Sparse Bayesian Learning in Compressive Sensing

Abstract

Traditional Compressive Sensing (CS) recovery techniques resorts a dictionary matrix to recover a signal. The success of recovery heavily relies on finding a dictionary matrix in which the signal representation is sparse. Achieving a sparse representation does not only depend on the dictionary matrix, but also depends on the data. It is a challenging issue to find an optimal dictionary to recover non-sparse data or to sparsify the data. Instead of finding the optimal dictionary matrix, this paper shows that Bayesian Learning recovery methods can achieve phenomenal results using general dictionary matrices for non-sparse signals. Our empirical results show that when Bayesian Learning is used to recover non-sparse signals it is not necessity to use an optimal dictionary matrix. Our experiments show that Bayesian Learning has superior performance compared to one of the state-of-the-art optimization techniques used in CS, the SPGL1 algorithm. Bayesian Learning outperformed SPGL1 in terms of number of iterations, SNR, and recovery quality.

1 Introduction

In many sensory applications such as Tele-monitoring of physiological signals, exploration seismology…etc., data are fairly large relive to the limited processing resources. These factors are low space memory, low processing power and ultra-low power communication schemes. High compression rates of these data are highly required in such application. And being confronted with Nyquist's sampling criterion in which exponential increase in volumes of data is an inevitable fact, then better compression /recover techniques is essential (Hermann, 2008).

Current nonlinear data-compression techniques are based on non-linear sampling (e.g., sampling by a CCD chip in a digital camera) followed by a nonlinear encoding to encode a relatively small number of significant transformed domain coefficients (e.g. FFT coeffients of the data) (Mallat, 2009). Compression is accomplished by keeping only the largest coefficients. These compression methods are lossy and compression errors after decompression may occur at different levels of compression (Mallat, 2009). A compression ratio basically states the compressed signal size as a fraction of the size of the original signal. The better the transform domain of the data the more it captures the energy in the sampled data, which leads to larger attainable compression ratios. Although this technique is one of the very basic structures of the digital revolution of many consumer devices, including digital cameras, iPods, etc. (Hermann, 2008), it is still wasteful and not efficient in two important ways. First, high resolution data has to be collected during the sampling step, which is expensive. Second, the encoding phase is nonlinear this means that if we select a compression ratio that is too high (i.e. less samples for cheaper operations), the decompressed signal may result with unacceptable errors. Sometimes in the worst case it is necessary to repeat collection of the high-resolution samples, which is infeasible and

inefficient. This is not acceptable in cases when the storage of high-resolution data is a concern or where the cost of acquisition is the main obstruction. Compressed Sensing (CS) replaces the combination of high-resolution sampling and nonlinear compression in a single step by randomized subsampling technique that linearly combines encoding and sampling in a single step. Randomized subsampling has an advantage which is to encode the samples linearly and does not require high-resolution data. Also encoding in a randomized sampling manner suppresses subsampling artifacts (Hermann, 2010). These artifacts whether they are caused by missing traces of data or by cross talk between sources are turned into relatively harmless incoherent Gaussian noise by randomized subsampling (Herrmann and Hennenfent, 2008).

The idea of CS is to recover sparse signals, signals which contain few non-zero elements, from few linear measurements by using convex optimization. It concerns the recovery of a high-dimensional sparse vector after the compression of this vector. CS relies on specific properties of the compressive-sensing matrix and the to-be-recovered signal should be sparse (Hermann, 2010). Sparse representations of signals received lots of attentions in the recent years. The problem of solving a sparse representation from compression is to search for the most compact representation of a transformation atoms, in which these form a dictionary matrix. In some cases these atoms can be general enough such as Fourier Transform operators, Discrete Cosine Transform (DCT), or different types of Wavelet transform. Choosing the correct dictionary depends on the recovered / compressed signal. So To find a dictionary with enough code words of the data to be recovered is a requirement to solve the convex problem which is one of the major problems in CS. Another problem in CS is to find the sparse representation of a signal x in a given dictionary D.

