \) of the objective in (5) is \( \nabla \varphi(X) + \nabla \psi(X) \in \mathbb{R}^{N_2 \times N_2} \), and the update in Step 3 of Algorithm 1 is indeed shown to be a projected gradient descent step. Typically, a backtracking strategy may be used for choosing the step size \( \gamma^t \), as detailed in the section devoted to numerical experiments.

3.3. Updating the Dictionary

To update the dictionary, i.e., to approximately solve (3) w.r.t. \( D \) alone, keeping \( X \) and \( A \) fixed at their current values, we employ one pass of a block-coordinate descent (BCD) algorithm on the columns of the dictionary [21]. The objective to decrease may be written as

\[
\sum_{i=1}^{p} \left\| DA^t_{i(t+1)} - x^t_{i(t+1)} \right\|^2_2 \quad \text{s.t.} \quad D \in \mathcal{D},
\]

(6)

and the update rule given by Steps 4–13 corresponds exactly to one iteration of [19, Algorithm 11] applied to (6), which ensures the monotonic decrease of the objective function.

4. NUMERICAL EXPERIMENTS

We first describe some aspects of the concrete models and our implementation and then proceed to discuss our experiments.

4.1. Experimental Setup

We consider several linear operators \( F \) corresponding to different types of measurements. We denote by \( F \) the (normalized) 2D-Fourier transform, and introduce two complex Gaussian matrices \( G \in \mathbb{C}^{M_1 \times N_1} \), \( H \in \mathbb{C}^{M_2 \times N_2} \), whose entries are i.i.d. samples from the distribution \( N(0, I/2) \) and \( iN(0, I/2) \). Then, we experiment with the operators \( F(X) = GX \), \( F(X) = GXG^* \), \( \mathcal{F}(X) = GXH^* \), and the coded diffraction model

\[
\mathcal{F}(X) = \begin{pmatrix} F(M_1 \odot X) \\ \vdots \\ F(M_n \odot X) \end{pmatrix}, \quad \mathcal{F}^t(Z) = \sum_{j=1}^{m} (M_j \odot F^*(Z)),
\]

(7)

where the \( M_j \)'s are admissible coded diffraction patterns (CDPs), see for instance [11, Sect. 4.1]; in our experiments we used ternary CDPs, such that each \( M_j \) is in \( \{0, \pm 1\}^{N_1 \times N_2} \). Note that the operator \( F \) is implemented using fast Fourier transforms.

To reconstruct \( X \), we choose an oversampling setting where \( M_1 = 4N_1, M_2 = 4N_2 \) and/or \( m = 2 \), respectively. Moreover, we corrupt our measurements with additive white Gaussian noise \( N \) such that \( \text{SNR}(Y, |F(X)|^2 + N) = 10 \text{dB} \) for the Gaussian-type, and 20 dB for CDP measurements, respectively. Note that this settings yields a quite heavy noise level for the Gaussian cases, and a relatively low oversampling ratio for the CDPs.

4.2. Implementation Details

We choose to initialize our algorithm with a simple random image \( X_0 \) in \( \mathcal{X} \) to demonstrate the robustness of our approach with respect to its initialization. Nevertheless, other choices are possible. For instance, one may also initialize \( X_0 \) with a power-method scheme similar to that proposed in [11], modified to account for the required real-valuedness and box-constraints. The dictionary is initialized as \( D_0 = \{I, F_0\} \in \mathbb{R}^{N_2 \times 2N_2} \), where \( F_0 \) is the 2D discrete cosine transform (see, e.g., [18]).

To update \( A \), we use the ISTA implementation from the SPAMS package [21] with its integrated backtracking line search (for \( L_\text{A} \)).

Regarding the step sizes \( \gamma^t \) for the update of \( X \) (Step 3 of Alg. 1), we adopt the following simple strategy: Whenever the gradient step leads to a reduction in the objective function value, we accept it; otherwise, we recompute the step with \( \gamma^t \) halved until a reduction was achieved (no more than 100 trials). Regardless of whether \( \gamma^t \) was reduced or not, we reset its value to \( 1.68\gamma^t \) for the next round; the initial step size is 1.

Finally, we consider non-overlapping \( 8 \times 8 \) patches (though our code can also handle overlap). We run PRDL (Algorithm 1) with \( k_1 = 25 \) and \( k_2 = 50 \); the regularization/penalty parameter values can be read from Table 1 (there, \( m_{ij} \) is the number of elements of \( Y_j \)). We remark that these parameter values were empirically found to work well for the instances considered here but are not the outcome of rigorous parameter benchmarking, which is a subject of future investigations.

4.3. Computational Experiments

We test our method on a collection of typical (grayscale) test images used in the literature, namely cameraman, house and peppers of size \( 256 \times 256 \), and lena, barbara, boat, fingerprint and mandrill of size \( 512 \times 512 \). All experiments were carried out on a Linux 64-bit quad-core machine (2.8 GHz, 8 GB RAM) running Matlab R2015a.

