2-28-2020

Dictionary Learning for Image Reconstruction via Numerical Non-convex Optimization Methods

Lewis M. Hicks
Portland State University

Dictionary Learning for Image Reconstruction via Numerical Non-convex Optimization Methods

Lewis Hicks[1]

An Undergraduate Honors Thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in University Honors and Mathematics with Honors

Thesis advisor: Mau Nam Nguyen[2]

Portland State University

2020

Abstract: This thesis explores image dictionary learning via non-convex (difference of convex, DC) programming and its applications to image reconstruction. First, the image reconstruction problem is detailed and solutions are presented. Each such solution requires an image dictionary to be specified directly or to be learned via non-convex programming. The solutions explored are the DCA (DC algorithm) and the boosted DCA. These various forms of dictionary learning are then compared on the basis of both image reconstruction accuracy and number of iterations required to converge.

[1]Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA. lmh6@pdx.edu

[2]Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA. mnn3@pdx.edu

Contents

1 Introduction

Convex optimization has been strongly developed since the 1960s, providing minimization techniques to solve many real-world problems. However, a challenge in modern optimization is to go from convexity to nonconvexity as nonconvex optimization problems appear frequently in many applications. This is the motivation for the search for new optimization methods to deal with broader classes of functions and sets where convexity is not assumed. One of the most successful approaches to go beyond convexity is to consider the class of DC (difference of convex) functions. Given a linear space X and two convex functions $g, h : X \to \mathbb{R}$, a DC optimization program minimizes $f = g - h$. It was recognized early by P. Hartman [7] that the class of DC functions exhibits many convenient algebraic properties. This class of functions is closed under many operations usually considered in optimization. In particular, it is closed with respect to taking linear combinations, maxima, and finite products of DC functions. Another nice feature of DC programming is that it possesses a very nice duality theory; see [16] and the references therein. Generalized differential properties of DC functions were investigated by Hirriart Urruty (see [8]).

Although the role of DC functions has been known earlier in optimization theory, the first algorithmic approach was developed by Pham Dinh Tao in 1985. The algorithm introduced by Pham Dinh Tao for minimizing $f = g - h$, called the DCA, is based on subgradients of the function h and subgradients of the Fenchel conjugate of the function g. This algorithm is summarized as follows: for each $x_k \in \mathbb{R}^n$, define $y_k \in \partial h(x_k)$ and $x_{k+1} \in \partial g^*(y_k)$. Under suitable conditions on the DC decomposition of the function f, the two sequences $\{x_k\}$ and $\{y_k\}$ in the DCA satisfy the monotonicity conditions in the sense that $\{g(x_k) - h(x_k)\}$ and $\{h^*(y_k) - g^*(y_k)\}$ are both decreasing. In addition, the sequences $\{x_k\}$ and $\{y_k\}$ converge to *critical points* of the primal function $g - h$ and the dual function $h^* - g^*$, respectively; see [2, 16, 17] and the references therein. The DCA is an effective algorithm for solving many nonconvex optimization problems without requiring the differentiability of the data. However, to deal with optimization problems of large scale, it is necessary to develop new optimization techniques to accelerate the convergence rate of this algorithm.

In this project, we focus on non-convex methods for developing an image dictionary. The image dictionary is used for the purposes of image reconstruction: a black and white digital image (bit depth 8) $\widetilde{M}$ of size $N_1 \times N_2$, expressed in a vectorized form $M \in \mathbb{R}^{N_1 N_2}$ is corrupted with a sampling operator A and Gaussian noise ξ. Using only the blurred image $b = A(M) + \xi$, we can restore the image using a version of the FISTA; see, e.g., [14, 22] and the references therein.

A vector is referred to as sparse when many of its entries are zeros. An image $y \in \mathbb{R}^{N_1 N_2}$ (in vectorized form) is said to have a sparse representation under if there is some $N_1 N_2 \times K$ matrix D, known as a dictionary, and a vector $h \in \mathbb{R}^K$ such that $y = Dh$. In this case, the dictionary D maps a sparse vector to a full image. The columns of D are called *atoms*, and given a suitable dictionary in this model, theoretically any image can be built from a linear combination of the columns (atoms) of the dictionary. Using a clever choice of dictionary allows us to work with sparse vectors, thereby reducing the amount of computer memory

Sampled image (SR=50%)

Recovered Image

needed to store an image. Further, sparse representations tend to capture the true image without extraneous noise.

The dictionary can be either directly specified using the DCT (disrete cosine transform) or using the DCA. In this paper, we present ways to accelerate the DCA used for dictionary learning, then compare the results of each type of enhancement used. A better dictionary should yield improved image restorations and converge in fewer iterations.

2 Problem formulation and accomplished goals

Although the focus of this work is on image dictionary learning, it is important to understand the specific application of image dictionaries: image restoration.

Consider a dictionary $D \in \mathbb{R}^{n_1 n_2 \times K}$ and an observed image $b \in \mathbb{R}^{n_1 n_2}$ which has been corrupted by a linear operator A and distorted by some noise ξ. A vectorized image $y \in \mathbb{R}^{n_1 n_2}$ is a "good" image if it has a sparse representation $h \in \mathbb{R}^K$ under the dictionary D, i.e.,

$$y = Dh, \text{ where } h \text{ is sparse.}$$

We require that $A(y) = A(Dh)$ be as close to the corrupted image b as possible by minimizing $\|A(Dh) - b\|^2$, while making sure that h is sparse. We thus add an additional regularization term to $\|A(Dh) - b\|^2$ to induce sparsity. The classical approach involves using the ℓ_1 norm regularization:

$$\text{minimize } \frac{1}{2}\|A(Dh) - b\|^2 + \lambda\|h\|_1,$$

where $\lambda > 0$ is a parameter. To solve this problem, we can use the FISTA (fast iterative shrinkage thresholding algorithm) to reconstruct images; see [4]. The goal of this work is to find the dictionary which most accurately and quickly reconstructs images. One option is to directly specify the dictionary using the DCT, given by

$$D_{ij} = \begin{cases} \sqrt{\frac{1}{n_1 n_2}}, & j = 1 \\ \sqrt{\frac{2}{n_1 n_2}} \cos(\frac{\pi}{n_1 n_2}(j-1)(i+\frac{1}{2})), & j = 2, ..., n_1 n_2. \end{cases}$$

However, more accurate dictionaries can be learned from a set of training images. Consider a set of T training images, each of size $n_1 \times n_2$. We represent each training image $x_t \in \mathbb{R}^{n_1 n_2}$ $(1 \leq t \leq T)$ as a column of a matrix $X \in \mathbb{R}^{n_1 n_2 \times T}$. The sparse representation of X under D is denoted by $H \in \mathbb{R}^{K \times T}$ (ideally, $DH = X$). In order to ensure that each column of the dictionary has equal weight, we impose the requirement that each column of D is at most of norm 1. We therefore need to find a dictionary which meets three criteria:

- The dictionary lends itself to accurately representing patches in the training set: $||DH - X||_F^2$ is small.
- The representation H must be sparse: $||H||_0$ is small.
- Each column of D is of at most norm 1: $||d_j||_2 \leq 1$ for $1 \leq j \leq K$.

