2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing

Discussions on Implementing Iterative Hard Thresholding Algorithm

Feng-Cheng Chang
Dept. of Innovative Information and Technology
Tamkang University, TAIWAN
Email: 135170@mail.tku.edu.tw

Hsiang-Cheh Huang
Department of Electrical Engineering
National University of Kaohsiung, TAIWAN
Email: hch.nuk@gmail.com

***Abstract*—Compressive sensing is a potential technology for lossy image compression. With a given quality, we may represent an image with a few significant coefficients in the transform domain. When the number of the significant coefficients is much less than the number of the pixels, the assumption of sparse representation is satisfied. Based on the sparse modeling theories, an image could be sensed with a relatively simple hardware and reconstructed with a powerful computer. We are interested in how to implement the iterative hard thresholding algorithm. The formula is not complex, but the implementation is not straightforward when the image resolution is high. Therefore, the computation complexity and the memory consumptions are analyzed. With the analysis result and the implementation experiences, we discuss the issues that should be considered carefully when implementing the algorithm in this paper.**

***Keywords*-compressive sensing; iterative hard thresholding**

I. Introduction

The mainstream lossy image compression scheme is based on the sampling, transformation, quantization, and entropy coding. According to the Nyquist-Shannon sampling theorem, we may sense a band-limited analog signal at the rate that is at least twice the bandwidth. By sampling at this rate, the sensed discrete-time signal could be reconstructed to the original analog waveform. The discrete-time signal is then transformed to a representation in which only a few coefficients are significant. This approach is feasible because most natural signals are sparse with respect to a certain transform domain. For example, a natural image could be approximated by only a few coefficients of the significant frequencies. Therefore, discrete Fourier transform (DFT) and discrete cosine transform (DCT) are popular in signal processing applications. The real data compression occurs at the following quantization and entropy coding steps. Due to the energy-concentration property, the quality degradation is not much if we carefully preserve the significant coefficients.

With the above process, the key point in the first processing phase is to obtain the significant coefficients. It is known that an image of N pixels could be reconstructed to an acceptable quality level using the K-significant coefficients in the transform domain. Moreover, if the image is smooth, the value K is much less than the value N. This property fits the assumption of the sparse modeling and representation [1]. This leads to a different view of the conventional sample-and-transform phase in image compression. When combining sparse representation and sensing, it is possible to reconstruct the K coefficients from the M measured values. We will see that M is also much less than N. It is effectively doing the compression while sampling the image, and thus the name compressive sensing.

This paper is organized as follows. We briefly introduce the basic concepts of compressive sensing in Sec. II. Then, one of the reconstruction approaches, iterative hard-thresholding (IHT) method, is introduced in Sec. III. The complexity of IHT is analyzed in Sec. IV. The practical issues for implementing IHT are discussed in Sec. V. Finally, we conclude the discussion in Sec. VI.

II. Compressive Sensing

Compressive sensing (CS) is potential for various image processing fields, including pattern recognition, image compression, digital watermarking, etc. In the following sections, the fundamental concepts of CS are described. In Sec. II-A and Sec. II-B, the theories of CS are briefly introduced. The sparsity and the related properties are described in Sec. II-C. Then, the reconstruction concepts is briefly introduced in Sec. II-D.

A. Sensing a Sparse Signal

As mentioned in the previous section, a natural image could be approximated by a small amount of transform-domain coefficients. The property is close to a sparse representation. Thus, it is possible to adopt some of the developed theories of sparse modeling to image compression. One of the interesting work is to combine sparse representation and sensing. Sensing (or sampling) a signal in a vector space is the process to project it to a given set of axes. Suppose the original vector is $\mathbf{x} \in \mathbb{R}^N$, and the sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$. The measured vector $\mathbf{y} \in \mathbb{R}^M$ could be expressed as

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x}. \tag{1}$$

To formalize the expression, the sensing matrix is composed by M unit vectors that represent M axes. The vector $\mathbf{y}$ is the sensing result of projecting $\mathbf{x}$ to the space $\mathbf{\Phi}$. When $\mathbf{x}$ is a sparse vector with only K non-zero components, we call that $\mathbf{x}$ is K-sparsity.