In general the CS is formulated as follows: Given an M × N, A matrix in equation 1 which is called the sensing matrix, where N > M and usually M << N. The compression rate is controlled by the variable M. For example if 50 % compression is required to the sparse data x then M is equal to N/2. Y is the compressed signal $Y \in R^M$, the problem of sparse representation is to find the N × 1 coefficient of sparse vector x from the compressed Y matrix using the A matrix and the dictionary matrix D.

$$Y = A\,x + v \tag{1}$$

Where v is a Gaussian noise associated in the process. Current CS algorithms use the compressed data Y, and the sensing matrix A to recover the original signal x. Their successes rely on the key assumption that most entries of the signal x are zero (i.e. x is sparse). If the data is not sparse by its nature then sparsifying the data is essential using a dictionary matrix D. One can seek a dictionary matrix $D \in R^{MXM}$, and x can be expressed as x = Dz and z is sparse, where $z = D^{-1}x$. Then, the model in equation 1 can be re-written as

$$Y = ADz + v \tag{2}$$

In such cases when x is not sparse as expressed in equation 2 the CS algorithms first recovers z using y and AD, and then recover the original signal x is recovered by solving x = Dz. It is very hard to find a D matrix that satisfies the sparisity of the data and when D is not valid then the recovery is not poor. The term sparse recovery is associated with CS in which sparisity is required for convex optimization to recover only sparse data.

The basic problem in CS or sparse signal recovery is to recover a sparse signal $\tilde{x}$ from a small number of linear measurements Y. There are lots of algorithms to solve this problem highlighted in equation 3 and it is believed that sparsity is essential to solve the compressed sensing problem.

$$\tilde{x} = \arg min ||x||_1 \quad subject\ to\ Y = Ax \tag{3}$$

In this expression, $x^{\sim}$ represents the estimated values of the L1 norm of x, and minimizing the L1 norm in equation 3 promotes sparsity in x and the equality constraint ensures that the solution respects the acquired data. In cases x is not a sparse representation then finding the optimum dictionary is a current challenging research topic (Hermann, 2008; Hermann 2010).

Many other conventional data compression methodologies such as wavelet compression cannot satisfy the required compression constraints mentioned above (low power operation, low processing speeds and power, cheap hardware, and low power communication schemes). It has been shown that compared to wavelet compression and Compressed Sensing (CS), when using sparse binary matrices as the sensing matrices (A matrix), can reduce energy consumption while achieving competitive data compression ratio. The use of sparse Binary matrices means the device cost largely reduce (Zhang 2011). However, current CS algorithms only work well for sparse signals or signals with sparse representation. Instead of seeking an optimal dictionary matrix for specific data and by using only a general dictionary method, this paper exploits the performance of Bayesian learning in compressed sensing recovery by implementing software that applies Block Sparse Bayesian Learning Algorithm presented by (Zhang in 2013). Our result shows that the Bayesian Learning method can recover the compressed sensing non-sparse signals. This paper shows a performance comparison between the Bayesian Learning Algorithm and one of the best state of the art CS algorithms that solves equation 3 called the SPGL1. Since Electroencephalography (EEG), Physiological brain signals, is neither sparse in the original time domain nor sparse in transformed domains we used it for our testing. Current CS algorithms cannot achieve good recovery quality for EEG signals. And at the best of our knowledge we have achieved the best recovery quality as showed in the results section. Our empirical results suggest that when using Bayesian Learning for EEG compression/recovery, the seeking of optimal dictionary matrices is not a crucial issue when using Bayesian Learning.

Section 2 explains the adopted Bayesian Learning Algorithm method that we have implemented which was published by Zhang 2013. Section 3 explains the experiments we carried out to show the performance of the recovery of Bayesian Learning algorithm with non-sparse EEG signals. The results section shows a performance comparison between SPGL1 and Bayesian Learning. Section 4, and 5 presents a discussion about practices of CS recovery methods and finally the conclusion and the references are presented.

2 Sparse Bayesian Learning

The Bayesian Learning Algorithm was initially proposed to recover a signal with a block structure by dividing it into non-overlapping blocks. It was proved that even if a signal has no specific block structure, the algorithm is still effective. This is important because most non-sparse signals don't have a distinctive block structure such as EEG signals. EEG signals have arbitrary waveforms and their z representation coefficients have no block structure. Therefore Bayesian Learning Algorithm can apply the CS recovery.

The input **x** is a divided into a set of concatenation of g non-overlapping blocks. The size of the block can be arbitrarily defined since this algorithm can work with non-distinctive block structure.

