Digital Signal Processing 55 (2016) 44–51

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Fast convolutional sparse coding using matrix inversion lemma

Michal Šorel [*], Filip Šroubek

Institute of Information Theory and Automation, Czech Academy of Sciences, Pod Vodárenskou věží 4, 182 08 Prague 8, Czech Republic

A R T I C L E   I N F O

Article history:
Available online 3 May 2016

Keywords:
Convolutional sparse coding
Feature learning
Deconvolution networks
Shift-invariant sparse coding

A B S T R A C T

Convolutional sparse coding is an interesting alternative to standard sparse coding in modeling shift-invariant signals, giving impressive results for example in unsupervised learning of visual features. In state-of-the-art methods, the most time-consuming parts include inversion of a linear operator related to convolution. In this article we show how these inversions can be computed non-iteratively in the Fourier domain using the matrix inversion lemma. This greatly speeds up computation and makes convolutional sparse coding computationally feasible even for large problems. The algorithm is derived in three variants, one of them especially suitable for parallel implementation. We demonstrate algorithms on two-dimensional image data but all results hold for signals of arbitrary dimension.

1. Introduction

Sparse coding methods learn a dictionary of basis vectors or functions so that observed data could be expressed as a linear combination of only a small number of these atoms [1]. Sparse coding first appeared in neuroscience as a model of visual cortex [2] but found applications in many classification and signal reconstruction tasks. In machine learning sparsity avoids over-fitting and can be thought of as a tool for feature extraction. In signal reconstruction sparse coding can serve as a form of Bayesian prior for image denoising [3], inpainting [4], deblurring [5], super-resolution [6] and audio signal representation [7]. Although finding the dictionary with which the training signals can be represented with optimal sparsity is strongly NP-hard [8], there is a number of effective heuristic algorithms giving an approximate solution in polynomial time [9,10]. Sparse coding is closely related to compressed sensing [11], with results showing that for incoherent dictionaries only a small number of projections is sufficient to exactly reconstruct the original signal [12,13]. Efficient sparse coding algorithms with provable guarantees appeared only recently [14–16].

In image processing applications, both the observed data and dictionary atoms correspond to image patches. A fundamental disadvantage of sparse coding is the assumption that image patches are independent, which typically leads to many atoms being translated versions of one another. The same issue can be expected in audio signals. *Convolutional sparse coding*, also called shift-invariant sparse coding [17–19], is an interesting alternative that found its use in audio classification [20], deconvolutional networks [21] and predictive sparse coding by neural networks [22]. In contrast to standard sparse coding that models a signal as a sparse combination of dictionary vectors, convolutional sparse coding models the signal as a sum of several convolutions of kernels and sparse feature maps.

The goal of this article is to describe a new fast algorithm for convolutional sparse coding. Our solution is based on the fact that the main problem of state-of-the-art algorithms [21,23] is a time consuming inversion of an operator related to convolution. This problem was sidestepped in [24] by using FISTA [25], where the inversion step is essentially replaced by one gradient descent step at the cost of much larger number of iterations necessary to achieve the same precision. In this paper, we adopt an approach close to [23] but show how the most time-consuming step of their algorithm can be computed non-iteratively in the Fourier domain using the matrix inversion lemma, which greatly speeds up computation. Derivation is relatively straightforward for one input signal [26] but more complicated for multiple inputs [27]. As our main contribution, we show three solutions of the multiple-input case, which are all equivalent for the single-input case. One of them is especially suitable for parallel implementation. We also compare efficiency of [21,23] and our algorithm under various conditions and demonstrate the ability of the proposed algorithms to learn kernels at several scales simultaneously. A paper using similar ideas to solve the problem of convolution sparse coding from incomplete data appeared recently in [28].

The rest of the paper is organized as follows. Sec. 2 states the problem of convolutional sparse coding. Sec. 3 shortly explains the main optimization tool we use, which is the alternating direc-

[*] Corresponding author.
E-mail addresses: sorel@utia.cas.cz (M. Šorel), sroubekf@utia.cas.cz (F. Šroubek).

http://dx.doi.org/10.1016/j.dsp.2016.04.012
1051-2004/© 2016 Elsevier Inc. All rights reserved.

Table 1
Notation.

z_k	feature maps $1\dots K$
d_k	convolutional kernels $1\dots K$
K	number of kernels and feature maps
N	number of pixels
L	number of input images
d_k^i	kth convolution kernel for input image i used in the consensus version of the algorithm
Z_k, D_k	operators corresponding to convolution with z_k and d_k
β	parameter of convolutional sparse coding balancing sparsity and accuracy
u_z, u_d	ADMM auxiliary variables for feature maps and convolution kernels
v_z, v_d	ADMM dual variables
λ	ADMM parameter
P	number of ADMM iterations

tion method of multipliers. Algorithms are derived in Sec. 4. Time complexity of the algorithms is summarized in Sec. 5 followed by experiments in Sec. 6 and conclusions in Sec. 7.

