ONLINE DICTIONARY LEARNING FOR SPARSE CODING

By

Julien Mairal

Francis Bach

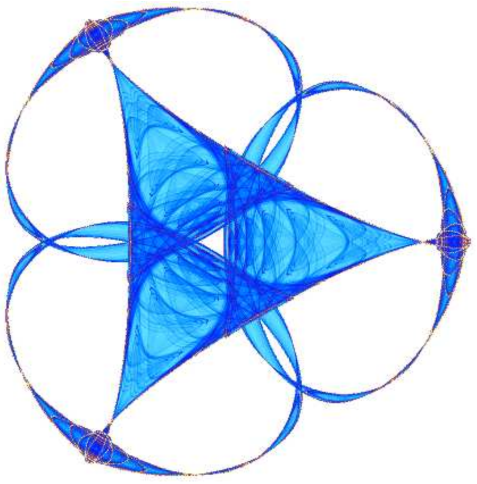
Jean Ponce

and

Guillermo Sapiro

IMA Preprint Series # 2249

(April 2009)



Online Dictionary Learning for Sparse Coding

Julien Mairal JULIEN.MAIRAL@INRIA.FR
Francis Bach FRANCIS.BACH@INRIA.FR
INRIA, Paris, France[1]

Jean Ponce JEAN.PONCE@ENS.FR
Ecole Normale Supérieure, Paris, France[1]

Guillermo Sapiro GUILLE@UMN.EDU
University of Minnesota - Department of Electrical and Computer Engineering, Minneapolis, USA

Abstract

Sparse coding—that is, modelling data vectors as sparse linear combinations of basis elements—is widely used in machine learning, neuroscience, signal processing, and statistics. This paper focuses on *learning* the basis set, also called dictionary, to adapt it to specific data, an approach that has recently proven to be very effective for signal reconstruction and classification in the audio and image processing domains. This paper proposes a new online optimization algorithm for dictionary learning, based on stochastic approximations, which scales up gracefully to large datasets with millions of training samples. A proof of convergence is presented, along with experiments with natural images demonstrating that it leads to faster performance and better dictionaries than classical batch algorithms for both small and large datasets.

1. Introduction

The linear decomposition of a signal using a few atoms of a *learned* dictionary instead of a predefined one—based on wavelets (Mallat, 1999) for example—has recently led to state-of-the-art results for numerous low-level image processing tasks such as denoising (Elad & Aharon, 2006) as well as higher-level tasks such as classification (Raina et al., 2007; Mairal et al., 2008a), showing that sparse learned models are well adapted to natural signals. Unlike decompositions based on principal component analysis and its variants, these models do not impose that the basis vectors be orthogonal, allowing more flexibility to adapt the representation to the data. While learning the dictionary has proven to be critical to achieve (or improve upon) state-of-the-art results, effectively solving the corresponding optimization problem is a significant computational challenge, particularly in the context of the large-scale datasets involved in image processing tasks, that may include millions of training samples. Addressing this challenge is the topic of this paper.

Concretely, consider a signal $\mathbf{x}$ in $\mathbb{R}^m$. We say that it admits a sparse approximation over a *dictionary* $\mathbf{D}$ in $\mathbb{R}^{m \times k}$, with k columns referred to as *atoms*, when one can find a linear combination of a "few" atoms from $\mathbf{D}$ that is "close" to the signal $\mathbf{x}$. Experiments have shown that modelling a signal with such a sparse decomposition (*sparse coding*) is very effective in many signal processing applications (Chen et al., 1999). For natural images, predefined dictionaries based on various types of wavelets (Mallat, 1999) have been used for this task. However, learning the dictionary instead of using off-the-shelf bases has been shown to dramatically improve signal reconstruction (Elad & Aharon, 2006). Although some of the learned dictionary elements may sometimes "look like" wavelets (or Gabor filters), they are tuned to the input images or signals, leading to much better results in practice.

Most recent algorithms for dictionary learning (Engan et al., 1999; Aharon et al., 2006; Lee et al., 2007) are second-order iterative *batch* procedures, accessing the whole training set at each iteration in order to minimize a cost function under some constraints. Although they have shown experimentally to be much faster than first-order gradient descent methods (Lee et al., 2007), they cannot effectively handle very large training sets (Bottou & Bousquet, 2007), or dynamic training data changing over time,

[1]WILLOW Project, Laboratoire d'Informatique de l'Ecole Normale Supérieure, ENS/INRIA/CNRS UMR 8548.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

such as video sequences. To address these issues, we propose an *online* approach that processes one element (or a small subset) of the training set at a time. This is particularly important in the context of image and video processing (Protter & Elad, 2009), where it is common to learn dictionaries adapted to small patches, with training data that may include several millions of these patches (roughly one per pixel and per frame). In this setting, online techniques based on stochastic approximations are an attractive alternative to batch methods (Bottou, 1998). For example, first-order stochastic gradient descent with projections on the constraint set (Kushner & Yin, 2003) is sometimes used for dictionary learning (see Aharon and Elad (2008) for instance). We show in this paper that it is possible to go further and exploit the specific structure of sparse coding in the design of an optimization procedure dedicated to the problem of dictionary learning, with low memory consumption and lower computational cost than classical second-order batch algorithms and without the need of explicit learning rate tuning. As demonstrated by our experiments, the algorithm scales up gracefully to large datasets with millions of training samples, and it is usually faster than more standard methods.