$x = \left[x_1, \dots, x_{d_1}, \dots, x_{d_{g-1}} + 1, \dots, x_{d_g}\right]^T$, where $\left[x_1, \dots, x_{d_1}\right] = x_1^T \dots \left[x_{d_{g-1}} + 1, \dots, x_{d_g}\right] = x_g^T$,

Some of these blocks are non-zero blocks and we will call it k blocks, where$k \ll g$. The location of the k blocks is unknown and arbitrary and by exploiting these blocks partitions the recovery performance is improved (Zang, 2012). The assumption is that each block $x_i \in R^{d_i \times 1}$ satisfy a parameterized multivariate Gaussian distribution with unknown parameters γ_i and $\mathbf{B}_i$:

$$p(\mathrm{x_i}; \gamma_i, \mathbf{B}_i) \sim N(0, \gamma_i \mathbf{B}_i), i = 1, \dots, g \quad (4)$$

where, γ_i is a nonnegative parameter that controls the block-sparsity of x. When γ_i = 0, the i^{th} block = 0. During the learning process $\gamma_i \to 0$ to prompt and encourage the idea of

sparsity at the block level.

The other parameter, $\mathbf{B}_i \in R^{d_i \times d_i}$, is a positive definite matrix that captures the inter-correlation structure of the i^{th} block. Inter-correlation is the correlation among the elements within the block. The Inter-Correlation is useful because it indicates a predictive relationship that can be exploited.

Assuming that the blocks are mutually uncorrelated to each other, the prior of $\mathbf{x}$ is $p(\mathrm{x}; \{\gamma_i, \mathbf{B}_i\}_i) \sim N(0, \Sigma_0)$, where $\Sigma_0 = \mathrm{diag}\{\gamma_1 \mathbf{B}_1, \ldots, \gamma_g \mathbf{B}_g\}$. Also assume that the noise vector is Gaussian distributed$p(\mathbf{v};\ \lambda) \sim N(0, \lambda \mathbf{I})$, and λ is a positive scalar. The posterior of $\mathbf{x}$ can be given by $p(\mathbf{x}|\mathbf{y};\ \lambda, \{\gamma_i, \mathbf{B}_i\}_{i-1}^{g}) = N(\mu_x, \Sigma_x)$ and $\mu_x = \Sigma_0 \mathbf{A}^{\mathrm{T}} (\lambda \mathbf{I} + \mathbf{A}\Sigma_0 \mathbf{\Phi}^{\mathrm{T}})^{-1} \mathbf{y}$ and$\Sigma_{\mathrm{x}} = \left(\Sigma_0^{-1} + \frac{1}{\lambda} \mathbf{A}^{\mathrm{T}} \mathbf{A}\right)^{-1}$. Once the learning parameters λ and $\{\gamma_i, \mathbf{B}_i\}_{i-1}^{g}$ are learned and estimated, $\hat{\mathrm{x}}$, the Maximum-A-Posteriori estimate of $\mathbf{x}$, is directly obtained from the mean of the posterior. The learning parameters are estimated by type II maximum likelihood procedure, whereas the goal is to minimize the following cost function to get the parameters.

$$L(\theta) = \log |\lambda \mathbf{I} + \mathbf{A}\Sigma_0 \mathbf{A}^{\mathrm{T}}| + \mathbf{y}^{\mathrm{T}} (\lambda \mathbf{I} + \mathbf{A}\Sigma_0 \boldsymbol{A}^{\mathrm{T}})^{-1} \mathbf{y}, \tag{5}$$

where θ denotes all the learning parameters λ, $\{\gamma_i, \mathbf{B}_i\}_{i-1}^{g}$.

This algorithm discussed the derived from of the learning rules γ_i, $\mathbf{B}_i$, and λ. Different γ_i learning rules help determine the best possible recovery performance when optimal values of λ and $\mathbf{B}_i$ are given, and lead to convergence. Although γ_i learning rule could sometimes lead to perfect recovery, the performance can be very poor if values of λ cannot be obtained.

