An Adaptive and Information Theoretic Method For Compressed Sampling

Akram Aldroubi  
Department of Mathematics  
Vanderbilt University  
Nashville, TN 37240  
Email: akram.aldroubi@vanderbilt.edu  

Haichao Wang  
Department of Mathematics  
Vanderbilt University  
Nashville, TN 37240  
Email: haichao.wang@Vanderbilt.Edu

Abstract—By considering an \( s \)-sparse signal \( x \sim (X, P) \) to be an instance of vector random variable \( X = (X_1, \ldots, X_n)^t \) we determine a sequence of binary sampling vectors for characterizing the signal \( x \) and completely determining it from the samples. Unlike the standard approaches, ours is adaptive and is inspired by ideas from the theory of Huffman codes. The method seeks to minimize the number of steps needed for the sampling and reconstruction of any sparse vector \( x \sim (X, P) \).

We prove that the expected total cost (number of measurements and reconstruction combined) that we need for an \( s \)-sparse vector in \( \mathbb{R}^n \) is no more than \( s \log n + 2s \).

I. INTRODUCTION

The standard methods for compressed sampling are non-adaptive. For \( s \)-sparse signals in \( \mathbb{R}^n \), a fixed \( m \times n \) matrix \( A \) is chosen for sampling. The vector of samples \( y = Ax \) is obtained, and the goal is to reconstruct \( x \) via a reconstruction algorithm that we wish to be stable and robust, see e.g., [3], [4], [7], [12], [15], [16], [8], [10], [11], [18]. The main difficulty in all of these methods is the deterministic construction of an appropriate matrix \( A \) that has a number of rows \( m \) of the order \( s \log n \). Another issue is a fast reconstruction algorithm.

Recently, there is an interest in adaptive methods for compressed sampling. One approach uses a Bayesian method combined with a gaussian model for the measurements and a Laplacian model for the sparsity [14]. The sampling vectors are chosen by minimizing a differential entropy. Another approach is a type of binary search algorithm that uses back projection and block adaptive sampling to focus on the possible nonzero component.

In this note, we present the results of an information theoretic adaptive method for compressed sampling developed in [1]. This method uses tools from the theory of Huffman codes to develop a deterministic construction of a sequence of binary sampling vectors \( \{a_\Lambda\} \). The signal \( x \in \mathbb{R}^n \) is assumed to be an instance of a vector random variable \( X = (X_1, \ldots, X_n)^t \) and the choice of the \( i \)-th row \( a_i \) of \( A \) depends on the sample \( y_{i-1} = (a_{i-1}, x) \), and is constructed in such a way as to make the average number of samples needed to determine a signal \( x \) as small as possible, and to make the reconstruction of \( x \) from those samples as fast as possible. This is done by taking advantage of the probability distribution of the random vector \( X \) in order to minimize the average number of samples needed to uniquely determine the unknown signal \( x \).

In our method, rather than constructing a fixed set of sampling vectors forming the rows of a single sampling matrix \( A \) for all possible signals, we construct \( s \) sequences of sampling vectors. Each sequence focuses on finding exactly one nonzero component of the signal \( x \).

II. HUFFMAN TREES AND THE CONSTRUCTION OF SAMPLING VECTORS

Let \( X = (X_1, \ldots, X_n)^t \) denote a vector of \( n \) random variables, and let \( \Omega = \{1, \ldots, n\} \) be the set of all indices, and \( \Lambda \subset \Omega \) to be a subset of indices. Then \( P_\Lambda = P(\Lambda) \) will denote the probability of \( X \) having nonzero components exactly at coordinates indexed by \( \Lambda \), i.e., \( P_\Lambda = Pr\{X_i \neq 0, i \in \Lambda ; X_i = 0, i \in \Lambda^c\} \). Here \( \Lambda^c \) denotes the complement of \( \Lambda \). Thus obviously \( \sum_\Lambda P_\Lambda = 1 \).

We will need the probabilities \( q_\Lambda = q(\Lambda) = Pr(E_\Lambda) \) for the events \( E_\Lambda = \{X_i \neq 0, \text{ for some } i \in \Lambda\} \), which is the probability that at least one of the components of \( X \) with index in \( \Lambda \) is nonzero. Note that \( q \) can be computed from \( P \) by \( q_\Lambda = \sum_{\eta \cap \Lambda \neq \emptyset} P_{\eta} \). However, also note that \( \sum_\Lambda q_\Lambda > 1 \) in general.