978-1-4799-5390-5/14 $31.00 © 2014 IEEE
DOI 10.1109/IIH-MSP.2014.12

According to the general concept of linear equations, it is possible to determine the unique $\mathbf{x}$ given that $M \geq N$. If $M < N$, there are more than one solution for the underdetermined linear system. However, the solution could be unique if some constraints are assumed, for instance, convex space. A few methods have been developed for solving the unique $\mathbf{x}$ in a sparse space, and the iterative hard thresholding (IHT) method will be introduced in the next section.

The design of the sensing matrix is also an interesting topic. For a given sparse representation model, there should be an optimal design for the projection space that leads to the best reconstruction result. The related researches show that random-sampling is quite effective for most of the cases. Although it may not be the optimal design for a given model, its simplicity is a good reason for the application developers. There are a variety of ways to make a random sensing matrix. For example, the random unit vectors could be generated by Gaussian random variables (Eq. 2).

$$\phi_{i,j} \sim \mathcal{N}(0, \frac{1}{M}). \tag{2}$$

An alternative distribution is the i.i.d. samples from symmetric Bernoulli distribution (Eq. 3).

$$\phi_{i,j} \sim \begin{cases} +\frac{1}{\sqrt{M}}, & \text{with probability } \frac{1}{2} \\ -\frac{1}{\sqrt{M}}, & \text{with probability } \frac{1}{2} \end{cases} \tag{3}$$

Another popular one is designed as in Eq. 4.

$$\phi_{i,j} \sim \begin{cases} +\sqrt{\frac{3}{M}}, & \text{with probability } \frac{1}{6} \\ 0, & \text{with probability } \frac{2}{3} \\ -\sqrt{\frac{3}{M}}, & \text{with probability } \frac{1}{6} \end{cases} \tag{4}$$

In our previous work [2], we compared the three sensing matrix generation methods. We found that Eq. 3 is a balanced design with low implementation complexity and fair reconstruction result.

B. Sensing a Non-sparse Signal

In the above discussions, we assume that $\mathbf{x}$ is K-sparsity. Sparse modeling is particularly useful when $K \ll N$. In many cases, it is not easy to directly obtain (or measure) the optimal K-sparsity representation. Therefore, the indirect approach should be used. Suppose the acquirable data $\mathbf{x}$ is a linear transformation of the corresponding sparse representation $\mathbf{z}$, i.e.,

$$\mathbf{x} = \mathbf{\Psi}\mathbf{z}. \tag{5}$$

Thus, Eq. 1 is re-written as Eq. 6.

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} = \mathbf{\Phi}\mathbf{\Psi}\mathbf{z}. \tag{6}$$

It could be interpreted as follows: given the acquired signal $\mathbf{x}$, we may project it to space $\mathbf{\Phi}$ and obtain the result $\mathbf{y}$. Based on the measured $\mathbf{y}$, we can derive the sparse signal $\mathbf{z}$ by solving Eq. 6. The original $\mathbf{x}$ could be reconstructed by applying the linear transformation $\mathbf{\Psi}$ to $\mathbf{z}$.

In the interpretation, we do not assume a specific transformation $\mathbf{\Psi}$. This is one of the interesting properties of compressive sensing. We may measure $\mathbf{y}$ from the acquirable $\mathbf{x}$ with a known projection space $\mathbf{\Phi}$. When a sparse transformation $\mathbf{\Psi}$ is determined, the corresponding representation $\mathbf{z}$ is reconstructed from $\mathbf{y}$. The reconstruction quality could be enhanced each time an improved transformation is developed. In other words, the sensed data is ready for the future improved sparse representation.