In noiseless environments, the parameter $\mathbf{B}_i(\forall i)$ affects local convergence by affecting the shape of basins of the attraction to local minima basin. So the global minimum of the cost function will always lead to the true sparse solution regardless the values of $\mathbf{B}_i$. For this reason$\mathbf{B}_i$ will be constrained to achieve better performance by avoiding over-fitting.

The λ rule is based on the bound-optimization method, which is also known as the Majorization-Minimization method, in which it maintains good performance. The learning rules for λ and $\mathbf{B}_i$ are as follows:

$$\lambda \leftarrow \frac{||(\mathrm{y} - \mathbf{A}\mu_x)||_2^2 + \sum_{\mathrm{i}=1}^{\mathrm{g}} \mathrm{Tr}\left(\Sigma_x^i (\mathbf{A}^{\mathrm{i}})^{\mathrm{T}} \mathbf{A}^{\mathrm{i}}\right)}{\mathrm{M}}, \tag{6}$$

The idea is to find a positive definite and symmetric matrix B so that it is only determined by one parameter. For many applications modeling elements of a block as a first-order Auto-Regressive (AR) process is a sufficient model for intra-block correlation. For this case the Toeplitz matrix was chosen to form this correlation and given as follows:

$$\mathrm{B_i} = \mathrm{Toeplitz}([1, \bar{r}, \ldots \bar{r}^{d_i - 1}]) = \begin{bmatrix} 1 & \bar{r} & \cdots & \bar{r}^{d_i - 1} \\ \vdots & & \ddots & \vdots \\ \bar{r}^{d_i - 1} & \bar{r}^{d_i - 2} & \cdots & 1 \end{bmatrix} \quad \forall i \tag{7}$$

where $\Sigma_x^i \in R^{d_i x d_i}$ is the i^{th} diagonal block in Σ_{x}, $\mathbf{A}^{\mathrm{i}} \in R^{M x d_i}$ is the sub-matrix of $\mathbf{A}$ which corresponds to the i^{th} block of $\mathbf{x}$, $\bar{r} = \frac{\overline{m_1}}{\overline{m_0}}$, $\bar{m}_{0,1} = \sum_{i=1}^{g} m_{0,1}^i$, where m_0 & m_1 are the average of the elements of the main diagonal and sub-diagonal respectively of the following matrix: $\frac{1}{g} \sum_{i=1}^{g} \frac{\Sigma_x^i + \mu_x^i (\mu_x^i)^T}{\gamma_i}$

In equation 5 notice the first term and the second term. We can notice that the first term on the left of the plus operator is concave when $\gamma \geq 0$, where $\gamma = [\gamma_1, \ldots, \gamma_g]^T$ and the second term is convex also when $\gamma \geq 0$.

To minimize the cost function it is required to find an upper-bound for the first term and then minimize the upper-bound. This is done by using the supporting hyper-plane of the first term as its upper-bound, we let γ^* be a given point in the γ –space given in the next equation.

$$\log|\,\lambda\,\mathbf{I}+\mathbf{A}\Sigma_0\mathbf{A}^{\mathrm{T}}| \le \sum_{\mathrm{i}=1}^{\mathrm{g}} \mathrm{Tr}((\Sigma_y^*)^{-1}\mathbf{A}^{\mathrm{i}}\mathbf{B}_i(\mathbf{A}^{\mathrm{i}})^{\mathrm{T}})\,\gamma_i + \log|\Sigma_y^*| - \sum_{\mathrm{i}=1}^{\mathrm{g}} \mathrm{Tr}((\Sigma_y^*)^{-1}\mathbf{A}^{\mathrm{i}}\mathbf{B}_i(\mathbf{A}^{\mathrm{i}})^{\mathrm{T}})\,\gamma_i^*$$

Where, $\Sigma_y^* = \lambda\,\mathbf{I}+\mathbf{A}\Sigma_0^*\mathbf{A}^{\mathrm{T}}$ and $\Sigma_0^* = \Sigma_0|_{\gamma=\gamma^*}$. Substituting this into equation (5), we get