1.1. Contributions

This paper makes three main contributions.

• We cast in Section 2 the dictionary learning problem as the optimization of a smooth nonconvex objective function over a convex set, minimizing the (desired) *expected* cost when the training set size goes to infinity.

• We propose in Section 3 an iterative online algorithm that solves this problem by efficiently minimizing at each step a quadratic surrogate function of the empirical cost over the set of constraints. This method is shown in Section 4 to converge with probability one to a stationary point of the cost function.

• As shown experimentally in Section 5, our algorithm is significantly faster than previous approaches to dictionary learning on both small and large datasets of natural images. To demonstrate that it is adapted to difficult, large-scale image-processing tasks, we learn a dictionary on a 12-Megapixel photograph and use it for inpainting.

2. Problem Statement

Classical dictionary learning techniques (Olshausen & Field, 1997; Engan et al., 1999; Aharon et al., 2006; Lee et al., 2007) consider a finite training set of signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ in $\mathbb{R}^{m \times n}$ and optimize the empirical cost function

$$f_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{x}_i, \mathbf{D}), \tag{1}$$

where $\mathbf{D}$ in $\mathbb{R}^{m \times k}$ is the dictionary, each column representing a basis vector, and l is a loss function such that $l(\mathbf{x}, \mathbf{D})$ should be small if $\mathbf{D}$ is "good" at representing the signal $\mathbf{x}$. The number of samples n is usually large, whereas the signal dimension m is relatively small, for example, $m = 100$ for 10×10 image patches, and $n \geq 100,000$ for typical image processing applications. In general, we also have $k \ll n$ (e.g., $k = 200$ for $n = 100,000$), and each signal only uses a few elements of $\mathbf{D}$ in its representation. Note that, in this setting, overcomplete dictionaries with $k > m$ are allowed. As others (see (Lee et al., 2007) for example), we define $l(\mathbf{x}, \mathbf{D})$ as the optimal value of the ℓ_1-*sparse coding* problem:

$$l(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^k} \frac{1}{2} ||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda ||\boldsymbol{\alpha}||_1, \tag{2}$$

where λ is a regularization parameter.[2] This problem is also known as *basis pursuit* (Chen et al., 1999), or the *Lasso* (Tibshirani, 1996). It is well known that the ℓ_1 penalty yields a sparse solution for $\boldsymbol{\alpha}$, but there is no analytic link between the value of λ and the corresponding effective sparsity $||\boldsymbol{\alpha}||_0$. To prevent $\mathbf{D}$ from being arbitrarily large (which would lead to arbitrarily small values of $\boldsymbol{\alpha}$), it is common to constrain its columns $(\mathbf{d}_j)_{j=1}^k$ to have an ℓ_2 norm less than or equal to one. We will call $\mathcal{C}$ the convex set of matrices verifying this constraint:

$$\mathcal{C} \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times k} \ \text{ s.t. } \forall j = 1, \ldots, k, \ \mathbf{d}_j^T \mathbf{d}_j \leq 1\}. \tag{3}$$

Note that the problem of minimizing the empirical cost $f_n(\mathbf{D})$ is not convex with respect to $\mathbf{D}$. It can be rewritten as a joint optimization problem with respect to the dictionary $\mathbf{D}$ and the coefficients $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n]$ of the sparse decomposition, which is not jointly convex, but convex with respect to each of the two variables $\mathbf{D}$ and $\boldsymbol{\alpha}$ when the other one is fixed:

$$\min_{\mathbf{D} \in \mathcal{C}, \boldsymbol{\alpha} \in \mathbb{R}^{k \times n}} \frac{1}{n} \sum_{i=1}^{n} \Big(\frac{1}{2} ||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda ||\boldsymbol{\alpha}_i||_1 \Big). \tag{4}$$

A natural approach to solving this problem is to alternate between the two variables, minimizing over one while keeping the other one fixed, as proposed by Lee et al. (2007) (see also Engan et al. (1999) and Aharon et al. (2006), who use ℓ_0 rather than ℓ_1 penalties, for related approaches).[3] Since the computation of $\boldsymbol{\alpha}$ dominates the cost of each iteration, a second-order optimization technique

[2]The ℓ_p norm of a vector $\mathbf{x}$ in $\mathbb{R}^m$ is defined, for $p \geq 1$, by $||\mathbf{x}||_p \triangleq (\sum_{i=1}^m |\mathbf{x}[i]|^p)^{1/p}$. Following tradition, we denote by $||\mathbf{x}||_0$ the number of nonzero elements of the vector $\mathbf{x}$. This "ℓ_0" sparsity measure is not a true norm.

[3]In our setting, as in (Lee et al., 2007), we use the convex ℓ_1 norm, that has empirically proven to be better behaved in general than the ℓ_0 pseudo-norm for dictionary learning.

can be used in this case to accurately estimate $\mathbf{D}$ at each step when $\boldsymbol{\alpha}$ is fixed.

As pointed out by Bottou and Bousquet (2007), however, one is usually not interested in a perfect minimization of the *empirical cost* $f_n(\mathbf{D})$, but in the minimization of the *expected cost*

$$f(\mathbf{D}) \stackrel{\Delta}{=} \mathbb{E}_{\mathbf{x}}[l(\mathbf{x}, \mathbf{D})] = \lim_{n\to\infty} f_n(\mathbf{D}) \text{ a.s.}, \tag{5}$$

where the expectation (which is assumed finite) is taken relative to the (unknown) probability distribution $p(\mathbf{x})$ of the data.[4] In particular, given a finite training set, one should not spend too much effort on accurately minimizing the empirical cost, since it is only an approximation of the expected cost.

