Study of Compressive Sensing Techniques Using Spike Detection

Pratikkumar N Patel
Graduate Student
Department of Computer Science
Boston University
Boston, MA 02215
pratikp@bu.edu

Aswin Vasudevan
Graduate Student
Department of Computer Science
Boston University
Boston, MA 02215
aswinv@bu.edu

Jacob Levy
Under Graduate Student
Department of Computer Science
Boston University
Boston, MA 02215
levyjr@bu.edu

Abstract

We have written this paper as part of our project of Compressive Sensing in Big Data subject. Here we have studied use of CS technics that can be used in developing better tools (software and hardware) in studies of electrophysiological data. We explain a brief about problems in these experiments, how CS can help, what algorithms we've tried, our results and observation and some of the future work required in this field.

1 Background

The dataset we are using originates from a collection of electrophysiological data received from electrode implants in the human brain. The implant in the brain collects data and executes basic processing to allow the device to run properly. The electrode implant sends a wireless signal to a receiver outside the human body. This receiver receives the data from the implant and executes a post-processing procedure on the data collected from the electrode. It is in this external receiver that detection of the neuron spikes and sorting of the dataset takes place.

1.1 Why do we need to understand spikes?

Some benefits of this electrophysiological technique are that it can help neuroscientists understand and investigate the the inner-workings of the nervous system and allow them to retrieve real-time motor commands from different parts of the body.

1.2 Problems of Spike Detection

Many times neuroscientists desire to conduct multiple experiments on a subject in order to analyze a large dataset. However, the electrode implant tends to overheat when used continuously in an experiment. This overheating can potentially cause brain damage and cause harm to the implant. An additional problem with spike detection is with the technological limitations of the implant itself. The electrode implant generally has low processing power, a limited battery life (in the case of a wireless

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

system), and a large bandwidth requirement to send signals between the implant and the receiver. However, all of these technological setbacks can be solved with the help of compressive sensing.

1.3 Compressive Sensing to the Rescue

By implementing a compressive sensing algorithm in the external receiver we can reduce the experiment time and reduce the risk of overheating because we are taking only sampling a fraction of the actual data. Compressive sensing can help solve the technological problems associated with spike detection by saving power in the implant and reducing the bandwidth requirement between the implant and the receiver by offloading the processing to the receiver.

2 Data Preprocessing

The raw signal received from the implant is extremely noisy and difficult to distinguish. This noise originates from spikes from far away neurons and other random electrical noises. However, compressive sensing can only operate correctly if the signal is sparse. A sparse signal is necessary in order to have a favorable recovery. To create this sparse signal we must apply a fir1 bandpass filter, detect spikes to make the signal sparse, and remove the excess noise by keeping the detected spikes.

2.1 Filtering

Our results on plotting the human data revealed us that the signals are dense and random. This randomness have a wave like pattern over the graph. The studies on application of compressive sensing algorithm showed us that by smoothing the signal variations we were able to achieve better results. The signal consist of mainly two entities, the spikes and the noise associated with the spikes.

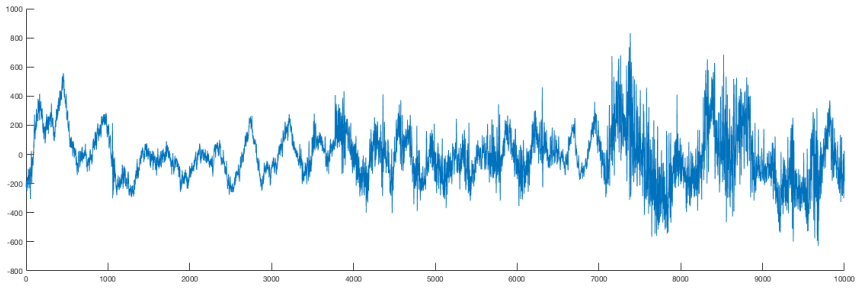


Figure 1: Raw Signal

The filtering process should enable us to bring in a threshold frequency as a cut off limit and collect the signals that are related to the spikes by high pass filtering. We have made use of standard kaiser window based filtering and FIR filtering techniques. The filtering process have direct impact over the sparsity as it it removes a lot signals thereby enabling low measurements of initial input to the compressive sensing algorithm. There are oscillation of signal between the passband and stopband. Our case study holds spikes both at the global maximum and local maximum. The classification of these maximum levels are input dependent but effective to be taken into consideration. The use of Kaiser Window will locate the spikes and stopband the noise that are at local maxima and global maxima.The kaiser window is defined by:

$$\omega[n] = \begin{cases} \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{N-1}\right)^2}\right)}{I_0(\pi\alpha)}, & 0 \leq n \leq N-1 \\ 0 & otherwise \end{cases} \tag{1}$$

The parameters that are used in the window function are:

- N length of the signal

- I_0 is the zeroth-order of bessel function
- α is a non-negative real number that determines the shape of the window

We have referenced Arya et al. (2015) for filtering details.