$$L(\gamma) \le \sum_{\mathrm{i}=1}^{\mathrm{g}} \mathrm{Tr}\left((\Sigma_y^*)^{-1}\mathbf{A}^{\mathrm{i}}\mathbf{B}_i(\mathbf{A}^{\mathrm{i}})^{\mathrm{T}}\right)\gamma_i + \mathbf{y}^{\mathrm{T}}(\,\lambda\,\mathbf{I}+\mathbf{A}\Sigma_0\mathbf{A}\boldsymbol{\Phi}^{\mathrm{T}})^{-1}\mathbf{y} + \log|\Sigma_y^*| - \sum_{\mathrm{i}=1}^{\mathrm{g}} \mathrm{Tr}\left((\Sigma_{\mathrm{y}}^*)^{-1}\boldsymbol{A}^{\mathrm{i}}\mathbf{B}_i(\mathbf{A}^{\mathrm{i}})^{\mathrm{T}}\right)\gamma_i^* = \tilde{L}(\gamma) \quad (8)$$

which is convex over γ. When $\gamma=\gamma^*$, then $L(\gamma^*)=\tilde{L}(\gamma^*)$. Further, for any $\gamma_{\min}$ which minimizes $\tilde{L}(\gamma)$, we have the following relationship: $L(\gamma_{\min}) \le \tilde{L}(\gamma_{\min}) \le \tilde{L}(\gamma^*) = L(\gamma^*)$. This shows that when we minimize the surrogate function $\tilde{L}(\gamma)$ over γ, the resulting minimum point decreases the $L(\gamma)$ in equation 3 efficiently.

By applying $\mathbf{y}^{\mathrm{T}}(\,\lambda\,\mathbf{I}+\mathbf{A}\Sigma_0\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{y} = \mathbf{min}_{\mathbf{x}}[\frac{1}{\lambda}||\mathbf{y}-\mathbf{A}\mathbf{x}||_2^2 + \mathbf{x}^{\mathrm{T}}\Sigma_0^{-1}\mathbf{x}]$, then defining a new upper bound for $\tilde{L}(\gamma)$, $G(\gamma,\mathbf{x})$ that is convex in both γ and x, the solution (γ^o) of $\tilde{L}(\gamma)$ is the solution $(\gamma^o,\mathbf{x}^{\mathrm{o}})$ of $G(\gamma,\mathbf{x})$. Whereas $G(\gamma,\mathbf{x})$ is the final surrogate cost function.

Taking the derivative of $G(\gamma,\mathbf{x})$ with respect to γ_i, we get

$$\gamma_i \leftarrow \sqrt{\frac{\mathbf{x}_{\mathrm{i}}^{\mathrm{T}}\mathbf{B}_{\mathrm{i}}^{-1}\mathbf{x}_{\mathrm{i}}}{Tr\left((\mathbf{A}^{\mathrm{i}})^{\mathrm{T}}(\Sigma_{\mathrm{y}}^*)^{-1}\boldsymbol{A}^{\mathrm{i}}\mathbf{B}_{\mathrm{i}}\right)}} \quad (9)$$

The next section shows the results of the implementation of the learning rule in the prediction model $\mu_x = \Sigma_0\mathbf{A}^{\mathrm{T}}(\,\lambda\,\mathbf{I}+\mathbf{A}\Sigma_0\boldsymbol{\Phi}^{\mathrm{T}})^{-1}\mathbf{y}$

3 Results

Firstly, the first experiment shows the results of the comparison between the two algorithms in terms of number of iterations, and SNR. It introduces two more measures used to measure the recovery of the signal. The second experiment shows the performance of both algorithms in noisy environments so that it ranges from 5 to 25 dB. The third experiment shows the performance of Bayesian Learning for 2D data. At the best of our knowledge these recovery results are the best results ever achieved on EEG Signals using compressed sensing techniques. Both algorithms were tested using the same data which were acquired from the dataset 1 of the Berlin BCI group (Blankertz, 2007).