Bottou and Bousquet (2007) have further shown both theoretically and experimentally that stochastic gradient algorithms, whose rate of convergence is not good in conventional optimization terms, may in fact in certain settings be the fastest in reaching a solution with low expected cost. With large training sets, classical batch optimization techniques may indeed become impractical in terms of speed or memory requirements.

In the case of dictionary learning, classical projected first-order stochastic gradient descent (as used by Aharon and Elad (2008) for instance) consists of a sequence of updates of $\mathbf{D}$:

$$\mathbf{D}_t = \Pi_{\mathcal{C}}\Big[\mathbf{D}_{t-1} - \frac{\rho}{t}\nabla_{\mathbf{D}} l(\mathbf{x}_t, \mathbf{D}_{t-1})\Big], \tag{6}$$

where ρ is the gradient step, $\Pi_{\mathcal{C}}$ is the orthogonal projector on $\mathcal{C}$, and the training set $\mathbf{x}_1, \mathbf{x}_2, \ldots$ are i.i.d. samples of the (unknown) distribution $p(\mathbf{x})$. As shown in Section 5, we have observed that this method can be competitive compared to batch methods with large training sets, when a good learning rate ρ is selected.

The dictionary learning method we present in the next section falls into the class of online algorithms based on stochastic approximations, processing one sample at a time, but exploits the specific structure of the problem to efficiently solve it. Contrary to classical first-order stochastic gradient descent, it does not require explicit learning rate tuning and minimizes a sequentially quadratic local approximations of the expected cost.

3. Online Dictionary Learning

We present in this section the basic components of our online algorithm for dictionary learning (Sections 3.1–3.3), as well as two minor variants which speed up our implementation (Section 3.4).

[4]We use "a.s." (almost sure) to denote convergence with probability one.

Algorithm 1 Online dictionary learning.

```
Require: $\mathbf{x} \in \mathbb{R}^m \sim p(\mathbf{x})$ (random variable and an algorithm to draw i.i.d samples of $p$), $\lambda \in \mathbb{R}$ (regularization parameter), $\mathbf{D}_0 \in \mathbb{R}^{m\times k}$ (initial dictionary), $T$ (number of iterations).
1: $\mathbf{A}_0 \leftarrow 0$, $\mathbf{B}_0 \leftarrow 0$ (reset the "past" information).
2: for $t = 1$ to $T$ do
3:    Draw $\mathbf{x}_t$ from $p(\mathbf{x})$.
4:    Sparse coding: compute using LARS
          $\boldsymbol{\alpha}_t \stackrel{\Delta}{=} \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}||\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}||_2^2 + \lambda||\boldsymbol{\alpha}||_1.$   (8)
5:    $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \frac{1}{2}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T$.
6:    $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \mathbf{x}_t\boldsymbol{\alpha}_t^T$.
7:    Compute $\mathbf{D}_t$ using Algorithm 2, with $\mathbf{D}_{t-1}$ as warm restart, so that
          $\mathbf{D}_t \stackrel{\Delta}{=} \arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda||\boldsymbol{\alpha}_i||_1,$
              $= \arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}\big(\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) - \mathrm{Tr}(\mathbf{D}^T\mathbf{B}_t)\big).$   (9)
8: end for
9: Return $\mathbf{D}_T$ (learned dictionary).
```

3.1. Algorithm Outline

Our algorithm is summarized in Algorithm 1. Assuming the training set composed of i.i.d. samples of a distribution $p(\mathbf{x})$, its inner loop draws one element $\mathbf{x}_t$ at a time, as in stochastic gradient descent, and alternates classical sparse coding steps for computing the decomposition $\boldsymbol{\alpha}_t$ of $\mathbf{x}_t$ over the dictionary $\mathbf{D}_{t-1}$ obtained at the previous iteration, with dictionary update steps where the new dictionary $\mathbf{D}_t$ is computed by minimizing over $\mathcal{C}$ the function

$$\hat{f}_t(\mathbf{D}) \stackrel{\Delta}{=} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda||\boldsymbol{\alpha}_i||_1, \tag{7}$$

where the vectors $\boldsymbol{\alpha}_i$ are computed during the previous steps of the algorithm. The motivation behind our approach is twofold:

• The quadratic function $\hat{f}_t$ aggregates the past information computed during the previous steps of the algorithm, namely the vectors $\boldsymbol{\alpha}_i$, and it is easy to show that it upperbounds the empirical cost $f_t(\mathbf{D}_t)$ from Eq. (1). One key aspect of the convergence analysis will be to show that $\hat{f}_t(\mathbf{D}_t)$ and $f_t(\mathbf{D}_t)$ converges almost surely to the same limit and thus $\hat{f}_t$ acts as a *surrogate* for f_t.

• Since $\hat{f}_t$ is close to $\hat{f}_{t-1}$, $\mathbf{D}_t$ can be obtained efficiently using $\mathbf{D}_{t-1}$ as warm restart.