Since the ℓ_0 norm in the second criterion is not continuous, we approximate it with a *regularization*: $||H||_0 \approx ||H||_{11} - ||H||_{21}$. In order to meet each criteria simultaneously, we define the function f^λ by

$$f^\lambda(D, H) = \frac{\lambda}{2} ||DH - X||_F^2 + ||H||_{11} - ||H||_{21}.$$

The parameter $\lambda > 0$ represents the tradeoff between sparsity in H and accuracy in D. We therefore seek to solve the optimization problem

$$\text{minimize } f^\lambda(D, H),$$

subject to $||d_j||_2 \leq 1$ for $1 \leq j \leq K$.

Unlike the problem of image reconstruction, the dictionary learning requires optimization with respect to two parameters. In order to solve such problems, two steps are repeated: finding the sparse matrix H and updating the dictionary D. Both processes are accomplished via the DCA and its accelerated versions.

3 Image reconstruction

In this section we provide the details of image reconstructions via convex and nonconvex optimization methods. The readers are referred to [14, 22] and the references therein for further information.

3.1 Patching

Through dividing the image into smaller pieces before beginning image reconstruction, improved results and execution speed are achieved. Patching is the process of dividing an $N_1 \times N_2$ image into smaller rectangular subdivisions of size $n_1 \times n_2$. The patches will be indexed by row $(1 \leq i \leq t_1)$ and column $(1 \leq j \leq t_2)$, where t_1 and t_2 are the number of patches per row and number of patches per column of the original image, respectively.

First, the original image $\widetilde{M} \in \mathbb{R}^{N_1 \times N_2}$ is *vectorized* by adjoining the columns of $\widetilde{M}$ end-to-end. In particular, if $m_1, m_2, ..., m_{N_2} \in \mathbb{R}^{N_1}$ are the columns $\widetilde{M}$, then $\widetilde{M} = [m_1 m_2 ... m_{N_2}]$ and its vectorized form is $M = [m_1^\top m_2^\top ... m_{N_2}^\top]^\top$. The same vectorization process can be applied to any matrix A. The vectorized form of A is denoted by $v(A)$.

For the patch in the ith row and the jth column, a *patch extraction matrix* $R_{ij} \in \mathbb{R}^{n_1 n_2 \times N_1 N_2}$ is defined through the indices of its upper-left corner (s, t), its number of rows n_1 and its number of columns n_2. In order to build R_{ij}, an indexing matrix $J \in \mathbb{R}^{n_1 \times n_2}$ is first defined by

$$J_{rq} = N_1((t-1) + (q-1)) + s + (r-1)$$

for $1 \leq q \leq n_2$ and $1 \leq r \leq n_1$. Next, the matrix J is vectorized by v and used to define each row $r_k \in \mathbb{R}^{N_1 N_2}$ $(1 \leq k \leq n_1 n_2)$ of R_{ij}:

$$r_k = e_{v(J)_k}^\top,$$

where $\{e_k : k \in \{1, ...N_1 N_2\}\}$ is the set of standard basis vectors of $\mathbb{R}^{N_1 N_2}$. Thus, the patch extraction matrix can be framed as an identity matrix with missing rows. Note that the patch extraction matrices do not depend on the contents of the original image, only its size. Therefore, a set of patching matrices can be generated once, saved to a file, and reused. The vectorized patch of the original image at index (i, j) is given by $P_{ij} = R_{ij} M \in \mathbb{R}^{n_1 n_2}$.

3.2 Sampling and noise

In order to distort the original vectorized image M, a fraction of pixels are removed and Gaussian noise is added. Given a sample rate $S \in [0, 1]$, a set $\Omega \subseteq \{1, 2, ..., N_1 N_2\}$ represents which pixels of the image are sampled. For $1 \leq k \leq N_1 N_2$, a real number $\omega_k \in [0, 1]$ is chosen at random. If $\omega_k \leq S$, then $k \in \Omega$.

Next, each row of a *sampling operator* $A \in \mathbb{R}^{|\Omega| \times N_1 N_2}$ is defined by

$$A_{k:} = e_k^\top$$

for all $k \in \Omega$, where $\{e_k : k \in \{1, ...N_1 N_2\}\}$ is the set of standard basis vectors of $\mathbb{R}^{N_1 N_2}$. Given a vectorized image $M \in \mathbb{R}^{N_1 N_2}$, $AM \in R^{|\Omega|}$ therefore represents the original image with $N_1 N_2 - |\Omega|$ pixels deleted. Next, random noise $\xi \in \mathbb{R}^{|\Omega|}$ is generated and added to create the blurred vectorized image $b = AM + \xi$.

3.3 Reconstructions of small images

In this section, we show how to apply techniques for general image restoration to a small blurred image b. The restored patch of size $n_1 \times n_2$ (usually 8×8) can be considered as a part of a larger image.

Since the sample operator for the entire image is large, computing products with it is inefficient. Furthermore, it does not need to be explicitly calculated. For each patch extraction

operator R_{ij}, we define $\mathcal{A} = A(R_{ij}^\top D)$. The value of $\mathcal{A}$ does not need to be found explicitly, so in practice functions $x \mapsto \mathcal{A}x$ and $x \mapsto \mathcal{A}^\top x$ are computed for each patch.

The goal of our optimization for each patch is to find a vector $h \in \mathbb{R}^K$ such that $y = Dh$ is close to the blurry patch b under the sample operator $\mathcal{A}$ and h is very sparse. Here, h is called the *sparse representation of y under D*. In essence, finding the value of h amounts to simultaneously minimizing two terms: an error term $\frac{1}{2}\|\mathcal{A}h - b\|^2$ and a sparsity penalty term $\|h\|_0$. However, the ℓ_0 norm cannot be used because it returns a discrete value (the integer number of non-zero entries in h). Therefore, we use the ℓ_1 regularization: $||h||_0 \approx ||h||_1$. Combining the two terms yields the overall function

$$f(h) = \frac{1}{2}\|\mathcal{A}h - b\|^2 + \lambda \, ||h||_1 \,, \tag{3.1}$$

where $\lambda > 0$ is a weight parameter which determines how sensitive the optimization is to the sparsity of h. By finding h for each patch of the image and recombining all patches, the restored image is generated.