C. The Number of Measurements

The sparsity K depends on the representation basis $\mathbf{\Psi}$. The optimal basis is often data-dependent. Some popular bases, such as the DCT and DWT, are general-purpose and suboptimal to a specific input data. They are useful due to not only data-independence but also consistency. For example, a signal representation in the DCT domain generally compliant to the sparse model. There are many insignificant coefficients that can be ignored (treated as zero) without much degradation in image quality. Therefore, the choice of K could be though of as a quality parameter.

In addition to the K-sparsity assumption, additional constraints such as the restricted isometric property (RIP) should be satisfied in order to solve the unique $\mathbf{z}$. Since K-sparsity is the optimal representation (under a given quality), we may expect that the dimensionality of $\mathbf{y}$ is larger than K. The required number of measurements depends on the type of reconstruction approach. For instance, many reconstruction algorithms are based on the ℓ_p-norm minimization. The theoretical choices of M depends on the level of the norm:

- For ℓ_0 minimization, only $2K$ measurements are needed;
- For ℓ_1 minimization, $M \geq C * \mu^2(\mathbf{\Phi}, \mathbf{\Psi}) * K * \log(N)$.

We could see that the ℓ_p-norm minimization is not the ideal reconstruction approach because $M > K$. However, M could be several orders less than N, which means a great compression ratio.

D. Reconstruction

To solve Eq. 6, the $\mathbf{\Phi}$ and $\mathbf{\Psi}$ should be known. The reconstruction problem is how to determine $\mathbf{z} \in \mathbb{R}^N$ from $\mathbf{y} \in \mathbb{R}^M$. This is equivalent to solving an underdetermined linear system because M is less than N. There are infinite solutions if $\mathbf{z}$ could be arbitrary. When $\mathbf{z}$ is constrained to K-sparse, there could be only one solution. Among the developed reconstruction methods, the most popular approach is the ℓ_p-norm minimization.

The ℓ_2 minimization has a convenient close form, but it does not usually determine a sparse solution. The ℓ_0 minimization, though giving the desired result, is NP-hard. It is computationally intensive and not practical for solving the problem. Therefore, the ℓ_1 minimization is the popular

approach. Many reconstruction methods are designed to solve the ℓ_1 minimization. Some of them are listed below:

- orthogonal matching pursuit
- gradient pursuit
- CoSaMP
- iterative hard thresholding (IHT)
- model-based IHT

Among these methods, iterative hard-thresholding (IHT) is a straightforward approach. The operations looks like the iterative linear equations solver that minimizes the residual. We are interested in how to implement the reconstruction process using IHT. The related issues are discussed in the following sections.

III. Iterative Hard Thresholding Algorithm

In this section, we focus on the iterative hard thresholding (IHT) [3] method for reconstructing the K-sparsity signal. To simplify the discussion, we assume that the acquired signal $\mathbf{x}$ is K-sparse. As long as the transformation $\mathbf{\Psi}$ satisfies the constraints, we can replace all the sensing matrix $\mathbf{\Phi}$ with $\mathbf{\Phi\Psi}$ in the following discussions.

A. Basic IHT Algorithm

Based on Eq. 1, we solve $\mathbf{x}$ iteratively with ℓ_1 minimization. The approximated solution at iteration i is denoted by $\mathbf{x}^{(i)}$. The difference to the measured vector is $\mathbf{y} - \mathbf{\Phi x}^{(i)}$. To minimize the difference, we need to remove the residual from $\mathbf{x}^{(i)}$. Ideally the residual could be computed by $\mathbf{\Phi^{-1}}(\mathbf{y} - \mathbf{\Phi x}^{(i)})$. The problem is that $\mathbf{\Phi^{-1}}$ is not easy to derive when $\mathbf{\Phi}$ is very large. Therefore, the "pseudo inverse matrix" concept comes to rescue. When $\mathbf{\Phi}$ is random, $\mathbf{\Phi^T\Phi}$ is close to an identity matrix. Thus,

$$\mathbf{\Phi^T\Phi} \approx \mathbf{I} = \mathbf{\Phi^{-1}\Phi}. \tag{7}$$

To stabilize the iterative process, a step size is introduced. The core formula to derive the revised solution is

$$\hat{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} + \mu\mathbf{\Phi^T}(\mathbf{y} - \mathbf{\Phi x}^{(i)}). \tag{8}$$