Algorithm 2 Dictionary Update.

```
Require: $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$ (input dictionary),
    $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_k] \in \mathbb{R}^{k \times k} = \frac{1}{2}\sum_{i=1}^{t} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T$,
    $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k] \in \mathbb{R}^{m \times k} = \sum_{i=1}^{t} \mathbf{x}_i \boldsymbol{\alpha}_i^T$.
1: repeat
2:   for $j = 1$ to $k$ do
3:     Update the $j$-th column to optimize for (9):
         $$\mathbf{u}_j \leftarrow \frac{1}{\mathbf{A}_{jj}}(\mathbf{b}_j - \mathbf{D}\mathbf{a}_j) + \mathbf{d}_j.$$
         $$\mathbf{d}_j \leftarrow \frac{1}{\max(||\mathbf{u}_j||_2, 1)}\mathbf{u}_j.$$      (10)
4:   end for
5: until convergence
6: Return D (updated dictionary).
```

3.2. Sparse Coding

The sparse coding problem of Eq. (2) with fixed dictionary is an ℓ_1-regularized linear least-squares problem. A number of recent methods for solving this type of problems are based on coordinate descent with soft thresholding (Fu, 1998; Friedman et al., 2007). When the columns of the dictionary have low correlation, these simple methods have proven to be very efficient. However, the columns of learned dictionaries are in general highly correlated, and we have empirically observed that a Cholesky-based implementation of the LARS-Lasso algorithm, an homotopy method (Osborne et al., 2000; Efron et al., 2004) that provides the whole regularization path—that is, the solutions for all possible values of λ, can be as fast as approaches based on soft thresholding, while providing the solution with a higher accuracy.

3.3. Dictionary Update

Our algorithm for updating the dictionary uses block-coordinate descent with warm restarts, and one of its main advantages is that it is parameter-free and does not require any learning rate tuning, which can be difficult in a constrained optimization setting. Concretely, Algorithm 2 sequentially updates each column of $\mathbf{D}$. Using some simple algebra, it is easy to show that Eq. (10) gives the solution of the dictionary update (9) with respect to the j-th column $\mathbf{d}_j$, while keeping the other ones fixed under the constraint $\mathbf{d}_j^T\mathbf{d}_j \leq 1$. Since this convex optimization problem admits separable constraints in the updated blocks (columns), convergence to a global optimum is guaranteed (Bertsekas, 1999). In practice, since the vectors $\boldsymbol{\alpha}_i$ are sparse, the coefficients of the matrix $\mathbf{A}$ are in general concentrated on the diagonal, which makes the block-coordinate descent more efficient.[5] Since our algorithm uses the value of $\mathbf{D}_{t-1}$ as a warm restart for computing $\mathbf{D}_t$, a single iteration has empirically been found to be enough. Other approaches have been proposed to update $\mathbf{D}$, for instance, Lee et al. (2007) suggest using a Newton method on the dual of Eq. (9), but this requires inverting a $k \times k$ matrix at each Newton iteration, which is impractical for an online algorithm.

3.4. Optimizing the Algorithm

We have presented so far the basic building blocks of our algorithm. This section discusses simple improvements that significantly enhance its performance.

Handling Fixed-Size Datasets. In practice, although it may be very large, the size of the training set is often finite (of course this may not be the case, when the data consists of a video stream that must be treated on the fly for example). In this situation, the same data points may be examined several times, and it is very common in online algorithms to simulate an i.i.d. sampling of $p(\mathbf{x})$ by cycling over a randomly permuted training set (Bottou & Bousquet, 2007). This method works experimentally well in our setting but, when the training set is small enough, it is possible to further speed up convergence: In Algorithm 1, the matrices $\mathbf{A}_t$ and $\mathbf{B}_t$ carry all the information from the past coefficients $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_t$. Suppose that at time t_0, a signal $\mathbf{x}$ is drawn and the vector $\boldsymbol{\alpha}_{t_0}$ is computed. If the same signal $\mathbf{x}$ is drawn again at time $t > t_0$, one would like to remove the "old" information concerning $\mathbf{x}$ from $\mathbf{A}_t$ and $\mathbf{B}_t$—that is, write $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T - \boldsymbol{\alpha}_{t_0}\boldsymbol{\alpha}_{t_0}^T$ for instance. When dealing with large training sets, it is impossible to store all the past coefficients $\boldsymbol{\alpha}_{t_0}$, but it is still possible to partially exploit the same idea, by carrying in $\mathbf{A}_t$ and $\mathbf{B}_t$ the information from the current and previous *epochs* (cycles through the data) only.

Mini-Batch Extension. In practice, we can improve the convergence speed of our algorithm by drawing $\eta > 1$ signals at each iteration instead of a single one, which is a classical heuristic in stochastic gradient descent algorithms. Let us denote $\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,\eta}$ the signals drawn at iteration t. We can then replace the lines 5 and 6 of Algorithm 1 by

$$\begin{cases} \mathbf{A}_t & \leftarrow \beta\mathbf{A}_{t-1} + \sum_{i=1}^{\eta} \frac{1}{2}\boldsymbol{\alpha}_{t,i}\boldsymbol{\alpha}_{t,i}^T, \\ \mathbf{B}_t & \leftarrow \beta\mathbf{B}_{t-1} + \sum_{i=1}^{\eta} \mathbf{x}\boldsymbol{\alpha}_{t,i}^T, \end{cases} \tag{11}$$

where β is chosen so that $\beta = \frac{\theta+1-\eta}{\theta+1}$, where $\theta = t\eta$ if $t < \eta$ and $\eta^2 + t - \eta$ if $t \geq \eta$, which is compatible with our convergence analysis.

