Low Mutual and Average Coherence Dictionary Learning Using Convex Approximation

To cite this version:

HAL Id: hal-02560161
https://inria.hal.science/hal-02560161
Submitted on 4 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT
In dictionary learning, a desirable property for the dictionary is to be of low mutual and average coherences. Mutual coherence is defined as the maximum absolute correlation between distinct atoms of the dictionary, whereas the average coherence is a measure of the average correlations. In this paper, we consider a dictionary learning problem regularized with the average coherence and constrained by an upper-bound on the mutual coherence of the dictionary. Our main contribution is then to propose an algorithm for solving the resulting problem based on convexly approximating the cost function over the dictionary. Experimental results demonstrate that the proposed approach has higher convergence rate and lower representation error (with a fixed sparsity parameter) than other methods, while yielding similar mutual and average coherence values.

Index Terms—Compressed sensing, sparse coding, mutual coherence, average coherence, dictionary learning

1. INTRODUCTION

1.1. Dictionary learning
Dictionary learning (DL) has been extensively utilized in a wide range of machine learning and signal processing applications, including image/signal enhancement and reconstruction [1, 2], and pattern recognition and classification [3]. A lot of algorithms have been proposed for this problem. To formally define it, given a training dataset $Y = [y_1, y_2, ..., y_l]$, $y_i \in \mathbb{R}^m$, a dictionary $D = [d_1, d_2, ..., d_n]$, $d_i \in \mathbb{R}^m$, is learned in such a way that it provides sparse coefficients for $y_i$'s. That is, the representations $X = [x_1, x_2, ..., x_l]$, $x_i \in \mathbb{R}^n$ are sufficiently sparse. To achieve this, the DL problem is usually formulated as follows [4]

$$\min_{D, X} \|Y - DX\|^2_F$$

(1)

in which, $\mathcal{D}$ and $\mathcal{X}$ are defined as $\mathcal{D} = \{D : \forall i, \|d_i\|_2 \leq \tau\}$ and $\mathcal{X} = \{X : \forall i, \|x_i\|_1 \leq \tau\}$, where, $\|\cdot\|_1$ denotes $\ell_1$ norm. To solve (1), many dictionary learning algorithms have been introduced [2, 4, 5, 6, 7], which are mainly based on alternating minimization on $D$ and $X$. Some methods impose additional constraints on the dictionary $D$ which can improve the performance [1, 8, 9]. Two important properties are reviewed in the next section.

1.2. Mutual and average coherences
One of the important properties of a dictionary is the maximum correlation between the columns of the dictionary which is called mutual coherence and denoted by $\mu(D)$ [10]. Another important property of a dictionary is the average correlation of dictionary columns, which is called average coherence and denoted by $\mu_{avg}(D)$. For a dictionary $D$, these two parameters are respectively defined as:

$$\mu(D) = \max_{i \neq j} \frac{|d_i^T d_j|}{\|d_i\|_2 \|d_j\|_2},$$

$$\mu_{avg}(D) = \sqrt{\frac{D^T D - I}{n(n-1)}}.$$  (2)

In dictionary learning, it is usually desired that the mutual coherence of the learned dictionary is small. This is because of two main reasons: On the one hand, it has been shown in [11] that a dictionary with low mutual coherence has well-conditioned sub-matrices. On the other hand, a signal with a sparse representation $x$ with sparsity parameter $s$, i.e. with $s$ nonzero coefficient $s$, can be recovered from $y = DX$ through $\ell_1$ minimization when [12]:

$$s \leq \frac{1}{2} (1 + \frac{1}{\mu(D)}).$$  (3)

According to (3), dictionaries with low mutual coherence are better for high $s$. However, the mutual coherence is lower bounded [13], and it can be shown that:

$$D \in \mathbb{R}^{m \times n} \Rightarrow \mu_{welch} \leq \mu(D) \leq 1,$$  (4)

where the $\mu_{welch}$ is the Welch bound [13], defined as:

$$\mu_{welch} = \sqrt{\frac{n - m}{m(n-1)}}.$$  

Furthermore, dictionaries with low average coherence are favorable in compressed sensing applications [14].