Note that $\hat{\mathbf{x}}$ may not be K-sparsity. Therefore, we need to apply the hard-thresholding function $H_K(\bullet)$ which selects the K largest (in magnitude) components and zero out the others. The IHT algorithm iteratively evaluate the following formula until it converges.

$$\mathbf{x}^{(i+1)} = H_K(\mathbf{x}^{(i)} + \mu\mathbf{\Phi^T}(\mathbf{y} - \mathbf{\Phi x}^{(i)})). \tag{9}$$

To start the iterative process, the initial $\mathbf{x}^{(0)}$ is zero. The convergence condition could be the maximum number of iterations, the sum of absolute difference (or the mean square error) to the measurements, or both.

B. Alternative IHT Algorithm

The selection of step size μ is important to the convergence in both speed and precision. A large step size would converge faster, but it is possible to make the algorithm never converge. A small step size approximates the solution slower, but the algorithm eventually returns the solution. Unfortunately, determining a proper step size is not easy. A solution is to make it adaptive: the closer it is to the convergence, the smaller the step size is. Another one of the approaches to circumvent the tuning of μ is to introduce the inverse matrix $(\mathbf{\Phi\Phi^T})^{-1}$ as

$$\mathbf{x}^{(n+1)} = H_K(\mathbf{x}^{(n)} + \mathbf{\Phi}^T(\mathbf{\Phi\Phi^T})^{-1}(\mathbf{y} - \mathbf{\Phi x}^{(n)})). \tag{10}$$

The use of the inverse matrix [4] guarantees the stability and removes the parameter μ, which makes the IHT easier to use.

IV. Complexity of IHT

The IHT formula is apparently simple. All the operations are multiplications and additions except for the hard-thresholding function. To implement the algorithm with a programming language, we would encounter the scalability problem. For instance, a 512×512 image contains 0.25M pixels, i.e., its DCT consumes 2G bytes of memory (assuming IEEE754 floating-point representation). To sense 8K measurements, 2G multiplications and 2G additions are required. In this section, we analyze both the computation complexity and memory consumption of the IHT algorithm.

A. Computation Complexity

In this section, we analyze the complexity when computing $\mathbf{x}^{(i+1)}$. Since the sensing matrix $\mathbf{\Phi}$ is known, part of Eq. 9 and Eq. 10 could be computed in advance. For Eq. 9, the constant μ and $\mathbf{\Phi^T}$ are multiplied. For Eq. 10, $\mathbf{\Phi\Phi^T}$ is computed, the inverse matrix is then computed, and finally multiply it with $\mathbf{\Phi^T}$. The complexity for each step (based on the last result) is listed below:

Formula	Subexpression	Complexity
Eq. 9	$\mu\mathbf{\Phi^T}$	$O(NM)$
Eq. 10	$\mathbf{\Phi\Phi^T}$	$O(NM^2)$
	$(\mathbf{\Phi\Phi^T})^{-1}$	$O(M^3)$
	$\mathbf{\Phi^T}(\mathbf{\Phi\Phi^T})^{-1}$	$O(NM^2)$

Although there is low-complexity method to compute inverse matrix (complexity $O(M^{\log_2(7)})$), it is not available in most of the programming libraries. Therefore, the complexity is estimated as using Gaussian elimination method. It is obvious that Eq. 10 requires more operations in the pre-computation phase than Eq. 9 does. This is a trade-off to guarantee the stable convergence without concerning the step size. The precomputed part is one-time effort, and it could be ignored if we repeatedly reuse the same $\mathbf{\Phi}$ for reconstruction. For the run-time computation complexity, both equations have the same number of operations to execute.