3.1 Experiment 1: Comparison of CS using DFT, DWT, Wavelets D matrix

Bayesian Learning is compared with one of the state of the art recovery algorithms SPGL1 in terms of recovery quality. Both methods algorithms adopted the same sensing matrix A. Thus, they have equal power consumption so the comparison of energy consumption is excluded. Three performance indexes were used to measure recovery quality. One was the Normalized Mean Square Error (NMSE), defined by $\frac{||x-\tilde{x}||_2^2}{||x||_2^2}$, where x is the original signal and $\tilde{x}$is the recovered signal. The second method is the Structural SIMilarity index (SSIM) introduced by (Zhang 2009) for 1-dimensional signals (the length of the sliding window used was 100). SSIM is a method to measure the similarity between the recovered signal and the original signal. The higher the SSIM the better the recovery quality and when the recovered signal is the same as the original signal the SSIM = 1. The results of the comparison are tabulated in table 1. These tests show the condition of matrix D of (size N-by-N) different types of non-sparse dictionary matrices as shown in table 1. Thus $x = D^{-1}z$ whereas z is recovered by both algorithm. The number of samples of the original signal N was equal to 20480 samples. The compression rate M = N/2 (50% compression). The sensing matrix A is a binary matrix of size M-by-N, in which every column contained 2 entries of ones located at random locations, while the rest of the entries are zeros. For the Bayesian Learning Algorithm the block partition is set to be 24 blocks which is chosen randomly. The

maximum number of iterations for the Bayesian Learning was set to 7 as for the SPGL1 was set to 350 iterations insuring the best performance of both algorithms. Both algorithms almost consumed the same time of 20.1 seconds to execute on a 2GHz Core duo processor and 2GB DDR2 RAM.

Table1: Comparison of Bayesian Learning with SPGL

Bayesian Learning Results	SNR	NMSE	SSIM	SPGL1 Results	SNR	NMSE	SSIM
1D DCT	28.3	0.08	0.82	1D DCT	20.2	0.18	0.60
1D Wavelet Haar	25.9	0.14	0.75	1DWavelet_Haar	21.8	0.24	0.63
1D_Wavelet db8	24.6	0.10	0.80	1D_Wavelet db8	22.4	0.20	0.67
1D_Wavelet db10	25.3	0.09	0.80	1D_Wavelet db10	20.4	0.19	0.68
1D_Wavelet db14	25.1	0.09	0.80	1D_Wavelet db14	21.6	0.19	0.67
1D_Wavelet db16	25.0	0.09	0.81	1D_Wavelet db16	21.9	0.20	0.67
1D_Wavelet db20	24.8	0.09	0.80	1D_Wavelet db20	23.4	0.19	0.67

As shown in the above table the Bayesian learning outperforms the SPGL1 algorithm in SNR, NSME, and SSIM with a significant less number of iterations using a general dictionary matrix. Figure 1 shows a segment of the original EEG signal and the reconstructed data using Bayesian Algorithm and SPGL1. The original data was compressed at 50% compression rate and it is clearly shown that the reconstruction of SPGL1 is noisy and the quality is poor. On the other hand Bayesian Learning algorithm achieved good reconstruction. As shown in this figure the transform coefficients of the data are not sparse and Bayesian Learning algorithm still reconstructed the EEG. This figure illustrates that Bayesian Learning technique does not need a sparsifying Dictionary matrix D to reconstruct the compressed signal.

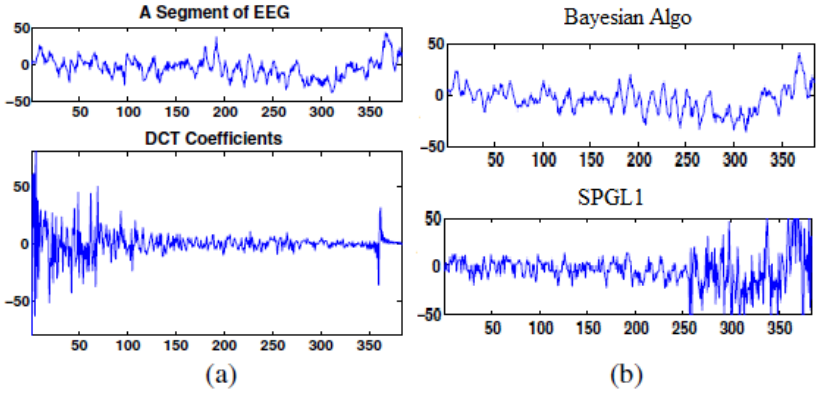

Figure 1: (a) EEG data segment & its DCT Coefficients. As shown in the figure the DCT coefficients are non-sparse, (b) Recovery using Bayesian Algorithm & SPGL1

3.2 Experiment 2: Comparison of CS with SPGL1 in noisy environment.

In this experiment we compared the Bayesian Learning algorithm with the SPGL1 at different noise levels. In this experiment the sensing matrix is the same as used in experiment 1, M = 128 and N = 256. The same Gaussian white noise is added for both algorithms so that the SNR, defined by SNR (dB), 20 log (||Ax||/ ||v||), ranging the signal from 5 dB to 25dB for the generated signal. The results are shown in Fig 2 the Bayesian Learning algorithm exhibited significant performance over SPGL1.