A. Construction of the Huffman Tree

We define a Huffman tree to be a binary tree whose leaves are the sets \( \{0\}, \ldots, \{n\} \), and we associate probabilities \( q(0), \ldots, q(n) \) to these nodes respectively. The Huffman tree is constructed from the leaves to the root as follows: Suppose that the nodes at the \( i \)-th step are \( \Lambda_1, \ldots, \Lambda_s \). Let \( i \) and \( j \) be such that

\[
q_{\Lambda_i} = \min_{1 \leq \Lambda \leq s} q_{\Lambda}, \quad q_{\Lambda_j} = \min_{1 \leq \Lambda \leq s, \Lambda \neq \Lambda_i} q_{\Lambda}.
\]

Then, the nodes at the \( (i + 1) \)-th step are obtained by replacing \( \Lambda_i \) and \( \Lambda_j \) with \( \Lambda_i \cup \Lambda_j \) in the list of the nodes at the \( i \)-th step and the probability associated to the node \( \Lambda_i \cup \Lambda_j \) will be \( q_{\Lambda_i \cup \Lambda_j} \), i.e., at each step the two nodes with smallest probabilities are combined. An illustrative example is shown below.
Example: Assume that $X \in \mathbb{R}^4$ is a 2-sparse random vector with probability mass function $P$ defined by: $P_\emptyset = 0.02, P_{\{1\}} = 0.07, P_{\{2\}} = 0.05, P_{\{3\}} = 0.03, P_{\{4\}} = 0.01, P_{\{1,2\}} = 0.31, P_{\{1,3\}} = 0.2, P_{\{1,4\}} = 0.03, P_{\{2,3\}} = 0.06, P_{\{2,4\}} = 0.12, P_{\{3,4\}} = 0.01$. The nodes at the first step are: $\{1\}, \ldots, \{4\}$. Simple computations yield: $q_{\{1\}} = 0.6, q_{\{2\}} = 0.54, q_{\{3\}} = 0.3, q_{\{4\}} = 0.26$. Therefore the nodes at the second step are $\{1\}, \{2\}, \{3, 4\}$. Also $q_{\{3,4\}} = 0.55$ and hence the nodes at the third step are $\{1\}, \{2, 3, 4\}$. The root node is $\{1, 2, 3, 4\}$ with probability $q_{\{1,2,3,4\}} = 1$. This completes the Huffman tree (See Figure 1).

B. Huffman sampling vectors

We first focus on finding one nonzero component, if it exists. Other nonzero components are then iteratively found one at a time in a similar manner until all of the nonzero components are exhausted. To do this, we first construct a Huffman tree associated with the random vector $X$. Each node $\Lambda$ which is not a leaf has two children $\Lambda_1$ and $\Lambda_2$. Note that $\Lambda_1 \cap \Lambda_2 = \emptyset$ and $\Lambda_1 \cup \Lambda_2 = \Lambda$. We denote $\ell_{\Lambda_1} = q_{\Lambda_1} (\log |\Lambda_1| + 1) + (1 - q_{\Lambda_1}) (\log |\Lambda_2| + 1)$ and $\ell_{\Lambda_2} = q_{\Lambda_2} (\log |\Lambda_2| + 1) + (1 - q_{\Lambda_2}) (\log |\Lambda_1| + 1)$. We associate a sampling vector to such nodes $\Lambda$ by:

$$a_\Lambda = \begin{cases} 
\chi_{\Lambda_1} & \text{if } \ell_{\Lambda_1} \leq \ell_{\Lambda_2}, \\
\chi_{\Lambda_2} & \text{if } \ell_{\Lambda_1} > \ell_{\Lambda_2},
\end{cases}$$

where $\chi_\Lambda$ denotes the characteristic function of the set $\Lambda$, i.e., for $\ell_{\Lambda_1} \leq \ell_{\Lambda_2}$, we have $a_\Lambda(i) = 1$ for $i \in \Lambda_1$ and $a_\Lambda(i) = 0$ for $i \in \Omega - \Lambda_1$, and for $\ell_{\Lambda_1} > \ell_{\Lambda_2}$ we have $a_\Lambda(i) = 1$ for $i \in \Lambda_2$ and $a_\Lambda(i) = 0$ for $i \in \Omega - \Lambda_2$.