[5]Note that this assumption does not exactly hold: To be more exact, if a group of columns in $\mathbf{D}$ are highly correlated, the coefficients of the matrix $\mathbf{A}$ can concentrate on the corresponding principal submatrices of $\mathbf{A}$.

Purging the Dictionary from Unused Atoms. Every dictionary learning technique sometimes encounters situations where some of the dictionary atoms are never (or very seldom) used, which happens typically with a very bad intialization. A common practice is to replace them during the optimization by elements of the training set, which solves in practice this problem in most cases.

4. Convergence Analysis

Although our algorithm is relatively simple, its stochastic nature and the non-convexity of the objective function make the proof of its convergence to a stationary point somewhat involved. The main tools used in our proofs are the convergence of empirical processes (Van der Vaart, 1998) and, following Bottou (1998), the convergence of quasi-martingales (Fisk, 1965). Our analysis is limited to the basic version of the algorithm, although it can in principle be carried over to the optimized version discussed in Section 3.4. Because of space limitations, we will restrict ourselves to the presentation of our main results and a sketch of their proofs, which will be presented in details elsewhere, and first the (reasonable) assumptions under which our analysis holds.

4.1. Assumptions

(A) The data admits a bounded probability density p with compact support K. Assuming a compact support for the data is natural in audio, image, and video processing applications, where it is imposed by the data acquisition process.
(B) The quadratic surrogate functions $\hat{f}_t$ are strictly convex with lower-bounded Hessians. We assume that the smallest eigenvalue of the semi-definite positive matrix $\frac{1}{t}\mathbf{A}_t$ defined in Algorithm 1 is greater than or equal to a non-zero constant κ_1 (making $\mathbf{A}_t$ invertible and $\hat{f}_t$ strictly convex with Hessian lower-bounded). This hypothesis is in practice verified experimentally after a few iterations of the algorithm when the initial dictionary is reasonable, consisting for example of a few elements from the training set, or any one of the "off-the-shelf" dictionaries, such as DCT (bases of cosines products) or wavelets. Note that it is easy to enforce this assumption by adding a term $\frac{\kappa_1}{2}||\mathbf{D}||_F^2$ to the objective function, which is equivalent in practice to replacing the positive semi-definite matrix $\frac{1}{t}\mathbf{A}_t$ by $\frac{1}{t}\mathbf{A}_t + \kappa_1\mathbf{I}$. We have omitted for simplicity this penalization in our analysis.
(C) A sufficient uniqueness condition of the sparse coding solution is verified: Given some $\mathbf{x} \in K$, where K is the support of p, and $\mathbf{D} \in \mathcal{C}$, let us denote by Λ the set of indices j such that $|\mathbf{d}_j^T(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)| = \lambda$, where $\boldsymbol{\alpha}^\star$ is the solution of Eq. (2). We assume that there exists $\kappa_2 > 0$ such that, for all $\mathbf{x}$ in K and all dictionaries $\mathbf{D}$ in the subset $\mathcal{S}$ of $\mathcal{C}$ considered by our algorithm, the smallest eigenvalue of $\mathbf{D}_\Lambda^T\mathbf{D}_\Lambda$ is greater than or equal to κ_2. This matrix is thus invertible and classical results (Fuchs, 2005) ensure the uniqueness of the sparse coding solution. It is of course easy to build a dictionary $\mathbf{D}$ for which this assumption fails. However, having $\mathbf{D}_\Lambda^T\mathbf{D}_\Lambda$ invertible is a common assumption in linear regression and in methods such as the LARS algorithm aimed at solving Eq. (2) (Efron et al., 2004). It is also possible to enforce this condition using an elastic net penalization (Zou & Hastie, 2005), replacing $||\boldsymbol{\alpha}||_1$ by $||\boldsymbol{\alpha}||_1 + \frac{\kappa_2}{2}||\boldsymbol{\alpha}||_2^2$ and thus improving the numerical stability of homotopy algorithms such as LARS. Again, we have omitted this penalization for simplicity.

4.2. Main Results and Proof Sketches

Given assumptions (A) to (C), let us now show that our algorithm converges to a stationary point of the objective function.

Proposition 1 (convergence of $f(\mathbf{D}_t)$ and of the surrogate function). *Let $\hat{f}_t$ denote the surrogate function defined in Eq. (7). Under assumptions (A) to (C):*
- *$\hat{f}_t(\mathbf{D}_t)$ converges a.s.;*
- *$f(\mathbf{D}_t) - \hat{f}_t(\mathbf{D}_t)$ converges a.s. to 0; and*
- *$f(\mathbf{D}_t)$ converges a.s.*