Kaiser Window's width of the main lobe is inversely proportional to the length of the filter. The attenuation in the side lobe is, however, independent of the length and is function of the type of the window. Therefore the length of the filter must be increased considerably to reduce the main lobe width and to achieve the desired transition band.

We also used FIR1 filter of matlab using hamming window to understand the nature of modification that a filter makes to the signal.The hamming window is optimized to minimize the maximum (nearest) side lobe, giving it a height of about one-fifth that of the Hann window. The hamming window function is defined by:

$$\omega(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right) \tag{2}$$

The constants of the hamming window function are selected at ideal level of 0.54 and 0.46. The high pass filtering by hamming window also rendered us the results that were close to the results that are produced by the kaiser window.

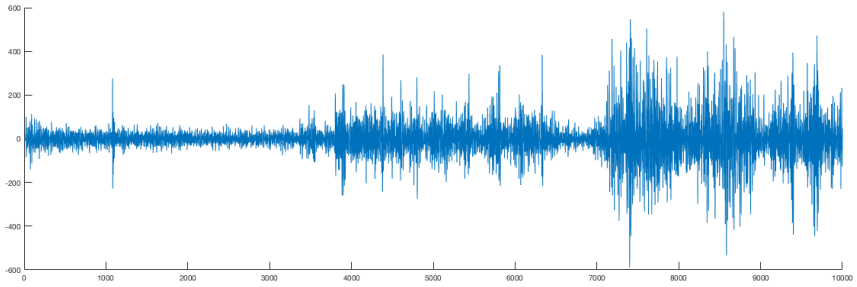


Figure 2: Filtered Signal

3 Spike Detection

We have referenced Gaidica (2015) to understand problems in spike detection and how to solve them. Below, we are explaining how his implementation of spike detection works.

Simplest method of doing spike detection is to draw a threshold on the signal and keep spikes and ignore the noise below the threshold. One major problem with this method is that it can catch a lot of noise even though the threshold is being selected very carefully. Unless the signal is very carefully collected, avoiding noise, it is very difficult to find spikes with just the threshold.

As described by Kim et al. (2003), Nonlinear Energy Operator is a very good way of finding and separating spikes signals from surrounding data. In this technique, we square the signal to increase the amplitude and subtract the neighboring signal values.

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1). \tag{3}$$

Where ψ is nonlinear energy

As we can see in above equation, this will make sure high amplitude data stays higher while reducing the noise from the signal especially around the spikes. Threshold is being calculated using the average of the median values we get from smoothed nonlinear energy. Once we have the threshold, we can simply use `peekseak.m` to find the locations of all the spikes in the signal.

These detected spike location information will be used later to create a sparse signal that we need for compressive sensing. We'll explain sparsification of the signal in next section.

4 Why do we need compressive sensing?

Compressive sensing theory was developed upon the idea that many signals can be represented using only a few non-zero coefficients in a suitable basis or dictionary.

Please refer Davenport et al. (2011) for more details.

CS can give us mathematical guarantees on accuracy of recovered signals from very few measurements provided the original signal was sparse. Compressive sensing can help us wherever it is very costly or physically impossible to sample many measurements. Like in our case, sampling all spikes and transmitting over to receiver end is a power hungry operation and can burn the electrodes and surrounding tissues.

If we look at our raw data from the electrode, it doesn't look sparse at all. But after a little preprocessing mentioned above it looks manageable (Figure 2) and can be converted to a sparse signal using spike detection.

Above signal is what we have after passing it through filter designed above. This signal can be passed through spike detection to find out locations of spikes in the signal. Please note above image shows just 10k samples of a signal that was sampled at 20kHz frequency.

Once we have spike locations, a new signal can be created using the data from window at each spike location. Rest of the new signal will not have any data. The new signal after keeping the spike data will look as shown in Figure 3.

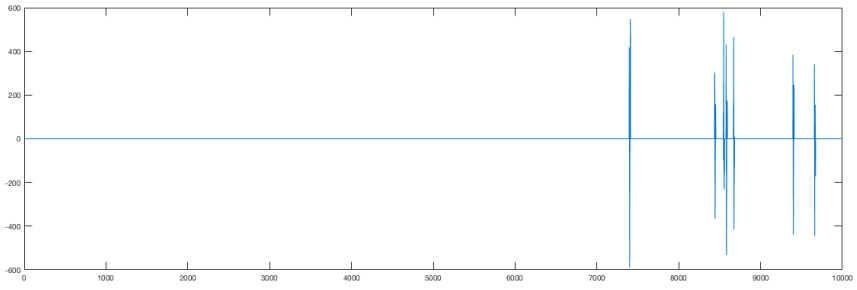


Figure 3: Spike Detected and Cleaned

To see how this looks, we have zoomed in on individual spikes as shown in figure 4 and 5.