The choice of the sampling vector in (1) can be seen as follows: Since our goal is to find the nonzero component as quickly as possible, it seems that we would need $a_\Lambda = \chi_{\Lambda_1}$ for the set $\Lambda_1$ with the highest probability $q_{\Lambda_1}$, $i = 1, 2$. However, the set $\Lambda_1$ with the highest probability $q_{\Lambda_1}$ may also have a large number of elements. Thus the choice should be a compromise between the size of the set and its probability.

C. Determination of a sparse vector $x$ using Huffman sampling vectors.

Let $x$ be an $s$-sparse signal in $\mathbb{R}^n$ which is an instance of $(X, P)$ (we will write $x \sim (X, P)$). We make the additional assumption that the conditional probability $Pr(x_{i} \neq 0 | X_i = 0, i \in \Lambda) = 0$ holds for any $\Lambda \subset \Omega$ (recall that $\Omega = \{1, \ldots, n\}$). This is a natural condition if the random variables $X_i, i = 1, \ldots, n$, in the random vector $X$ do not have a positive mass concentration except possibly at zero.

1) Finding a nonzero component: Algorithm 1 below is used to find the position and the corresponding value of one of the nonzero components of $x$ (if any).

**Algorithm 1:**

1. Initialization: $\Lambda = \Omega$;
2. Repeat until $|\Lambda| = 1$
   a. If $(a_\Lambda, x) \neq 0$, $\Lambda = \Lambda_1$
   b. Else $\Lambda = \Lambda_2$
   end repeat
3. Output the (only) element $t_1 \in \Lambda$
4. Output $x_{t_1} = \langle \chi_{(t_1)}, x \rangle$

**Remark 1:** If the vector $x = 0$, then the algorithm will find an output $x_{t_1} = 0$, otherwise it will output the value $x_{t_1}$ of one of the nonzero components of $x$ and its index $t_1$.

The first observation is that Algorithm 1 is optimal for 1-sparse vectors. This should not be a surprise since the algorithm was inspired by the theory of Huffman codes. We have

**Theorem 2.1:** Given a 1-sparse vector $x \sim (X, P)$ in $\mathbb{R}^n$. Then the average number of samples $L_1(X, P) = \sum_{i} P_i \ell_{\Lambda_i}$ needed to find $x$ using Algorithm 1 is less than or equal to the average number of samples $L_\Lambda(X, P)$ for finding $x$ using any algorithm $\Lambda$ with binary sampling vectors.

The 1-sparse case is a special case. It is optimal because the sampling scheme can be associated exactly with the Huffman codes and we have that $q_{\Lambda} = P_{\Lambda}$ and $\sum_i q_{i} = \sum_i P_i = 1$. In fact, the choice of $\Lambda$ in (1) can be chosen to be either $\chi_{\Lambda_1}$ or $\chi_{\Lambda_2}$ independently of the values of $\ell_{\Lambda_1}, \ell_{\Lambda_2}$. However, for the general $s$-sparse case, we do not have $q_{\Lambda} = P_{\Lambda}$ anymore, and the sampling vectors cannot be associated with Huffman codes directly. Thus for the $s$-sparse case, the average number of sampling vectors is not necessarily optimal and we need to estimate this number to have confidence in the algorithm.

D. Algorithm for finding all nonzero components of $x$

The general algorithm for finding the $s$-sparse vector $x \sim (X, P)$ which is an instance of the $s$-sparse random vector $(X, P)$, can now be described as follows:

**Algorithm 2:**

Initialization: $k=1; \omega = \emptyset$;
Repeat until $k > s$ or $x_{t_k} = 0$
1. $\Lambda = \Omega - \omega$;
2. Repeat until $|\Lambda| = 1$
   a. If $(a_\Lambda, x) \neq 0$, $\Lambda = \Lambda_1$
   b. Else $\Lambda = \Lambda_2$
   end repeat
3. Output the (only) element $t_k \in \Lambda$;
4. Output $x_{t_k} = \langle \chi_{(t_k)}, x \rangle$;
5. $\omega = \omega \cup \{t_k\}$;
6. $k = k + 1$;
end repeat

Algorithm 2 repeats Algorithm 1 at most $s$ times and adds one extra sample to determine the value of each nonzero
component once its position is known. We have the following theorem whose proof can be found in [1]:

**Theorem 2.2:** Given a nonzero s-sparse vector \( x \sim (X, P) \) in \( \mathbb{R}^n \) the average number of samples \( L \) needed to find the position of one nonzero component of \( x \) using Algorithm 1 is at most \( s \log n + 3s \).