3.3.1 The gradient descent method and Nesterov's acceleration

A straightforward way to optimize a function is via a *gradient descent* method. Given a differentiable and convex function $f : \mathbb{R}^n \to \mathbb{R}$, we know that ∇f is a vector which points in the direction of fastest increase. After starting at some initial guess h_0, we can move in the direction of fastest decrease, ie. the direction of $-\nabla f$. In addition, we specify a *step size* t for each iteration. The general gradient descent algorithm is as follows:

Gradient descent algorithm
INPUT: $h_0 \in \mathbb{R}^n, N \in \mathbb{N}, t_0, t_2, ..., t_N > 0$.
for $k \in \{0, \ldots, N\}$
  Set $h_{k+1} := h_k - t_k \nabla f(h_k)$.
end

In the case where f is a C^1 function whose gradient is Lipschitz continuous with constant $\ell > 0$, an accelerated gradient method can be used to improve its convergence rate. One of the most well-known such methods is *Nesterov's accelerated gradient method* (see [24]):

Accelerated gradient descent
INPUT: $h_0 \in \mathbb{R}^n$, $N \in \mathbb{N}$, $\alpha \in (0, \frac{1}{\ell}]$
Set $y_1 := h_0$
for $k \in \{1, \ldots, N\}$
  Set $x_k := y_k - \alpha \nabla f(y_k)$
  Set $t_{k+1} := \frac{1+\sqrt{1+4t_k^2}}{2}$
  Set $y_{k+1} := x_k + \frac{t_k - 1}{t_{k+1}} (x_k - x_{k-1})$
end

3.3.2 Nesterov's smoothing

One way to solve the image reconstruction problem is to use a gradient descent approach. However, the objective function f, given by

$$f(h) = \frac{1}{2}\left|\left|\mathcal{A}h - b\right|\right|^2 + \lambda\left|\left|h\right|\right|_1$$

is convex, but is not differentiable everywhere. In particular, the $||\cdot||_1$ term is non-smooth. In order to solve this issue, Nesterov's smoothing technique is applied to generate a smooth approximation of $||\cdot||_1$; see [15]. This technique will be applied later to dictionary learning and appears often in optimization involving non-smooth functions.

Consider $x \in \mathbb{R}^n$. Then $||x||_1 = \sum_{i=1}^n |x_i|$. By definition, $|x_i| = \max\{-x_i, x_i\}$. For each x_i, it holds that $|x_i| = \max_{u_i \in [-1,1]}\{x_i u_i\}$.

If we define the unit box to be $Q := \overline{\mathbb{B}}_\infty(0;1)$ (the unit ball under the ℓ_∞ norm), it follows that

$$||x||_1 = \sum_{i=1}^n |x_i| = \sum_{i=1}^n \max_{u_i\in[-1,1]}\{x_i u_i\} = \max_{u\in Q}\sum_{i=1}^n x_i u_i = \max_{u\in Q}\{\langle x, u\rangle\}.$$

Nesterov's smoothing technique states that given a function q of the form

$$q(x) = \max_{u\in Q}\{\langle \mathcal{A}x, u\rangle - \phi(x)\},$$

q can be approximated by the smooth function q_μ, defined by

$$q_\mu(x) := \max_{u\in Q}\left\{\langle \mathcal{A}x, u\rangle - \phi(x) - \frac{\mu}{2}||x||_2^2\right\},$$

where $\mu > 0$ is a parameter which controls the smoothness of the approximation. In our case, we can use the approximation with $\mathcal{A} = I_n$ and $\phi = 0$. Thus, if we define $p = ||\cdot||_1$, then

$$\begin{aligned}
p(x) \approx p_\mu(x) &= \max_{u\in Q}\left\{\langle x, u\rangle - \frac{\mu}{2}||x||_2^2\right\} \\
&= -\frac{\mu}{2}\min_{u\in Q}\left\{-2\left\langle \frac{x}{\mu}, u\right\rangle + ||u||_2^2\right\} \\
&= -\frac{\mu}{2}\min_{u\in Q}\left\{-\left|\left|\frac{x}{\mu}\right|\right|_2^2 + \left|\left|\frac{x}{\mu}\right|\right|_2^2 + -2\left\langle \frac{x}{\mu}, u\right\rangle + ||u||_2^2\right\} \\
&= -\frac{\mu}{2}\min_{uinQ}\left\{\left|\left|\frac{x}{\mu}\right|\right|_2^2 + \left|\left|u - \frac{x}{\mu}\right|\right|_2^2\right\} \\
&= \frac{1}{2\mu}||x||_2^2 - \frac{\mu}{2}\min_{u\in Q}\left\{\left|\left|u - \frac{x}{\mu}\right|\right|_2\right\}^2 \\
&= \frac{1}{2\mu}||x||_2^2 - \frac{\mu}{2}d\left(\frac{x}{\mu}; Q\right)^2 \\
&= \frac{1}{2\mu}||x||_2^2 - \frac{\mu}{2}\left|\left|\frac{x}{\mu} - \Pi_Q\left(\frac{x}{\mu}\right)\right|\right|^2.
\end{aligned}$$

The last equalities follow since Q is a convex set. Note that the projection map is given by $[\Pi_Q(x)]_i = \max(-1, \min(1, x_i))$. The smoothed approximation has a gradient of

$$\nabla p_\mu(x) = \Pi_Q\left(\frac{x}{\mu}\right).$$

3.3.3 Smoothed gradient descent for image reconstruction

We can approximate f by f_μ, where

$$f_\mu(h) = \frac{1}{2}||\mathcal{A}h - b||_2^2 + \lambda\left(\frac{1}{2\mu}||h||_2^2 - \frac{\mu}{2}\left|\left|\frac{h}{\mu} - \Pi_Q\left(\frac{h}{\mu}\right)\right|\right|^2\right).$$

This function has a gradient of

$$\nabla f_\mu(h) = \mathcal{A}^\top(\mathcal{A}h - b) + \lambda\Pi_Q\left(\frac{h}{\mu}\right).$$

Thus, we can write a gradient descent algorithm based on Nesterov's smoothing to solve the image reconstruction problem:

Gradient descent algorithm for image reconstruction
INPUT: $h_0 \in \mathbb{R}^n, N \in \mathbb{N}, t_0, t_2, ..., t_N > 0, \mu > 0$.
for $k \in \{0, ..., N\}$
Set $h_{k+1} := h_k - t_k\left(\mathcal{A}^\top(\mathcal{A}h - b) + \lambda\Pi_Q\left(\frac{h}{\mu}\right)\right)$.
end

Although this algorithm serves as a possible method to reconstruct images, a faster algorithm such as Nesterov's accelerated gradient method can be used to solve similar convex problems.