As shown in the figure the Normalized mean square error of the reconstructed signal decays faster in the Bayesian Learning case than that of the SPGL1. At all noise cases the Bayesian Learning shows less error than the SPGL1 which makes the Bayesian learning method more

tolerant to noise than the SPGL1.

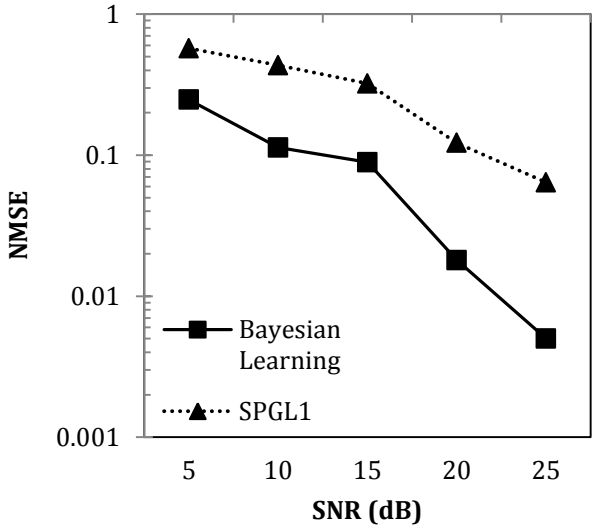

Figure2: Noise tolerance test between Bayesian Learning and SPGL1

3.3 Experiment 3: Results of multiple Channel Recovery 2D EEG data.

In this experiment 5 channels of EEG signals was compressed and reconstructed in only 7 iterations using Bayesian Learning.

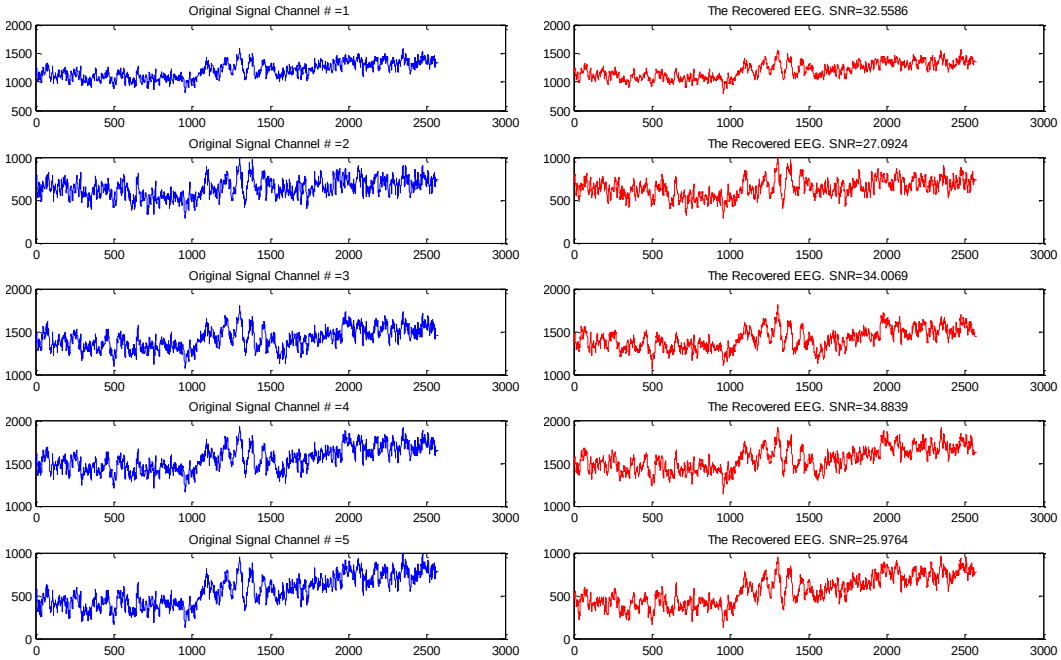

Figure3: Multiple channels Reconstruction using Bayesian Learning of Average 32 dB SNR