As a corollary of Theorem 2.2 we immediately get

**Corollary 2.3:** Given a nonzero s-sparse vector \( x \sim (X, P) \) in \( \mathbb{R}^n \), the average number of sampling vectors \( L \) needed to find all nonzero components of \( x \) using Algorithm 2 is at most \( s \log n + 2s \).

**Remark 2:**

(i) Corollary 2.3 states that the upper bound on the expected total cost (number of measurements and reconstruction combined) that we need for an s-sparse vector in \( \mathbb{R}^n \) using Algorithm 2 is no more than \( s \log n + 2s \).

(ii) If the probability distribution \( P \) is uniform then the combined cost of the measurements and reconstruction is exactly \( s \log n + 2s \).

**E. Noisy measurements**

In practice, the measurements \( \{y_i\} \) may be corrupted by noise. Typically the noise is modeled as additive and uncorrelated: \( y_i = \langle x, \alpha_i \rangle + \eta_i \) (see [5]). Our algorithm needs to be modified accordingly to deal with noisy measurements case. Algorithm 2 can be modified by changing the statement \( \langle a_i^{T-\omega}, x \rangle \neq 0 \) to the statement \( \langle a_i^{T-\omega}, x \rangle > T \), where the threshold \( T \) is of the same order as the standard deviation of \( \eta_i \).

Consider the model \( Y = X + \eta \) where the signal \( X \sim N(0, \sigma_X^2) \) and the noise \( \eta \sim N(0, \sigma_\eta^2) \). Then \( Y \sim N(0, \sqrt{\sigma_X^2 + \sigma_\eta^2}) \). We set the threshold in Algorithm 2 to be \( T = E[|\eta|] = \frac{2\sigma_\eta}{\sqrt{\pi}} \), and consider a measure of error (for one sample) given by the probability

\[
p(e) = P(|Y| < T \text{ and } |X| \geq T) + P(|Y| \geq T \text{ and } |X| < T)
\]

After easy computation, we have that

\[
p(e) = \text{erf}(\frac{\sigma_\eta}{\sqrt{\pi} \sigma_X}) + \text{erf}(\frac{\sigma_\eta}{\sqrt{\pi} \sigma_Y}) - \text{erf}(\frac{\sigma_\eta}{\sqrt{\pi} \sigma_X}) \text{erf}(\frac{\sigma_\eta}{\sqrt{\pi} \sigma_Y}),
\]

where \( \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \) is the error function.

Using the Taylor series, we obtain

\[
p(e) = \frac{4t}{\pi} - \frac{4t^2}{\pi^2} + o(t^2),
\]

where \( t = \frac{\sigma_\eta}{\sigma_X} \) is the ratio of the standard deviations of the noise and the signal. Thus, for a relatively large signal to noise ratio, \( t \) will be small and we get that the probability of at least one error in the sampling-reconstruction for an s-sparse vector is bounded above by the quantity

\[
p_e(s) \approx s(\log n + 1) \frac{4t}{\pi}.
\]

It can be seen that \( p_e(s) \) is essentially linear in the sparsity \( s \), linear in \( t \) and logarithmic in the dimension \( n \), as can also be seen in the simulations below.