3.3.4 The FISTA for image reconstruction

The FISTA is a common technique used to solve convex optimization problems. In the case of (3.1), it relies on splitting the objective function into two convex parts: $f = \phi + \psi$, where $\phi(h) := \frac{1}{2}||\mathcal{A}h - b||_2^2$ and $\psi(h) = \lambda||h||_1$.

Note that ϕ is continuously differentiable and its gradient is Lipschitz continuous with $L > 0$. Define the *shrinkage operator* ρ_L componentwise as

$$[\rho_L(y)]_i := \begin{cases} \text{sign}\left([z_L(y)]_i\right)\left(|[z_L(y)_i]| - \frac{\lambda}{L}\right), & |[z_L(y)]_i| > \frac{\lambda}{L} \\ 0, & |[z_L(y)]_i| \leq \frac{\lambda}{L} \end{cases}.$$

The algorithm is then as follows:

FISTA with backtracing for image reconstruction
INPUT: $L_0 > 0, \eta > 1, h_0 \in \mathbb{R}^n, N \in \mathbb{N}$
Set $y_1 := h_0$.
for $k \in \{1, ..., N\}$
  Set $L := L_{k-1}$.
  do
    Set $L := \eta L$.
  while $f(\rho_L(y_k)) > Q_L(\rho_L(y_k), h_k)$
  Set $L_k := L$.
  Set $h_k := \rho_{L_k}(y_k)$.
  Set $t_{k+1} := \frac{1+\sqrt{1+4t_k^2}}{2}$.
  Set $y_{k+1} := h_k + \frac{t_k - 1}{t_{k+1}}(h_k - h_{k-1})$.
end
return h_N.

This algorithm is useful because its convergence does not rely on setting many parameters, just an initial guess and $\eta > 1$. The first instructions within the for loop are the backtracing steps; this technique will be used in the boosted DCA. The FISTA was used to test the image reconstruction abilities of learned and DCT dictionaries; see [22] and the references therein.

4 Dictionary learning

In this section, methods of dictionary learning via the DCA and the boosted DCA will be detailed. Given a set of training data X, Each method is fundamentally the same in two ways:

- Optimization is done in two phases: first with respect to the sparse representation H and next with respect to the dictionary D. This ensures that the objective function will converge with respect to both variables, ie. the function is minimized at the final values of D and H.
- Both of these phases are completed using variations of the DCA.

The goal will therefore be to find two DC programs for each method: one for sparsity and one for updating the actual dictionary. In summary, the dictionary learning process proceeds as follows:

General dictionary learning
INPUT: $X, H_0, D_0, N \in \mathbb{N}$
for $k \in \{0, \ldots, N\}$
  Find H_{k+1} (depending on X, H_k and D_k) through DC programming.
  Find D_{k+1} (depending on X, H_{k+1} and D_k) through DC programming.
end

4.1 The general DCA

Consider a function of the form $f = g - h$, where g and h are convex.

The DCA consists of two steps repeated until a convergence criterion has been met: finding $y_{k+1} \in \partial h(x_k)$, then finding $x_{k+1} \in \partial g^*(y_{k+1})$. Here, g^* represents the Fenchel conjugate of g. In optimizing via the DCA, we usually use the following property of Frechet conjugates:

$$x \in \partial g^*(y) \iff y \in \partial g(x).$$

The general DCA is given below:

General DCA
INPUT: $x_0, N \in \mathbb{N}$
for $k = 1, \ldots, N$
    Find $y_{k+1} \in \partial h(x_k)$.
    Find $x_{k+1} \in \partial g^*(y_{k+1})$.
end

4.2 DCA dictionary learning

4.2.1 The sparse coding phase

In order to find a DC representation of f^λ designed to solve the sparse coding portion of dictionary learning, we use Nesterov's smoothing technique.

Recall from the image reconstruction section that if $p(x) := ||x||_1$, its Nesterov approximation is given by

$$p_\mu(x) = \frac{1}{2\mu} ||x||_2^2 - \frac{\mu}{2} d\left(\frac{x}{\mu}, Q\right)^2,$$

for some parameter $\mu > 0$ and Q is the unit box under the ℓ_∞ norm. Define $\widetilde{Q} := Q \times Q \times \ldots \times Q = Q^T \subset \mathbb{R}^{K \times T}$. This is the set of all $K \times T$ matrices such $||h_t||_\infty \leq 1$, for $1 \leq t \leq T$. It follows that

$$\sum_{t=1}^{T} d\left(\frac{h_t}{\mu}; Q\right)^2 = \sum_{t=1}^{T} \left|\left|\frac{h_t}{\mu} - \Pi_Q\left(\frac{h_t}{\mu}\right)\right|\right|_2^2 = \left|\left|\frac{H}{\mu} - \Pi_{\widetilde{Q}}\left(\frac{H}{\mu}\right)\right|\right|_F^2 = d\left(\frac{H}{\mu}; \widetilde{Q}\right)^2,$$

where the distance in the last equality is measured with the Froebenius norm.

We therefore can approximate $||H||_{11}$ by

$$||H||_{11} = \sum_{t=1}^{T} ||h_t||_1 \approx \sum_{t=1}^{T} p_\mu(h_t) = \sum_{t=1}^{T} \frac{1}{2\mu} ||h_t||_2^2 - \frac{\mu}{2} d\left(\frac{h_t}{\mu}, Q\right)^2 = \frac{1}{2\mu} ||H||_F^2 - \frac{\mu}{2} d\left(\frac{H}{\mu}; \widetilde{Q}\right)^2.$$