During recent years, many dictionary learning algorithms have been proposed trying to reduce mutual coherence [8, 9, 15]. A recent approach, called Gradient-based ISDL [16] (GSD), has been proposed in [17], which minimizes the following cost function:

$$\min_{D \in \mathcal{D}, X \in \mathcal{X}} \|Y - DX\|^2_F + \frac{\lambda}{2} \|D^T D - H\|^2_F,$$  (5)
where
\[ \mathcal{H} \triangleq \left\{ H \in \mathbb{R}^{n \times n} : H = H^T, h_{ii} = 1, \forall i \max_{i \neq j} |h_{ij}| \leq \mu_0 \right\}, \]
in which \( \mu_0 \geq \mu_{\text{welch}} \). To solve the above problem, an alternative minimization approach has been used in [17], while this problem does not yield closed form solution for updating the dictionary. In other words, the gradient of the cost function, \( F(D) \), over \( D \) is computed as follows:
\[ F(D) \triangleq \| Y - DX \|_F^2 + \frac{\lambda}{2} \| D^T D - H \|_F^2 \quad (6) \]
\[ \nabla_D F(D) = 2(DX - Y)X^T + 2\lambda D(D^T D - H). \quad (7) \]
Then solving \( F(D) \) does not result to a closed-form solution, because (7) is non-linear over \( D \). For this reason, [17] uses gradient descent to update the dictionary, that is,
\[ D_{k+1} = D_k - \alpha \nabla_D F(D_k). \quad (8) \]
Then, \( H \) is updated using the following formula, in which \( \mu \) denotes the iteration number:
\[ h^\mu_{ij} = \begin{cases} \frac{1}{\mu} & i \neq j, |\mu| \leq \mu_0 \\ \text{sgn}(\mu)|\mu| & i \neq j, |\mu| \geq \mu_0 \\ 1 & i = j \end{cases} \quad (9) \]
in which, \( \mu = u^k_{ij} \), the \((i, j)\) entry of \( U_k = D_k^T D_k \).

In some older papers, e.g. [8, 9, 15], a cost function similar to (6) has been used by having identity matrix \( I \) instead of \( H \). And in [18], a combination of these two cost function have been used. In all of these papers, the cost function is non-linear over \( D \), so they all use an update equation of the form (8).

In this paper, we propose a new approach that approximates these non-convex cost functions over the dictionary by a convex function. This leads to a closed-form solution for the dictionary. As our simulations will confirm, the new algorithm results in improved performance in dictionary recovery.

The rest of the paper is organized as follows. Section 2 presents the main idea of our algorithm and related discussions. Then, the new algorithm is experimentally evaluated in Section 3.

2. THE PROPOSED ALGORITHM

To develop our proposed algorithm, consider the pair \((D_a, X_a)\), which is assumed to be known. Then, following [4, 19], we can write
\[ \begin{align*}
D &= D_a + D - D_a \\
X &= X_a + X - X_a
\end{align*} \quad (10) \]
\[ \begin{align*}
DX &= (D_a + D - D_a)(X_a + X - X_a) = D_a X + DX_a \\
& - D_a X_a + (D - D_a)(X - X_a)
\end{align*} \quad (11) \]
\[ \begin{align*}
D^T D &= (D_a + D - D_a)^T (D_a + D - D_a) = D_a^T D \\
& + D^T D_a - D_a^T D_a + (D - D_a)^T (D - D_a). \quad (12)
\end{align*} \]
Assuming that \((D - D_a)(X - X_a)\|_F \) and \((D - D_a)^T (D - D_a)\|_F \) are small, we can write
\[ \begin{align*}
DX &\approx D_a X + DX_a - D_a X_a \\
D^T D &\approx D_a^T D + D^T D_a - D_a^T D_a, \quad (13) \\
\end{align*} \]
from which we can propose the following approximate problem which is convex over \( D \):
\[ \begin{align*}
\min_{D \in \mathbb{R}^{n \times X}, k} & \| Y - DX \|_F^2 + \frac{\lambda}{2} \| D^T D - H \|_F^2 \\
\min_{D \in \mathbb{R}^{n \times X}, k} & \| Y + D_a X_a - D_a X - DX_a \|_F^2 \\
& + \frac{\lambda}{2} \| D_a^T D + D^T D_a - D_a^T D_a - H \|_F^2. \quad (16)
\end{align*} \]
To solve the above new problem, we use an alternating minimization approach, by optimizing the cost over one variable while fixing the other one. This procedure is summarized as follows.

2.1. Updating sparse coefficients (first term of (16))

In this stage, suppose \( D_a = D_{k-1}, D = D_k, X_a = X_k, Z_k = Y - (D_k - D_{k-1}) X_k \), then we can write:
\[ X_{k+1} = \min_{X \in \mathbb{C}^n} \| Z_k - D_{k-1} X \|_F^2. \quad (17) \]
To solve (17), we note that it is a sparse coding problem, with a lot of solvers existing in the literature [20].