We evaluate our approach with the following question in mind: Can we improve upon the quality of reconstruction compared to standard phase retrieval algorithms? Standard methods cannot exploit sparsity if the underlying basis or dictionary is unknown; as we will see, the introduced (patch-)sparsity indeed allows for better recovery results (at least in the oversampling and noise regimes considered here).

To evaluate the achievable sparsity, we look at the average number of nonzeros in the columns of \( A \) after running our algorithm. Generally, smaller values indicate an improved suitability of the learned dictionary for sparse patch coding (high values often occur if the regularization parameter \( \lambda \) is too small and the dictionary is learning the noise, which is something we would like to avoid).

To assess the quality of the image reconstruction, we consider two standard measures, namely the peak signal-to-noise ratio (PSNR) of the reconstructed \( X \) as well as its structural similarity index (SSIM) [26]. For PSNR, larger values are better, and SSIM-values closer to 1 (always ranging between 0 and 1) indicate better visual quality.

Table 1 displays the CPU times, PSNR- and SSIM-values and mean patch representation vector sparsity levels obtained for the various measurement types, averaged over the instance groups of the same size. The concrete example in Figure 1 shows the results from PRDL and plain Wirtinger Flow (WF; the real-valued, [0, 1]-box constrained variant, which corresponds to running Algorithm 1 with \( \mu = 0 \) and omitting the updates of \( A \) and \( D \)) for CDP measurements of the mandrill image. (In all tests, we let the Wirtinger Flow method run for the same number of iterations (75) and use the same starting points as for the PRDL runs.)

The PRDL method consistently provides better image reconstructions than WF, which clearly shows that our approach successfully introduces sparsity into the phase retrieval problem and exploits

\[ \text{http://spams-devel.gforge.inria.fr/} \]
Table 1. Test results for \( m_Y \) Gaussian-type and coded diffraction pattern (CDP) measurements. Reported are mean values (geometric mean for CPU times) per measurement type over the three 256 × 256 and five 512 × 512 instances w.r.t. the reconstructions from PRDL (\( X_{\text{PRDL}} \) and \( R(DA) \)) with parameters \((\mu, \lambda)\) and (real-valued, \([0, 1]\)-constrained) Wirtinger Flow (\( X_{\text{WF}} \)), resp. (CPU times in seconds, PSNR in dB).

| \( F \) type | reconstruction | \( (\mu, \lambda)/m_Y \) | time | PSNR | SSIM | avg. \( ||a^i||_0 \) | \( (\mu, \lambda)/m_Y \) | time | PSNR | SSIM | avg. \( ||a^i||_0 \) |
|-------------|----------------|-------------------|------|------|------|----------------|-------------------|------|------|------|----------------|
| \( G\hat{X} \) | \( X_{\text{PRDL}} \) \( R(DA) \) \( X_{\text{WF}} \) | (0.5,0.068) | 10.85 | 24.75 | 0.5370 | 7.02 | (0.5,0.068) | 52.93 | 21.18 | 0.5592 | 2.12 |
| \( G\hat{X}G^* \) | \( X_{\text{PRDL}} \) \( R(DA) \) \( X_{\text{WF}} \) | (0.5,0.226) | 42.55 | 22.42 | 0.4042 | 6.72 | (0.5,0.068) | 238.91 | 22.54 | 0.5275 | 12.01 |
| \( G\hat{X}H^* \) | \( X_{\text{PRDL}} \) \( R(DA) \) \( X_{\text{WF}} \) | (0.5,0.226) | 41.92 | 22.62 | 0.4114 | 6.51 | (0.5,0.068) | 242.71 | 22.53 | 0.5269 | 11.99 |
| CDP (cf. (7)) | \( X_{\text{PRDL}} \) \( R(DA) \) \( X_{\text{WF}} \) | (0.05,0.003) | 7.51 | 27.20 | 0.7434 | 7.79 | (0.05,0.003) | 31.17 | 27.16 | 0.7764 | 11.85 |

**Fig. 1.** PRDL example: Image original is the 512 × 512 “mandrill” picture, measurements are noisy CDPs (obtained using two ternary masks). (a) final dictionary (excerpt), (b) image reconstruction \( R(DA) \) from sparsely coded patches, (c) reconstruction \( X_{\text{WF}} \) after 75 WF iterations. Final PSNR 25.39 dB (\( X_{\text{PRDL}} \): 26.45, \( X_{\text{WF}} \): 13.42) and SSIM 0.7212 (\( X_{\text{PRDL}} \): 0.7601, \( X_{\text{WF}} \): 0.1562), average \( ||a^i||_0 \) is 11.95.
6. REFERENCES