Therefore, we can approximate the function f^λ by

$$\begin{aligned} f^\lambda(D,H) \approx f_\mu^\lambda(D,H) &= \frac{\lambda}{2}\left|\left|DH-X\right|\right|_F^2 - \left|\left|H\right|\right|_{21} + \frac{1}{2\mu}\left|\left|H\right|\right|_F^2 - \frac{\mu}{2}d\left(\frac{H}{\mu};\widetilde{Q}\right)^2 \\ &= \frac{\lambda}{2}\left|\left|DH-X\right|\right|_F^2 - \left|\left|H\right|\right|_{21} + \frac{1}{2\mu}\left|\left|H\right|\right|_F^2 - \frac{\mu}{2}d\left(\frac{H}{\mu};\widetilde{Q}\right)^2 + \left(\frac{\gamma_1}{2}\left|\left|H\right|\right|_F^2 - \frac{\gamma_1}{2}\left|\left|H\right|\right|_F^2\right) \\ &= \left(\frac{1}{2\mu}+\frac{\gamma_1}{2}\right)\left|\left|H\right|\right|_F^2 - \left(\frac{\mu}{2}d\left(\frac{H}{\mu};\widetilde{Q}\right)^2 - \frac{\lambda}{2}\left|\left|DH-X\right|\right|_F^2 + \frac{\gamma_1}{2}\left|\left|H\right|\right|_F^2 + \left|\left|H\right|\right|_{21}\right). \end{aligned}$$

Each term is defined as

$$g_\mu(H) := \left(\frac{1}{2\mu}+\frac{\gamma_1}{2}\right)\left|\left|H\right|\right|_F^2 \text{ and } h_\mu^\lambda(D,H) := \frac{\mu}{2}d\left(\frac{H}{\mu};\widetilde{Q}\right)^2 - \frac{\lambda}{2}\left|\left|DH-X\right|\right|_F^2 + \frac{\gamma_1}{2}\left|\left|H\right|\right|_F^2 + \left|\left|H\right|\right|_{21}.$$

Under these definitions, $f_\mu^\lambda = g_\mu - h_\mu^\lambda$. The parameter $\gamma_1 > 0$ is chosen such that γ_1/λ is greater than the absolute value of the largest eigenvalue of $D^\top D$. If γ_1 is sufficiently large, h_μ^λ will be convex. Since g_μ is also convex, f_μ^λ is a smooth, DC approximation of f^λ.

In order to perform optimization via the DCA, we need to find subdifferentials of h_μ and g_μ^λ, which are functions of matrices. For a function $F : \mathbb{R}^{m\times n} \to \mathbb{R}, W \mapsto F(W)$, we define its subgradient ∂F columnwise by

$$[\partial F(W)]_j := \partial_{w_j} F(W),$$

where ∂_{w_j} denotes taking the vector subgradient with respect to the jth column of W. If the function is differentiable at W, we write $\partial F(W) = \{\nabla F(W)\}$.

In finding the gradient of the function h_μ, each term is differentiable everywhere except for the last (the $||H||_{21}$ term). It is non-differentiable whenever $||h_t||_2 = 0$ for some $t \in \{1, ..., T\}$.

Consider the function $a = ||\cdot||_2$. Then if $x \neq 0$, $\nabla a(x) = x/\left|\left|x\right|\right|_2$, so $\partial a(x) = \{x/\left|\left|x\right|\right|_2\}$. If $x = 0$, then $\partial a(0) = \mathbb{B}_2(0;1)$. Define the function ω by

$$\omega(x) = \begin{cases} 0, & x = 0 \\ \frac{x}{||x||_2} & x \neq 0. \end{cases}$$

Thus, $\omega(x) \in \partial a(x)$ for all x.

The matrix subgradient of $F = ||\cdot||_{21}$ is given columnwise by

$$[\partial F]_t = \partial_{h_t}\left|\left|H\right|\right|_{21} = \partial_{h_t}\sum_{j=1}^T a(h_j) = \sum_{j=1}^T \partial_{h_t} a(h_j) = \partial_{h_t} a(h_t) \ni \omega(h_t).$$

Define $\Omega : \mathbb{R}^{K\times T} \to \mathbb{R}$ columnwise by $[\Omega(H)]_t := \omega(h_t)$. Then $\Omega(H) \in \partial F(H)$ for all H. Differentiating the other terms of h_μ^λ yields

$$\nabla h_\mu^\lambda(H) = \frac{H}{\mu} - \Pi_{\widetilde{Q}}\left(\frac{H}{\mu}\right) - \lambda D^\top(DH-X) + \gamma_1 H + \Omega(H).$$

This is not technically the true gradient of h_μ^λ: ∇h_μ^λ always returns one element of ∂h_μ^λ. To complete the first step of the DCA, it suffices to set

$$Y := \frac{H}{\mu} - \Pi_{\widetilde{Q}}\left(\frac{H}{\mu}\right) - \lambda D^\top (DH - X) + \gamma_1 H + \Omega(H).$$

For matrix subgradients, the properties of the Fenchel conjugates still apply. Thus, $H \in \partial g_\mu^*(Y) \iff Y \in \partial g_\mu(H)$. Since g_μ is differentiable,

$$Y \in \partial g_\mu(H) \implies Y = \nabla g_\mu(H).$$

The gradient of g_μ is readily calculated to be

$$\nabla g_\mu(H) = \left(\frac{1}{\mu} + \gamma_1\right) H = \left(\frac{1 + \gamma_1 \mu}{\mu}\right) H.$$

Therefore, to satisfy the second step of the DCA, we set H such that $Y = \left(\frac{1+\gamma_1\mu}{\mu}\right) H$, or

$$H = \left(\frac{\mu}{1 + \gamma_1 \mu}\right) Y.$$

The DCA for sparse coding can be summarized as follows:

DCA for sparse coding
INPUT: $\lambda > 0, \gamma_1 > 0, X, D, H_0, N \in \mathbb{N}$
for $k = 1, \ldots, N$
  Set $Y_{k+1} := H_k/\mu - \Pi_{\widetilde{Q}}(H_k/\mu) - \lambda D^\top (DH_k - X) + \gamma_1 H_k + \Omega(H_k)$.
  Set $H_{k+1} := \left(\frac{\mu}{1+\gamma_1\mu}\right) Y_{k+1}$.
end
return H_{N+1}

4.2.2 The dictionary update phase

Again applying the same techniques, we can develop a DCA method for updating the dictionary after finding an adequate sparse representation. We apply similar techniques of differentiating functions with respect to matrices, except instead we take derivatives with respect to D.

First, define the constraint set for each dictionary atom by $C := \overline{\mathbb{B}}_2(0; 1)$. In order for D to be within the defined constraint set, its atoms must satisfy $d_j \in C$ for $1 \leq j \leq K$. In order to develop a constraint for the entire dictionary D, simply define
$\widetilde{C} := C \times C \times \ldots \times C = C^K \subset \mathbb{R}^{n_1 n_2 \times K}$. This is the set of all $n_1 n_2 \times K$ matrices such that each column has a norm less than or equal to 1.