2.2. Updating the dictionary (the two terms of (16))

In this stage, we assume \( X = X_a = X_{k+1} \) and \( D_a = D_k \). Then, the cost function for updating the dictionary of iteration \( k + 1 \) would be as follows:
\[ G(D) = \| Y - DX_{k+1} \|_F^2 + \frac{\lambda}{2} \| D_k^T D + D^T D_k - D_k^T H_k \|_F^2. \]
\[ \nabla_D G(D) = (DX_{k+1} - Y)X_{k+1} + \lambda D_k(D_k^T D + D^T D_k - D_k^T D_k - H_k) = DX_{k+1}X_{k+1} + \lambda D_k D_k^T D + \lambda D_k D^T D_k \\
- YX_{k+1}^T - \lambda D_k (D_k^T D_k + H_k). \]
To minimize \( G(D), \nabla_D G(D) \) is set to zero. By defining the auxiliary variables
\[ \begin{align*}
W_k &= (X_{k+1})^T X_{k+1} \\
A_k &= H_k + D_k^T D_k \\
C_k &= YX_{k+1} + \lambda D_k A_k
\end{align*} \quad (18) \]
the following equation is obtained to be solved in \( D \):
\[ DW_k + \lambda D_k D^T D_k + \lambda D_k D_k^T D = C_k. \quad (19) \]
By using the substitutions
\[ \begin{align*}
M_1 &= DW_k + \lambda D_k D^T D_k \\
M_2 &= \lambda D_k D_k^T D_k \\
M_1 + M_2 &= C_k \rightarrow \text{vec}(M_1) + \text{vec}(M_2) = \text{vec}(C_k)
\end{align*} \quad (20) \]
and using [21]:
\[
\sum_n A_n X B_n = R \rightarrow (\sum_n B_n^T \otimes A_n) \text{vec}(X) = \text{vec}(R),
\]
where \(\otimes\) denotes Kronecker product, one obtains
\[
\left\{ \begin{array}{l}
\text{vec}(M_1) = (W_k^T \otimes I_m + I_n \otimes (\lambda D_k^T D_k)) \text{vec}(D) \\
\text{vec}(M_2) = \lambda(D_k^T D_k) \text{vec}(D^T)
\end{array} \right.
\]
\[
(W_k^T \otimes I_m + I_n \otimes (\lambda D_k^T D_k)) \text{vec}(D) + \lambda(D_k^T D_k) \text{vec}(D^T) = \text{vec}(C). 
\]

To solve the above equation with respect to \(D_k\), we first determine a matrix \(B_k\) such that \((D_k^T \otimes D_k) \text{vec}(D^T)\) is equal to \(B_k \text{vec}(D)\). It is not difficult to see that such a \(B_k\) is obtained as:
\[
\left\{ \begin{array}{l}
Q_k \triangleq (D_k^T \otimes D_k), D_k \in \mathbb{R}^{m \times n}, 1 \leq i \leq n, 1 \leq j \leq m \\
B_k(:, ((i - 1)m + j)) \triangleq Q_k(:, (i + (j - 1)n))
\end{array} \right.
\]

So
\[
(W_k^T \otimes I_m + I_n \otimes (\lambda D_k^T D_k) + \lambda B_k) \text{vec}(D_{k+1}) = \text{vec}(C_k)
\Rightarrow \text{vec}(D_{k+1}) = (W_k^T \otimes I_m + I_n \otimes (\lambda D_k^T D_k) + \lambda B_k)^{-1} \text{vec}(C_k),
\]
which determines \(D_{k+1}\) in closed-form.

### 2.3. Updating \(H\):

It is updated by (9).

Note that our approach can be used on many dictionary learning algorithms to convexify the cost function. As an example, we apply it here on GSD [17] and RAMC [18]. The two new obtained methods are called Convex-GSD and Convex-RAMC.

The final algorithm (Convex-RAMC) is summarized in Algorithms 1 and Convex-GSD algorithm is achieved when \(\beta_1 = 0\) in Algorithm 1.

**Algorithm 1** The proposed algorithm (Convex-RAMC)

**Input:** \(Y, D^0, s\) (sparsity parameter)

**Initialization:** Set initial dictionary \(D^1 = D^0\).

for \(k = 1\) to MaxIteration do

- **Sparse approximation:** \(X_{k+1} = \text{OMP}(Z_k, D_{k-1}, s)\).

- **Dictionary update:** Dictionary is updated by equations (22).

- Normalize the columns of \(D_{k+1}\).

- Update \(H_{k+1}\) using (9) and replace \(H_{k+1}\) with \(\beta_1 I + \beta_2 H_{k+1}\) in (18), in which \(0 \leq \beta_1 \leq 1, 0 \leq \beta_2 \leq 1\) and \(\beta_1 + \beta_2 = 1\).

end for

### 3. SIMULATION RESULTS

In this section, we experimentally evaluate Convex-GSD and Convex-RAMC, and compare them with GSD [17], RAMC [18] and MOD [6] for recovering a known dictionary. Our simulations were performed in MATLAB R2017b environment on a system with 4.00 GHz 17 CPU and 16 GB RAM, under Microsoft Windows 10 operating system. As a rough measure of complexity, we will mention the run times of the algorithms. The performance measures are root mean square error (RMSE) defined as
\[
\varepsilon_k = \frac{\|Y - D_k X_k\|_2}{\sqrt{ml}} \quad [18], \%
\]
percentage of atom recovery, mutual coherence and average coherence (2).