2. Convolutional sparse coding

Building on the analogy with compressed sensing, where the sparse representation is provably recoverable using l_1 norm [13], the convolutional sparse coding can be stated as a bi-convex problem

$$\min_{d,z} \frac{1}{2}\left\| y - \sum_{k=1}^{K} d_k * z_k \right\|^2 + \beta \sum_{k=1}^{K} \|z_k\|_1 \quad \text{s.t.} \quad \|d_k\| \le 1, \tag{1}$$

where y is an observed signal, z_k are sparse *feature maps* and d_k corresponding *convolution kernels*. The number of convolution kernels K and positive scalar β are user parameters. Complete list of used variables is provided in Table 1. In this paper, we assume circular boundary conditions, i.e. equivalence of convolution with element-wise multiplication in the Fourier domain, therefore the feature maps are of the same size and dimension as the observed signal. Motivated by applications in learning visual features and modeling image data, we use two-dimensional images in our experiments but convolutional sparse coding can be applied to signals of arbitrary dimension.

Analogously to standard sparse coding and other machine learning and signal modeling approaches, we are interested in two different modes of operation. First we learn convolution kernels from training data by solving the optimization problem (1) as stated above. We call this phase *kernel learning*. Second, in *feature extraction* phase, the kernels are fixed and features are computed only by minimization over feature maps. Even though the terms feature extraction and kernel learning come from machine learning, the same operations are important even for signal modeling and reconstruction. The role of kernel learning is to estimate an *a priori* signal distribution and feature extraction corresponds to Bayesian inference from noisy measurements.

As in sparse coding [10,29], all efficient methods of kernel learning in the literature [21,23,24] alternately minimize over the feature maps z_k while keeping the filters d_k fixed and over the filters while keeping the feature maps fixed, taking the advantage that both sub-problems are convex. In this way the feature extraction is essentially run in each iteration of the kernel learning algorithm.

3. Alternating direction method of multipliers

The main optimization tool we use in both convex sub-problems, which was less efficiently used already in [23], is the alternating direction method of multipliers (ADMM) [30]. Here we present its simplified form, which is equivalent to Douglas–Rachford splitting algorithm [31]. ADMM is a method to minimize the sum of two convex not necessarily differentiable functions

$$\min_x f(x) + g(x). \tag{2}$$

The algorithm is especially useful, if we can efficiently compute a so-called proximal or proximity operator of both functions, defined for f as

$$\operatorname{prox}_{\lambda f}(a) = \arg\min_x \lambda f(x) + \frac{1}{2}\|x - a\|^2 \tag{3}$$

and similarly for g, where scalar $\lambda > 0$ is a parameter. ADMM consists of iteratively executing three update steps

$$x \leftarrow \operatorname{prox}_{\lambda f}(v - u) \tag{4}$$

$$v \leftarrow \operatorname{prox}_{\lambda g}(x + u) \tag{5}$$

$$u \leftarrow u + x - v \tag{6}$$

two of them being computations of proximal operators for f and g, and (6) is a simple update of an auxiliary variable. Stopping criteria for ADMM are discussed in [30], Section 3.3.1.

4. Algorithm

In this section, we show that both convolutional sparse coding sub-problems can be written as a sum of two functions suitable for ADMM and derive how to efficiently compute their proximal operators. The main difference with respect to [23] is much faster computation of one of the proximal operators.

4.1. Minimization over feature maps

We start with the minimization of (1) over feature maps z_i, which can be written in short as

$$\min_z \frac{1}{2}\|y - Dz\|^2 + \beta\|z\|_1 \tag{7}$$

where $D = [D_1, \dots, D_K]$ is an operator composed of convolutions with K kernels d_k and $z = \left[z_1^T, \dots, z_K^T\right]^T$ is a vector of vectorized feature maps. K denotes the number of feature maps and β is a parameter balancing model accuracy and sparsity of the representation. The number of elements in y will be denoted by N. Note that from this place on, we use vector notation, where quantities y, z, etc. are vectors and convolutions with d_k and z_k are expressed as multiplications with circulant (for 2D data block-circulant) matrices D_k and Z_k, respectively.

This is a special case of l_1-regularized linear regression also called Lasso [32]. Authors of [21] solved (7) by a continuation approach, [23,33] used ADMM, decomposing (7) into two functions

$$f_z(z) = \frac{1}{2}\|y - Dz\|^2 \quad \text{and} \tag{8}$$

$$g_z(z) = \beta\|z\|_1 . \tag{9}$$

The proximal operator of l_1 norm

$$\arg\min_x \alpha\|x\|_1 + \frac{1}{2}\|x - a\|^2 \tag{10}$$

is a very fast element-wise operation called soft thresholding [30]

$$\operatorname{prox}_{\alpha|x|}(a) = S_\alpha(a) = \begin{cases} a - \alpha & a > \alpha \\ 0 & |a| \le \alpha \\ a + \alpha & a < \alpha, \end{cases} \tag{11}$$

where in our case $\alpha = \lambda\beta$.