The original problem can be restated as

$$\text{minimize } f^{\lambda}(D, H),$$

subject to $D \in \widetilde{C}$. Consider the indicator function for $\widetilde{C}$, $I_{\widetilde{C}}$. For any matrix D, $I_{\widetilde{C}}(D) = 0$ if $D \in \widetilde{C}$ and $I_{\widetilde{C}}(D) = \infty$ otherwise. Therefore, if D does not satisfy the constraint, $f(D, H) + I_{\widetilde{C}}(D)$ could never be at a minimum value. In fact, $f(D, H) + I_{\widetilde{C}}(D)$ could only be minimized if $D \in \widetilde{C}$. Therefore, we can recast the original problem as

$$\text{minimize } f^{\lambda}(D, H) + I_{\widetilde{C}}(D).$$

Explicitly writing each term,

$$f^{\lambda}(D, H) + I_{\widetilde{C}} = \frac{\lambda}{2} ||DH - X||_F^2 + ||H||_{11} - ||H||_{21} + I_{\widetilde{C}}(D).$$

During the dictionary update step, H is assumed to be fixed. Therefore, the $||H||_{11}$ and $||H||_{21}$ terms are not important in updating the dictionary. Excluding these terms, the sparsity tradeoff term λ is also unimportant. Define $\widetilde{f}(D) := \frac{1}{2} ||DH - X||_F^2 + I_{\widetilde{C}}(D)$. Then the dictionary update step is designed to solve the problem

$$\text{minimize } \widetilde{f}(D).$$

Note that for any choice of γ_2,

$$\begin{aligned}
\widetilde{f}(D) &= \frac{1}{2} ||DH - X||_F^2 + I_{\widetilde{C}}(D) + \left(\frac{\gamma_2}{2} ||D||_F^2 - \frac{\gamma_2}{2} ||D||_F^2 \right) \\
&= \left(I_{\widetilde{C}}(D) + \frac{\gamma_2}{2} ||D||_F^2 \right) - \left(\frac{\gamma_2}{2} ||D||_F^2 - \frac{1}{2} ||DH - X||_F^2 \right) \\
&= \widetilde{g}(D) - \widetilde{h}(D),
\end{aligned}$$

where $\widetilde{g}(D) := I_{\widetilde{C}}(D) + \frac{\gamma_2}{2} ||D||_F^2$ and $\widetilde{h}(D) := \frac{\gamma_2}{2} ||D||_F^2 - \frac{1}{2} ||DH - X||_F^2$. The parameter γ_2 is chosen to be larger than the absolute value of the greatest eigenvalue of $H^\top H$, which will ensure that $\widetilde{h}$ is convex. Thus, $\widetilde{f}$ is a DC function.

In order to accomplish the first step of our DCA, we note that $\widetilde{h}$ is differentiable and find that

$$\nabla \widetilde{h}(D) = \gamma_2 D - (DH - X)H^\top.$$

Therefore, the first step is accomplished by simply setting

$$Y = \nabla \widetilde{h}(D) = \gamma_2 D - (DH - X)H^\top.$$

The second step of the DCA is accomplished via two smaller processes executed in order: by first finding D from Y assuming no constraints apply and next projecting onto the constraint set $\widetilde{C}$. We note that since $||\cdot||_F^2$ is differentiable,

$$Y = \nabla \left(\frac{\gamma_2}{2} ||D||_F^2 \right) = \gamma_2 D \implies D = \frac{Y}{\gamma_2}.$$

Projecting this answer onto the constraint set, we satisfy the second step of the DCA by setting

$$D = \Pi_{\widetilde{C}}\left(\frac{Y}{\gamma_2}\right).$$

In summary, these two steps can be combined in the algorithm below, which is the DCA for the dictionary update step:

DCA for dictionary update
INPUT: $\gamma_2 > 0, X, D_0, H, N \in \mathbb{N}$
for $k \in \{0, ..., N\}$
    Set $Y_{k+1} := \gamma_2 D_k - (D_k H - X)H^\top$.
    Set $D_{k+1} := \Pi_{\widetilde{C}}(Y_{k+1}/\gamma_2)$.
end
return D_{N+1}.

4.2.3 The dictionary learning algorithm

Combining the sparse coding and dictionary update phases, the resulting dictionary learning algorithm is listed below:

DCA for dictionary learning
INPUT: $\mu, \gamma_1, \gamma_2 > 0, X, D_0, H_0, N, N_1, N_2 \in \mathbb{N}$
for $i \in \{0, ..., N\}$
    for $k \in \{0, ..., N_1\}$
        Set $Y_{k+1} := H_k/\mu - \Pi_{\widetilde{Q}}(H_k/\mu) - \lambda D_i^\top (D_i H_k - X) + \gamma_1 H_k + \Omega(H_k)$.
        Set $H_{k+1} := \left(\frac{\mu}{1+\gamma_1\mu}\right) Y_{k+1}$.
    end
    Set $H_{i+1} := H_{N_1+1}$.
    for $k \in \{N_1 + 1, ..., N_1 + N_2 + 1\}$
        Set $Y_{k+1} := \gamma_2 D_k - (D_k H_{i+1} - X)H_{i+1}^\top$.
        Set $D_{k+1} := \Pi_{\widetilde{C}}(Y_{k+1}/\gamma_2)$.
    end
    Set $D_{i+1} := D_{N_2+1}$.
end
return D_{N+1}

In applying this technique, it usually makes sense to start with μ large and gradually decrease its value. This ensures stability early in the process and gradually moves the dictionary learning towards a more accurate answer. To accomplish this, an initial value μ_i and a final value μ_f are input.

In addition, we can replace γ_1 and γ_2 by $\gamma := \max\{\gamma_1, \gamma_2\}$, since we only require that γ_1 and γ_2 be sufficiently large. Applying these tweaks, the resulting algorithm is listed below.

DCA for dictionary learning

INPUT: $0 < \mu_N < \mu_0, \gamma > 0, X, D_0, H_0, N, N_1, N_2 \in \mathbb{N}$

Set $\sigma := (\mu_N/\mu_0)^N$.

for $i \in \{0, ..., N\}$

  for $k \in \{0, ..., N_1\}$

    Set $Y_{k+1} := H_k/\mu_i - \Pi_{\widetilde{Q}}(H_k/\mu_i) - \lambda D_i^\top (D_i H_k - X) + \gamma H_k + \Omega(H_k)$.

    Set $H_{k+1} := \left(\frac{\mu_i}{1+\gamma\mu_i}\right) Y_{k+1}$.

  end

  Set $H_{i+1} := H_{N_1+1}$.

  for $k \in \{N_1 + 1, ..., N_1 + N_2 + 1\}$

    Set $Y_{k+1} := \gamma D_k - (D_k H_{i+1} - X) H_{i+1}^\top$.