The above figure shows the original signal and the reconstructed data of the compression associated with its SNR. The Original data is 2560 samples per channel sampled at 1000Hz. The reconstruction time taken for reconstructing these 5 channels is 9.95 seconds in 7 iterations which was executed on the same machine mentioned in experiment one. On the other hand the total time taken to reconstruct each channel separately is on average 12.25 seconds. In this experiment the sensing matrix is the same in experiment one, compression

rate is 50 %, and the dictionary matrix is a DCT matrix. At the best of our knowledge figure 3 illustrates the best CS reconstruction results was ever illustrated to EEG signals. The result of average NMSE and SSIM are 0.0007 and 0.9876 respectively and these results are the best results ever published so far.

4 Discussion:

CS resorts to a dictionary matrix for non-sparse signal recovery. But the success of this approach relies on the sparsity of its coefficients and the dictionary matrix. To find a dictionary matrix in which a signal can be sparse is an important issue and it is a more complex problem when the signal is non-sparse by its nature neither in its transform. We proved that using various popular general dictionary matrices, the transform representation of such a non-sparse signal like EEG signals are still not sparse. We showed that one of the state of the art CS algorithms SPGL1 has poor performance and their recovery quality is not suitable. Instead of seeking optimal dictionary matrices, this study proves a method using general dictionary matrices but achieves good recovery quality for non-sparse signals. The empirical results show that when using the Bayesian learning for non-sparse signals for compression/recovery the seeking of optimal dictionary matrices is not very crucial issue. We suggest a research direction to improve the proposed Bayesian learning algorithm to achieve better performance in terms of learning speed.

5 Conclusion:

In applications of which large data volumes is acquired and given hardware, memory and processing power limitations, high compression rates without trading off quality is an important matter. Compressed Sensing solves this problem, but the problem is extremely difficult in current algorithms for non-sparse signals in the time domain and in the transformed domains. To improve the problem, this study shows that the use of block sparse Bayesian learning has superior performance to other existing state of the art CS algorithms in recovering non-sparse signals. Experimental results show that it recovered EEG signals with good quality. It is suggested to improve this algorithm by applying Bayesian optimization techniques to improve speed.

References

[1] Benjamin Blankertz, Guido Dornhege, Matthias Krauledat, Klaus-Robert Müller, and Gabriel Curio. The non-invasive Berlin Brain-Computer Interface: Fast acquisition of effective performance in untrained subjects. NeuroImage, 37(2):539-550, 2007

[2] Felix J. Herrmann: Randomized sampling and sparsity: getting more information from fewer samples. Geophysics 75, WB173 (2010); doi:10.1190/1.350614

[3] Felix J. Herrmann, Michael P. Friedlander, Ozgur Yilmaz: Fighting the curse of dimensionality: compressive sensing in exploration seismology

[4] Mallat, S. G., 2009, A Wavelet Tour of Signal Processing: the Sparse Way: Academic Press

[5] Hennenfent, G., and F. J. Herrmann, 2008, Simply denoise: wavefield reconstruction via jittered undersampling: Geophysics, 73.

[6] Herrmann, F., D. Wang, G. Hennenfent, and P. Moghaddam, 2008a, Curvelet-based seismic data processing: a multiscale and nonlinear approach: Geophysics, 73, A1–A5. (doi:10.1190/1.2799517).

[7] Extension of SBL Algorithms for the Recovery of Block Sparse Signals with Intra-Block Correlation Zhilin Zhang, Bhaskar D. Rao IEEE Trans. on Signal Processing, vol. 61, no. 8, pp. 2009-2015, 2013

[8] Sparse Signal Recovery with Temporally Correlated Source Vectors Using Sparse Bayesian Learning Zhilin Zhang, Bhaskar D. Rao IEEE Journal of Selected Topics in Signal Processing, vol.5, no. 5, pp. 912-926, 2011

[9] Z. Wang and A. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," IEEE Signal Processing Magazine, vol. 26, no. 1, pp. 98–117, 2009.

[10] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," J. of Mach. Learn. Res., vol. 1, pp. 211–244, 2001.