### II. Examples and Simulations

In this section, we provide some examples and test our algorithm on synthetic data. In the first experiment we use an exponential distribution to generate the position of the nonzero components of the s-sparse vectors, and uniform distribution for their values. The signal \( x \) is generated by first generating an integer index \( i \in [1, n] \) using the exponential pdf with a mean of 10, and then constructing the component \( x(i) = A \cdot \text{rand} - 0.5 \), where \( \text{rand} \) is a random variable with uniform distribution in \([0, 1]\). This process is repeated \( s \) times to construct \( s \) sparse signals. In all the other experiments, we use a uniform probability distribution for both signal and noise. The signal \( x \) is generated by first generating an integer index \( i \in [1, n] \) with a uniform distribution, and then constructing the component \( x(i) = A \cdot \text{rand} - 0.5 \), where \( \text{rand} \) is a random variable with uniform distribution in \([0, 1]\). As in the construction of the exponential pdf case, this process is repeated \( s \) times to construct \( s \) sparse signals. Each additive noise \( \eta_i \), is generated by \( \eta_i = N \ast (\text{rand} - 0.5) \) and added to the measurement \( y_i \). All the experiments are done using Matlab 7.4 on a Macintosh MacBook Pro 2.16 GHz Intel Core Duo processor 1GB 667 MHz RAM.

#### A. Noiseless cases

**Simulation 3.1:** Our first experiment is a sparse vector \( x \sim (X, P) \) in a space of dimension \( n = 2^{15} \) with an exponential pdf with mean 10 for the location of the nonzero components and a uniform distribution for the values of the components as described above. We have tested our algorithm with \( s = 1, 3, 5, 7, 9, 11, 13 \). The mean and variance of the number of sampling vectors needed for the various sparsity \( s \) (for the combined sampling and reconstruction) is shown in Table I.

<table>
<thead>
<tr>
<th>( s \log n )</th>
<th>15</th>
<th>45</th>
<th>75</th>
<th>105</th>
<th>135</th>
<th>165</th>
<th>195</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>9.11</td>
<td>27.5</td>
<td>46.17</td>
<td>61.47</td>
<td>81.26</td>
<td>97.2</td>
<td>111.88</td>
</tr>
<tr>
<td>Var</td>
<td>10.15</td>
<td>16.6</td>
<td>25.5</td>
<td>25.88</td>
<td>31.88</td>
<td>30.94</td>
<td>37.37</td>
</tr>
</tbody>
</table>

#### TABLE I

Mean and variance of number of sampling vectors as functions of sparsity \( s \) for \( n = 2^{15} = 32768 \).

From the table, it can be seen that the average number of steps needed for recovering an \( s \) sparse signal \( x \) (second row) is (as predicted) less than the theoretical upper bound \( s \log n \). Since for each sparsity \( s \) the experiments was repeated many times, the variance of the number of steps needed to recover an \( s \) sparse vector is given in the third row.

#### B. Noisy measurements

**Simulation 3.2:** In this test we fix the following values: \( n = 512, A = 20, N = 0.1 \). For each value of \( s \), we construct 100 s-sparse signals in \( \mathbb{R}^n \). We test the effect of \( s \) on the \( \ell^2 \) relative error \( \frac{\|x - x^\dagger\|_2}{\|x\|_2} \) (in percent) as a function of the sparsity \( s \). The results are displayed in Figure 2. A regression line is fitted to the data and shows that the relative error increases linearly with \( s \), as predicted in Section II-E.
The value of vectors are binary and their construction are deterministic and can be produced explicitly for each $n$. Without noise the reconstruction is exact, and the average cost for sampling and reconstruction combined for an $s$ sparse vector is bounded by $s \log(n) + 2s$. As a comparison, the standard approaches use $O(s \log n)$ for the sampling alone, and the reconstruction is usually much more expensive (e.g. $O(n^3)$ for Linear Programming) so that the combined sampling and reconstruction is more costly than our method. We have also shown that the method is stable in noisy measurements. However, the current method and algorithms are not adapted to the compressive signals and developments for these cases will be investigated in future research. We hope that the approach will stimulate further developments and interactions between the area of information theory and compressed sampling.

IV. CONCLUSION

We have presented an information theoretic approach to compressed sampling of sparse signals. Using ideas similar to those in Huffman coding, we constructed an adaptive sampling scheme for sparse signals. In our scheme, the sampling vectors are binary and their construction are deterministic and

![Fig. 2. Relative $\ell^2$ error of reconstruction from noisy measurements.](image)

![Fig. 3. Relative $\ell^2$ error of reconstruction from noisy measurements.](image)

![Fig. 4. Relative $\ell^2$ error of reconstruction from noisy measurements.](image)

ACKNOWLEDGMENT

This work was supported in part by NSF Grant DMS-0807464.

REFERENCES