Subexpression	Complexity
$\mathbf{\Phi x}^{(i)}$	$O(NM)$
$\mathbf{y} - \mathbf{\Phi x}^{(i)}$	$O(M)$
residual	$O(NM)$
$\hat{\mathbf{x}}$	$O(N)$
$H_K(\bullet)$	$O(NK)$

B. Memory Consumption

To estimate the memory consumption, we assume each single value is a floating-point number. Thus, the required memory of the given data structure is represented as the number of floating-point values. The following analysis shows the minimum memory required, and the in-place operation has been taken into account.

Date Structure	Memory Consumption
$\mathbf{x}^{(i)}$	N
$\mathbf{\Phi}$	NM
$\mathbf{\Phi x}^{(i)}$	M
precomputed part	NM
residual	N

V. Implementation Issues

When implementing CS mechanism, the sensing part is relatively easy. The single-pixel camera [5] demonstrated the possibility to implement in wired hardware. However, the reconstruction part is too complex to be implemented in an embedded system. As shown in the previous section, both the computation complexity and memory consumption are pretty high. Considering the runt-time computation, the overall complexity is $O(NM)$. Since M is proportional to $K\log(N)$ (and $K \ll N$), the overall complexity is $O(N\log(N))$. For high-resolution images, the computation time is thus not negligible. This implies that a powerful processor is required for reconstruction within a reasonable time. For low-resolution images, unfortunately, CS is not applicable to this situation. CS should work under the assumption of sparse representation. When N is small, either $\frac{K}{N}$ is not small enough to satisfy the assumption or $\frac{M}{N}$ is too large to be sensible in terms of "compressive".

Considering the memory consumption, the largest two pieces of data structures are $\mathbf{\Phi}$ and the precomputed matrix. Each of them consists NM numbers. Suppose a 512×512 image is sensed 8192 times, there are 2G numbers in each matrix. Thus, a total 32G bytes of memory (IEEE754 format) are required. Even on a high-end PC, it is not likely to keep all the data structures in the main memory. There are two approaches to solve the memory consumption issue:

- **Use the virtual memory:** Modern operating systems are capable of managing virtual memory. We can allocate the required memory blocks as if they are all in the main memory. The OS takes care about when to swap the memory pages between the main memory and the virtual memory.
- **Slicing the matrix multiplication:** As a programmer, we may choose when to load the necessary data. One method is to slice the matrices, and load the necessary chunks into the main memory to do the partial multiplication.

No matter which approach we use, the drawback is to read/write matrix data on the hard disk. It degrades the run-time performance of the IHT algorithm. The former is consistent and portable, but its performance relies on how smart the OS determines the locality of the memory access pattern. The latter requires a function that loads and multiplies the sliced matrices. We prefer the latter approach because the memory usage is under control. Part of the memory could be reserved for other processes and thus reduce the interferences on the memory bus.

VI. Conclusions

In this paper, we reviewed the fundamental concepts of compressive sensing. The interesting property is that the sensed result is ready for the future reconstruction. The computation complexity and the memory consumption of the iterative hard thresholding method were analyzed. According to the analysis result and our programming experiences on this topic, we discussed the implementation issues: (1) the image resolution should be high enough to satisfy the sparsity model; (2) the computation complexity of IHT is not negligible; (3) the memory consumption is so high that we need to slice the matrix multiplication to reduce the memory contention.

Acknowledgment

This work was partially supported by the NSC, Taiwan, under Grants NSC 102-2221-E-032-066.

References

[1] E. Candes and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, 2008.

[2] F.-C. Chang and H.-C. Huang, "Comparison on different random basis generator of a single-pixel camera," in *Robot, Vision and Signal Processing (RVSP), 2013 Second International Conference on*, Dec 2013, pp. 1–4.

[3] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265 – 274, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520309000384

[4] A. Maleki, "Coherence analysis of iterative thresholding algorithms," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, 2009, pp. 236–243.

[5] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-pixel imaging via compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 83–91, 2008.