Proof sktech: The first step in the proof is to show that $\mathbf{D}_t - \mathbf{D}_{t-1} = O\big(\frac{1}{t}\big)$ which, although it does not ensure the convergence of $\mathbf{D}_t$, ensures the convergence of the series $\sum_{t=1}^{\infty} ||\mathbf{D}_t - \mathbf{D}_{t-1}||_F^2$, a classical condition in gradient descent convergence proofs (Bertsekas, 1999). In turn, this reduces to showing that $\mathbf{D}_t$ minimizes a parametrized quadratic function over $\mathcal{C}$ with parameters $\frac{1}{t}\mathbf{A}_t$ and $\frac{1}{t}\mathbf{B}_t$, then showing that the solution is uniformly Lipschitz with respect to these parameters, borrowing some ideas from perturbation theory (Bonnans & Shapiro, 1998). At this point, and following Bottou (1998), proving the convergence of the sequence $\hat{f}_t(\mathbf{D}_t)$ amounts to showing that the stochastic positive process

$$u_t \overset{\triangle}{=} \hat{f}_t(\mathbf{D}_t) \geq 0, \tag{12}$$

is a quasi-martingale. To do so, denoting by $\mathcal{F}_t$ the filtration of the past information, a theorem by Fisk (1965) states that if the positive sum $\sum_{t=1}^{\infty} \mathbb{E}[\max(\mathbb{E}[u_{t+1} - u_t|\mathcal{F}_t], 0)]$ converges, then u_t is a quasi-martingale which converges with probability one. Using some results on empirical processes (Van der Vaart, 1998, Chap. 19.2, Donsker Theorem), we obtain a bound that ensures the convergence of this series. It follows from the convergence of u_t that $f_t(\mathbf{D}_t) - \hat{f}_t(\mathbf{D}_t)$ converges to zero with probability one. Then, a classical theorem from perturbation theory (Bonnans & Shapiro, 1998, Theorem 4.1) shows that $l(\mathbf{x}, \mathbf{D})$

is C^1. This, allows us to use a last result on empirical processes ensuring that $f(\mathbf{D}_t) - \hat{f}_t(\mathbf{D}_t)$ converges almost surely to 0. Therefore $f(\mathbf{D}_t)$ converges as well with probability one. ■

Proposition 2 (convergence to a stationary point). *Under assumptions (A) to (C),* $\mathbf{D}_t$ *is asymptotically close to the set of stationary points of the dictionary learning problem with probability one.*

Proof sktech: The first step in the proof is to show using classical analysis tools that, given assumptions (A) to (C), f is C^1 with a Lipschitz gradient. Considering $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ two accumulation points of $\frac{1}{t}\mathbf{A}_t$ and $\frac{1}{t}\mathbf{B}_t$ respectively, we can define the corresponding surrogate function $\hat{f}_\infty$ such that for all $\mathbf{D}$ in $\mathcal{C}$, $\hat{f}_\infty(\mathbf{D}) = \text{Tr}(\mathbf{D}^T\mathbf{D}\tilde{\mathbf{A}}) - \text{Tr}(\mathbf{D}^T\tilde{\mathbf{B}})$, and its optimum $\mathbf{D}_\infty$ on $\mathcal{C}$. The next step consists of showing that $\nabla\hat{f}_\infty(\mathbf{D}_\infty) = \nabla f(\mathbf{D}_\infty)$ and that $-\nabla f(\mathbf{D}_\infty)$ is in the normal cone of the set $\mathcal{C}$—that is, $\mathbf{D}_\infty$ is a stationary point of the dictionary learning problem (Borwein & Lewis, 2006). ■

5. Experimental Validation

In this section, we present experiments on natural images to demonstrate the efficiency of our method.

5.1. Performance evaluation

For our experiments, we have randomly selected 1.25×10^6 patches from images in the Berkeley segmentation dataset (Martin et al., 2001), which is a standard image database; 10^6 of these are kept for training, and the rest for testing. We used these patches to create three datasets A, B, and C with increasing patch and dictionary sizes representing various typical settings in image processing applications:

Data	Signal size m	Nb k of atoms	Type
A	$8 \times 8 = 64$	256	b&w
B	$12 \times 12 \times 3 = 432$	512	color
C	$16 \times 16 = 256$	1024	b&w

We have normalized the patches to have unit ℓ_2-norm and used the regularization parameter $\lambda = 1.2/\sqrt{m}$ in all of our experiments. The $1/\sqrt{m}$ term is a classical normalization factor (Bickel et al., 2007), and the constant 1.2 has been experimentally shown to yield reasonable sparsities (about 10 nonzero coefficients) in these experiments. We have implemented the proposed algorithm in C++ with a Matlab interface. All the results presented in this section use the mini-batch refinement from Section 3.4 since this has shown empirically to improve speed by a factor of 10 or more. This requires to tune the parameter η, the number of signals drawn at each iteration. Trying different powers of 2 for this variable has shown that $\eta = 256$ was a good choice (lowest objective function values on the training set — empirically, this setting also yields the lowest values on the test set), but values of 128 and and 512 have given very similar performances.

Our implementation can be used in both the online setting it is intended for, and in a regular batch mode where it uses the entire dataset at each iteration (corresponding to the mini-batch version with $\eta = n$). We have also implemented a first-order stochastic gradient descent algorithm that shares most of its code with our algorithm, except for the dictionary update step. This setting allows us to draw meaningful comparisons between our algorithm and its batch and stochastic gradient alternatives, which would have been difficult otherwise. For example, comparing our algorithm to the Matlab implementation of the batch approach from (Lee et al., 2007) developed by its authors would have been unfair since our C++ program has a built-in speed advantage. Although our implementation is multi-threaded, our experiments have been run for simplicity on a single-CPU, single-core 2.4Ghz machine. To measure and compare the performances of the three tested methods, we have plotted the value of the objective function on *the test set*, acting as a surrogate of the expected cost, as a function of the corresponding training time.