    Set $D_{k+1} := \Pi_{\widetilde{C}}(Y_{k+1}/\gamma)$.

  end

  Set $D_{i+1} := D_{N_2+1}$.

  Set $\mu_{i+1} := \mu_i \sigma$.

end

return D_{N+1}

4.3 The boosted DCA

Although the algorithm above accurately produces image dictionaries, improvements could be made by applying the boosting technique for the DCA. This involves introducing a line search to the overall process, which backtraces to improve the convergence rate. Below is the general boosted DCA (see [25]).

Boosted DCA

INPUT: x_0, $N \in \mathbb{N}$,

$\alpha > 0, \bar{t} > 0, 0 < \beta < 1$.

for $k = 1, \ldots, N$

  Find $z_{k+1} \in \partial h(x_k)$.

  Find $y_{k+1} \in \partial g^*(z_{k+1})$.

  Set $d_{k+1} := y_{k+1} - x_k$.

  if $d_{k+1} = 0$: **break.**

  Set $t := \bar{t}$.

  while $f(y_{k+1} + t d_{k+1}) > f(y_{k+1}) - \alpha t \, ||d_{k+1}||_2^2$

    Set $t := \beta t$.

  end

  Set $x_{k+1} := y_{k+1} + t d_{k+1}$.

end for

return x_{N+1}.

The first two steps of the boosted DCA follow the conventional DCA. A backtracing section is applied afterwards in order to guide the solution towards a minimum again. In our case, we use the Nesterov smoothed approximations for the first two steps and the exact value of f for the backtracing steps.

For the sparse coding phase, the boosted DCA is as follows:

Boosted DCA for sparse coding
INPUT: $\gamma_1, \mu > 0$, D, H_0, X,
$\alpha > 0, \bar{t} > 0, 0 < \beta < 1$.
for $kk = 1, \ldots, N$
  Set $Z_{k+1} := H_k/\mu - \Pi_{\widetilde{Q}}(H_k/\mu) - \lambda D^\top(DH_k - X) + \gamma_1 H_k + \Omega(H_k)$.
  Set $Y_{k+1} := \left(\frac{\mu}{1+\gamma_1\mu}\right) Z_{k+1}$.
  Set $d_{k+1} := Y_{k+1} - H_k$.
  if $d_{k+1} = 0_{K\times T}$: **break**
  Set $t := \bar{t}$.
  while $f^\lambda(D, Y_{k+1} + td_{k+1}) > f^\lambda(D, Y_{k+1}) - \alpha\,||d_{k+1}||_F^2$
    Set $t := \beta t$.
  end
  Set $H_{k+1} := Y_{k+1} + td_{k+1}$.
end for
return H_{N+1}

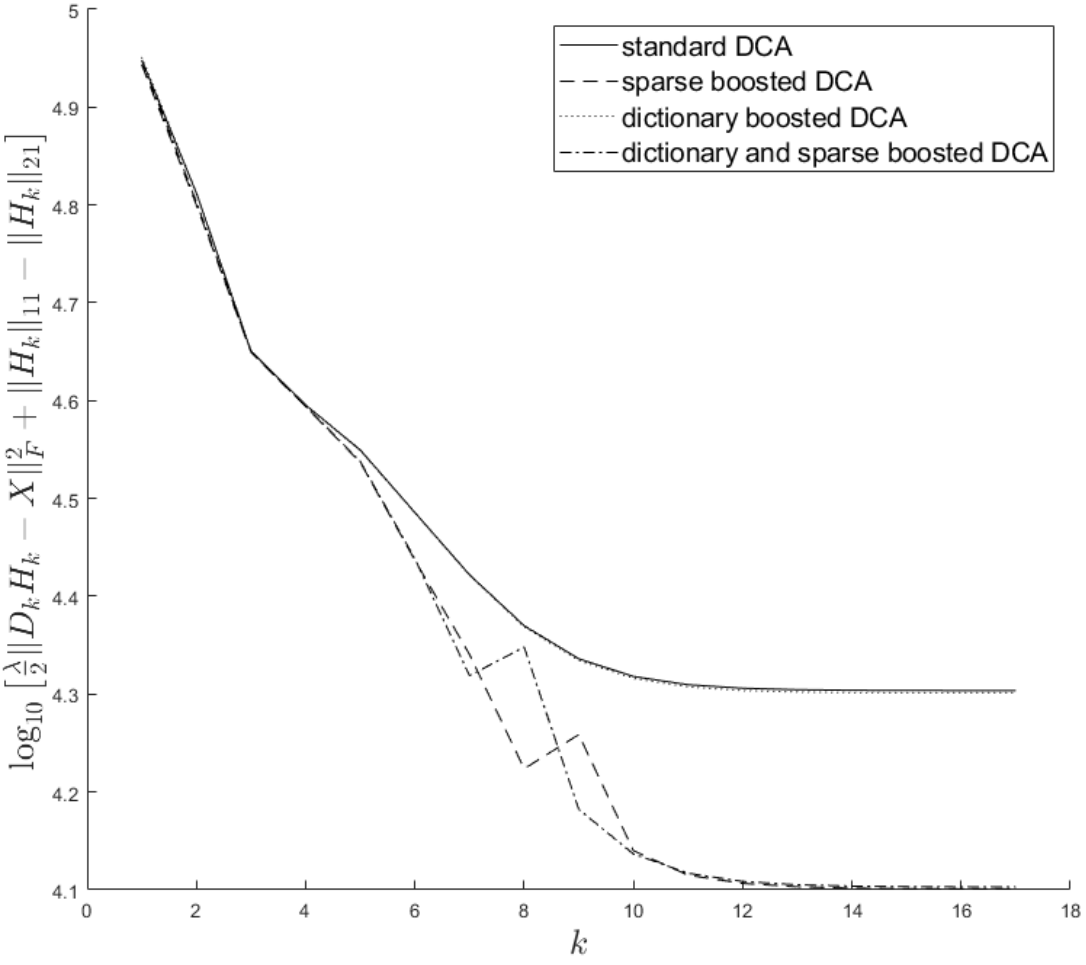
In the case of updating the dictionary, no Nesterov smoothing was used. Therefore, the function $\widetilde{f}$ can be used with no penalty to the accuracy of the algorithm. Below is the boosted DCA for the dictionary update.