Assuming that \(D_k\) is the true dictionary and \(D\) is the recovered dictionary, we say that the \(i\)th atom of the dictionary \(D\) is successfully recovered if
\[
\min_{i \neq j} (1 - |D(:,i)^T D_i(:,j)|) < 0.01.
\]

For OMP, we used the available MATLAB code at http://www.cs.technion.ac.il/~ronrubin/software.html. We generated a Gaussian random matrix \(D_i \in \mathbb{R}^{20 \times 50}\) with zero mean and unit variance. Then 2500 training data \(Y_{1:2500}^i\) were generated by random linear combinations of dictionary atoms. According to the size of the dictionary, the Welch bound (4) is computed as \(\mu_{\text{welch}} = 0.1749\) and we chose \(\mu_0 = \mu_{\text{welch}}\). In our simulation, we assume \(\beta_1 = 0.2, \beta_2 = 0.8, \text{SNR} = 30 \text{dB}\) and \(s = 7\) (sparsity parameter).

In all simulations, the sparsity parameter \(s\) is constant while the hyper-parameter \(\lambda\) (balancing the two terms of the cost \(\ell(D)\)) has two values 5 and 10. We performed 2000 iterations between the sparse coding and dictionary updating.

The dictionary was initialized by randomly choosing different signals from the training set followed by a normalization. We repeated all simulations 400 times and the averaged results are reported here.

Run times of algorithms are also compared as a rough measure of computational complexity. The average running times and iterations number of the algorithms for achieving a percentage of recovery equal to 80 are shown in Table 1.

Figures 1 to 8 are the results of the simulation of our algorithms and its comparisons with other mentioned algorithms. According to all the figures and table, our methods have higher convergence rate and lower RMSE than the other algorithms while mutual and average coherence of our methods are similar to those achieved by GSD and RAMC. The overall running time to converge of our methods is lower than the other methods. According to Fig 3 and 4, for \(\lambda = 5\) mutual and average coherences are similar for all the methods, while our methods have higher convergence rate than the other methods. In Figs. 7 and 8, mutual and average coherence of GSD are a little bit lower than Convex-GSD but the final percentage of recovery of Convex-GSD is 15 percent higher than GSD (see Fig. 5).

### 4. CONCLUSION

In this paper, we proposed a new approach to convexify the cost function of dictionary learning problem with low mutual and average coherence. According to our simulations on synthetic dictionary recovery, our approach increases the convergence rate and decreases RMSE, while mutual and average coherence of our algorithms are reduced well.
Fig. 1: Evaluation of percentage of recovery with assumptions: $\lambda = 5$.

Fig. 2: Evaluation of RMSE with assumptions: $\lambda = 5$. The graphs of Convex-GSD and Convex-RAMC are almost superimposed.

Fig. 3: Evaluation of average coherence with assumptions: $\lambda = 5$.

Fig. 4: Evaluation of mutual coherence with assumptions: $\lambda = 5$.

Fig. 5: Evaluation of percentage of recovery with assumptions: $\lambda = 10$.

Fig. 6: Evaluation of RMSE with assumptions: $\lambda = 10$. The graphs of Convex-GSD and Convex-RAMC are almost superimposed.

Fig. 7: Evaluation of average coherence with assumptions: $\lambda = 10$.

Fig. 8: Evaluation of mutual coherence with assumptions: $\lambda = 10$.

Table 1: Number of iterations and average running time (in seconds) for achieving percentage of recovery $= 80$. Average running times are reported in parentheses. In this table and all figures, $s = 7$ and SNR $= 30$dB are supposed.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\lambda = 5$</th>
<th>$\lambda = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex-GSD</td>
<td>91 (8.3s)</td>
<td>123 (10.1s)</td>
</tr>
<tr>
<td>Convex-RAMC</td>
<td>83 (7.5s)</td>
<td>102 (10.1s)</td>
</tr>
<tr>
<td>GSD</td>
<td>697 (18.6s)</td>
<td>1913 (59.3s)</td>
</tr>
<tr>
<td>RAMC</td>
<td>643 (17.2s)</td>
<td>626 (19.4s)</td>
</tr>
<tr>
<td>MOD</td>
<td>452 (9.4s)</td>
<td>449 (10.5s)</td>
</tr>
</tbody>
</table>
5. REFERENCES