Online vs Batch. Figure 1 (top) compares the online and batch settings of our implementation. The full training set consists of 10^6 samples. The online version of our algorithm draws samples from the entire set, and we have run its batch version on the full dataset as well as subsets of size 10^4 and 10^5 (see figure). The online setting systematically outperforms its batch counterpart for every training set size and desired precision. We use a logarithmic scale for the computation time, which shows that in many situations, the difference in performance can be dramatic. Similar experiments have given similar results on smaller datasets.

Comparison with Stochastic Gradient Descent. Our experiments have shown that obtaining good performance with stochastic gradient descent requires using both the mini-batch heuristic *and* carefully choosing the learning rate ρ. To give the fairest comparison possible, we have thus optimized these parameters, sampling η values among powers of 2 (as before) and ρ values among powers of 10. The combination of values $\rho = 10^4$, $\eta = 512$ gives the best results on the training and test data for stochastic gradient descent. Figure 1 (bottom) compares our method with stochastic gradient descent for different ρ values around 10^4 and a fixed value of $\eta = 512$. We observe that the larger the value of ρ is, the better the eventual value of the objective function is after many iterations, but the longer it will take to achieve a good precision. Although our method performs better at such high-precision settings for dataset

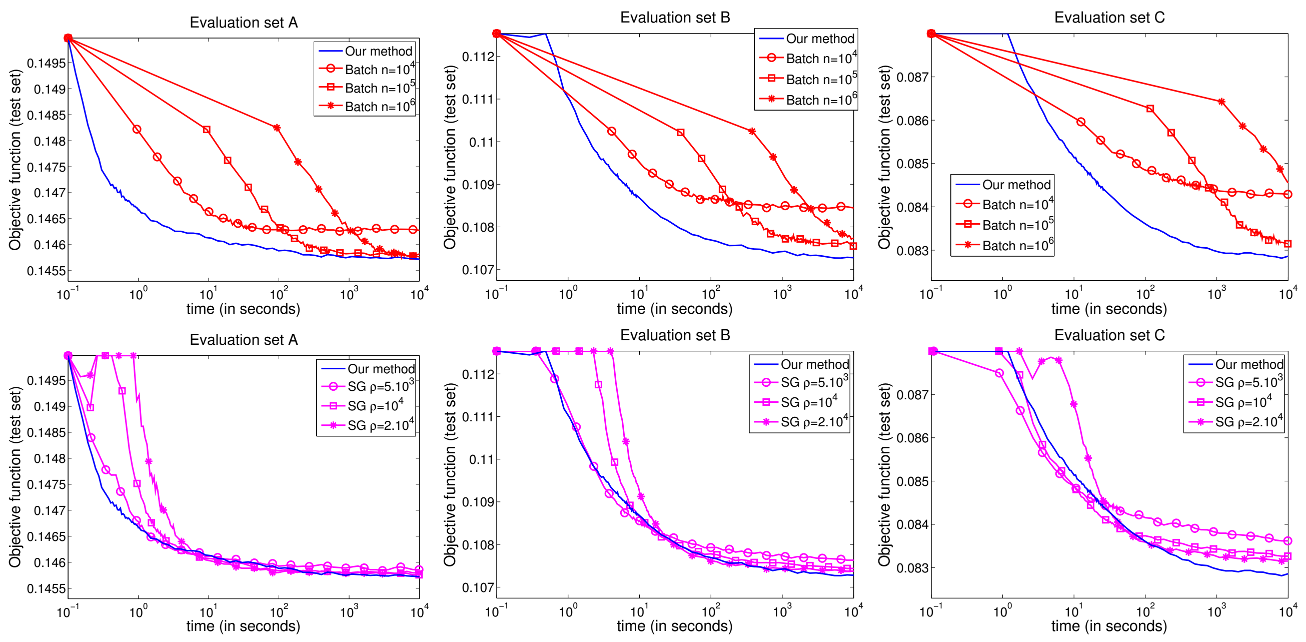


Figure 1. **Top:** Comparison between online and batch learning for various training set sizes. **Bottom:** Comparison between our method and stochastic gradient (SG) descent with different learning rates ρ. In both cases, the value of the objective function evaluated on the test set is reported as a function of computation time on a logarithmic scale. Values of the objective function greater than its initial value are truncated.

C, it appears that, in general, for a desired precision and a particular dataset, it is possible to tune the stochastic gradient descent algorithm to achieve a performance similar to that of our algorithm. Note that both stochastic gradient descent and our method only start decreasing the objective function value after a few iterations. Slightly better results could be obtained by using smaller gradient steps during the first iterations, using a learning rate of the form $\rho/(t+t_0)$ for the stochastic gradient descent, and initializing $\mathbf{A}_0 = t_0\mathbf{I}$ and $\mathbf{B}_0 = t_0\mathbf{D}_0$ for the matrices $\mathbf{A}_t$ and $\mathbf{B}_t$, where t_0 is a new parameter.