Boosted DCA for dictionary update
INPUT: γ_2, D_0, H, X,
$\alpha > 0, \bar{t} > 0, 0 < \beta < 1$.
for $kk = 1, \ldots, N$
  Set $Z_{k+1} := \gamma_2 D_k - (D_{k+1}H - X)H^\top$.
  Set $Y_{k+1} := \Pi_{\widetilde{C}}(Z/\gamma_2)$.
  Set $d_{k+1} := Y_{k+1} - D_k$.
  if $d_{k+1} = 0_{n_1 n_2 \times K}$: **break**
  Set $t := \bar{t}$.
  while $\widetilde{f}(Y_{k+1} + td_{k+1}) > \widetilde{f}(Y_{k+1}) - \alpha t\,||d_{k+1}||_F^2$
    Set $t := \beta t$.
  end
  Set $D_{k+1} := Y_{k+1} + td_{k+1}$.
end for
return D_{N+1}

In making a boosted dictionary learning algorithm, one can choose to use the boosted sparse coding phase, the boosted dictionary update phase, or both at once. These combinations were tested and compared with the standard version of the DCA.

5 Results and discussion

For each method tested, $\lambda = 1.5$, $\gamma = 5000\lambda = 7500, \mu_i = 1, \mu_f = 10^{-6}$. The number of atoms was $K = 256$. A maximum of 5000 iterations were used overall for each method (if the dictionary converged within 10^{-4}, the program exited early). The convergence graphs for each technique are shown below, with the y-axis in logarithmic scale.



The dictionaries were each tested via the FISTA for image reconstruction, with $\lambda = 0.01$, $\eta = 1.001$. The $N_1 \times N_2 = 512 \times 512$ black and white Lena image was used for the test, with a sample rate of 50%. Let $\widetilde{M}$ be the original image and let $\widetilde{M}'$ be the restored image, in matrix form. Define the relative error RE and the peak-signal-to-noise ratio $PSNR$ by

$$RE = \frac{\left|\left|\widetilde{M}' - \widetilde{M}\right|\right|_F}{\left|\left|\widetilde{M}\right|\right|_F}; \qquad PSNR = 20 \log_{10} \frac{\sqrt{N_1 N_2}}{\left|\left|\widetilde{M}' - \widetilde{M}\right|\right|_F}.$$

For the corrupted image, the RE was 70.7% and the PSNR was 8.47.

A reconstructed image is "good" if it has a low RE and a high PSNR. The table below summarizes the results for these dictionaries.

Dictionary learning results			
Dictionary	No. iterations to converge	RE(%)	PSNR
DCT	Not applicable	6.18	29.6
DCA	17	6.18	29.6
DCA, sparse boost	18	6.16	29.7
DCA, dict. boost	17	6.18	29.6
DCA, both boost	21	6.11	29.7

Notice that the DCA method performed better overall than the DCT method. The DCT does have one primary advantage though: it does not have to be learned from data.

References

[1] M. Aharon, M. Elad, A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process.* **54** (2006), 4311–4322.

[2] L.T.H. An, P.D. Tao. DC programming and DCA: thirty years of developments. *Mathematical Programming. Special Issue: DC Programming - Theory, Algorithms and Applications.* **169(1)** (2018), 5–64.

[3] A. Beck, M. Teboulle. Smoothing and first order methods: a unified framework. *SIAM J. Optim.* **22** (2012), 557–580.

[4] A. Beck, M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2** (2009), 183–202.

[5] F.H. Clarke. Nonsmooth Analysis and Optimization. John Wiley & Sons, Inc., New York, 1983.

[6] J.R. Giles. A Survey of Clarke's Subdifferential and the Differentiability of Locally Lipschitz Functions. In: Progress in Optimization. Applied Optimization **30**. Springer, Boston, MA.

[7] P. Hartman. On functions representable as a difference of convex functions. *Pacific J. Math.* **9** (1959), 707–713.

[8] J.B. Hiriart-Urruty. Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. Lecture Note in *Economics and Math. Systems.* **256** (1985), 37–70.

[9] J. Mairal, F. Bach, J. Ponce, G. Sapiro. Online dictionary learning for sparse coding. Proc. *26th Int'l Conf. Machine Learning.* Montreal, Canada, 2009.

[10] D. Martin, C. Fowlkes, D. Tal, J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Proc. *8th Int'l Conf. Computer Vision.* **2** (2001), 416–423.

[11] B.S. Mordukhovich. Variational Analysis and Generalized Differentiation, I: Basic Theory, II: Applications. Grundlehren Series (Fundamental Principles of Mathematical Sciences), **330** and **331**, Springer, Berlin, 2006.

[12] B.S. Mordukhovich, M.N. Nam. An Easy Path to Convex Analysis and Applications. Morgan & Claypool, 2014.

[13] B.S. Mordukhovich, N.M. Nam, and N.D. Yen, Fréchet subdifferential calculus and optimality conditions in nondifferentiable programming. *Optimization.* **55** (2006), 685–708.

[14] N.M. Nam, L.T.H. An, N.T. An, D. Giles, Smoothing techniques and difference of convex functions algorithms for image reconstructions, *Optimization* (2019), accepted.

[15] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program., Ser. A.* **103** (2005), 127–152.

[16] T. Pham Dinh, H.A. Le Thi. Convex analysis approach to D.C. programming: Theory, algorithms and applications. *Acta Math. Vietnam.* **22** (1997), 289–355.

[17] T. Pham Dinh, H.A. Le Thi. A d.c. optimization algorithm for solving the trust-region subproblem. *SIAM J. Optim.* **8** (1998), 476–505.

[18] L. Vandenberghe. Optimization methods for large-scale systems, *EE236C lecture notes*, UCLA.

[19] J. Xin, S. Osher, Y. Lou. Computational aspects of L1-L2 minimization for compressive sensing. *Advances in Intelligent Systems and Computing.* **359** (2015), 169–180.

[20] P. Yin, Y. Lou, Q. He, J. Xin. Minimization of L1-L2 for compressed sensing. *SIAM J. of Sci. Comput.* **37** (2015), A536–A563.

[21] Y. Xu, W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sci.* **6** (2013), 1758–1789.

[22] Y. Xu, W. Yin. A fast patch dictionary method for whole image recovery. *Inverse Problems and Imaging.* **10** (2016), 563–583.

[23] C. Zălinescu. Convex Analysis in General Vector Spaces, World Scientific, Singapore, 2002.

[24] Y. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. akad. nauk Sssr* **269** (1983), 534–547

[25] A. Artacho, F. Javier. The boosted difference of convex functions algorithm for nonsmooth functions. *SIAM Journal on Optimization.* **30(1)** (2020), 980–1006.