5.2. Application to Inpainting

Our last experiment demonstrates that our algorithm can be used for a difficult large-scale image processing task, namely, removing the text (*inpainting*) from the damaged 12-Megapixel image of Figure 2. Using a multi-threaded version of our implementation, we have learned a dictionary with 256 elements from the roughly 7×10^6 undamaged 12×12 color patches in the image with two epochs in about 500 seconds on a 2.4GHz machine with eight cores. Once the dictionary has been learned, the text is removed using the sparse coding technique for inpainting of Mairal et al. (2008b). Our intent here is of course *not* to evaluate our learning procedure in inpainting tasks, which would require a thorough comparison with state-the-art techniques on standard datasets. Instead, we just wish to demonstrate that the proposed method can indeed be applied to a realistic, non-trivial image processing task on a large image. Indeed, to the best of our knowledge, this is the first time that dictionary learning is used for image restoration on such large-scale data. For comparison, the dictionaries used for inpainting in the state-of-the-art method of Mairal et al. (2008b) are learned (in batch mode) on only 200,000 patches.

6. Discussion

We have introduced in this paper a new stochastic online algorithm for learning dictionaries adapted to sparse coding tasks, and proven its convergence. Preliminary experiments demonstrate that it is significantly faster than batch alternatives on large datasets that may contain millions of training examples, yet it does not require learning rate tuning like regular stochastic gradient descent methods. More experiments are of course needed to better assess the promise of this approach in image restoration tasks such as denoising, deblurring, and inpainting. Beyond this, we plan to use the proposed learning framework for sparse coding in computationally demanding video restoration tasks (Protter & Elad, 2009), with dynamic datasets whose size is not fixed, and also plan to extend this framework to different loss functions to address discriminative tasks such as image classification (Mairal et al., 2008a), which are more sensitive to overfitting than reconstructive ones, and various matrix factorization tasks, such as non-negative matrix factorization with sparseness constraints.

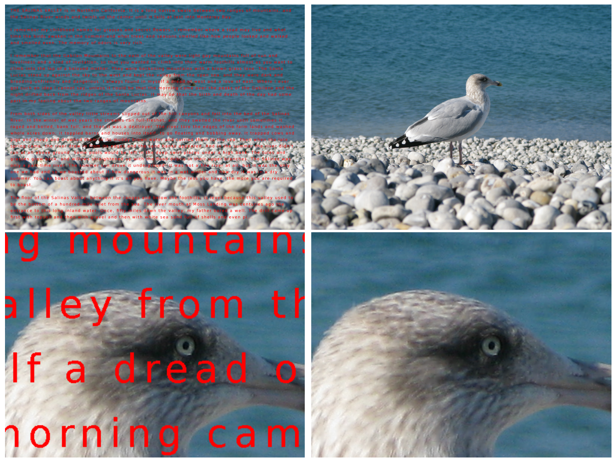


Figure 2. Inpainting example on a 12-Megapixel image. Top: Damaged and restored images. Bottom: Zooming on the damaged and restored images. (Best seen in color)

Acknowledgments

This paper was supported in part by ANR under grant MGA. The work of Guillermo Sapiro is partially supported by ONR, NGA, NSF, ARO, and DARPA.

References

Aharon, M., & Elad, M. (2008). Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM J. on Im. Sc.*.

Aharon, M., Elad, M., & Bruckstein, A. M. (2006). The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Trans. SP*.

Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific Belmont, Mass.

Bickel, P., Ritov, Y., & Tsybakov, A. (2007). Simultaneous analysis of Lasso and Dantzig selector. preprint.

Bonnans, J., & Shapiro, A. (1998). Optimization problems with perturbation: A guided tour. *SIAM Review*.

Borwein, J., & Lewis, A. (2006). *Convex analysis and nonlinear optimization: theory and examples*. Springer.

Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (Ed.), *Online learning and neural networks*.

Bottou, L., & Bousquet, O. (2007). The tradeoffs of large scale learning. *Adv. NIPS*.

Chen, S., Donoho, D., & Saunders, M. (1999). Atomic decomposition by basis pursuit. *SIAM J. on Sc. Comp.*.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. IP*.

Engan, K., Aase, S. O., & Husoy, J. H. (1999). Frame based signal compression using method of optimal directions (MOD). *Proc. of the IEEE Intern. Symp. Circ. Syst.*.

Fisk, D. (1965). Quasi-martingale. *Transactions of the American Mathematical Society*.

Friedman, J., Hastie, T., Hölfling, H., & Tibshirani, R. (2007). Pathwise coordinate optimization. *Annals of statistics*.

Fu, W. (1998). Penalized Regressions: The Bridge Versus the Lasso. *J. of comp. and graph. stats*.

Fuchs, J. (2005). Recovery of exact sparse representations in the presence of bounded noise. *IEEE Trans. IT*.

Kushner, H., & Yin, G. (2003). *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Lee, H., Battle, A., Raina, R., & Ng, A. Y. (2007). Efficient sparse coding algorithms. *Adv. NIPS*.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2008a). Supervised dictionary learning. *Adv. NIPS*.

Mairal, J., Elad, M., & Sapiro, G. (2008b). Sparse representation for color image restoration. *IEEE Trans. IP*.

Mallat, S. (1999). *A wavelet tour of signal processing, second edition*. Academic Press, New York.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proc. ICCV*.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*.

Osborne, M., Presnell, B., & Turlach, B. (2000). A new approach to variable selection in least squares problems. *IMA J. of Num. Analysis*.

Protter, M., & Elad, M. (2009). Image sequence denoising via sparse and redundant representations. *IEEE Trans. IP*.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. *Proc. ICML*.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *J. Royal. Statist. Soc B.*.

Van der Vaart, A. (1998). *Asymptotic Statistics*. Cambridge University Press.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*.