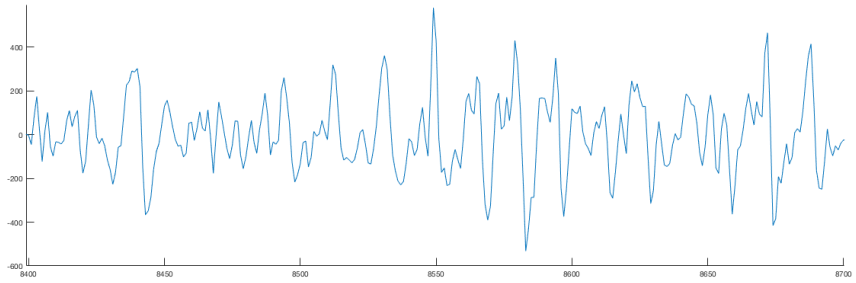


Figure 4: Filtered Signal Zoomed In

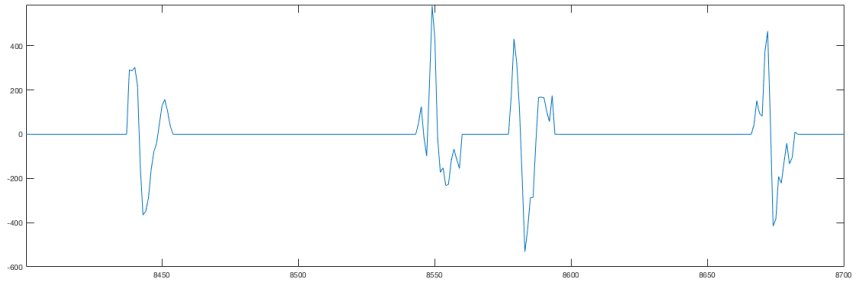


Figure 5: Cleaned Signal Zoomed In

5 Compressive sensing techniques

In our project, we have used below mentioned algorithms to recover sparse signals. Here we are explaining all algorithms in brief. For more information, please follow references. Results for all algorithms are mentioned in following section.

5.1 L0-minimization Algorithms

5.1.1 Subspace Pursuit

Please refer Dai et al. (2009) for more details.

The main intention behind using the subspace pursuit is that the algorithm has low reconstruction complexity for sparse signals. On removing the noise data we ensure that the signal contains only the spikes there by sparsifying the signal. The algorithm also produces results in noisy regimes.

- The algorithm takes K columns from Sensing matrix A and spans the candidate subspace.
- If the result produced is large then the algorithm increments and adds new basis vector.
- The searching of vectors is maintained at constant rate.
- When the resultant do not lie in the current estimate then refines reliable candidate list.
- The reliability is based on order statistics of inner product between signal and A.

The complexity of the SP algorithm is upper bounded by $O(mNK)$.

5.1.2 Orthogonal Matching Pursuit

Input:

- The input will be the cleaned signal with spikes (b).

- The Sensing Matrix formed by sampling methods(A).

Algorithm:

- Initialize the residual, time and index set to Null.
- The index vector is take at i units as per the input
- The residual and row vector of A are compared individually and the Max value if a_i.
- Update $V_t = V_{t-1} \bigcup \{v_t\}$
- Solve the least-squares problem

$$\min_{c \in \mathbb{C}^{V_t}} ||b - \sum_{j=1}^{t} c(v_j) a_{v_j}||_2 \tag{4}$$

- Now the new residual will be

$$r_t = r_{t-1} - \sum_{j=1}^{t} c(v_j) a_{v_j} \tag{5}$$

- Increment time until stopping criteria

5.2 L1-minimization Algorithms

5.2.1 ALM

The Augmented lagrangian methods are the set of algorithm to solve constrained optimization problems. It can be visualized as method of multipliers for recovery. In the sparse cleaned signal that consist of spikes is fed as input to the algorithm along with the sensing matrix. The penalty method used solves the problem by :

$$\min \Phi_k(x) = f(x) + \frac{\mu_k}{2} \sum_{i \in I} c_i(x)^2 - \sum_{i \in I} \lambda_i c_i(x) \tag{6}$$

The Alm algorithm in our implementation converges the results that are obtained and if the candidate set is large then it again implements the minimization by having larger Mu value. The Parameter for each iteration is incremented at the steps given by :

$$\lambda_i \leftarrow \lambda_i - \mu_k c_i(x_k) \tag{7}$$

5.2.2 FISTA

Please refer Beck et al. (2009) for more details.

This method takes advantage of Iterative shrinkage thresholding method by applying along with the global rate of convergence. The basic idea is to solve the computational intensity of the input signal by using cheap matrix operation on sensing matrix. The operation of ISTA is given by :

$$x_{k+1} = \tau_{\lambda t_k} \left(x_k - 2 t_k A^T \left(A x_k - b \right) \right) \tag{8}$$

The shrinkage step involved is given by the equation:

$$\tau_\alpha(x)_i = (|x_i| - \alpha) + sgn(x_i) \tag{9}$$

But on every step of ISTA involves gradient step to smooth and shrinkage operation thus the algorithm is considered to be slow method. Thus its modified by function values to fasten the process as FISTA given by:

$$\min_x \{F(x) \equiv f(x) + g(x)\} \tag{10}$$

where f and g are convex functions. This algorithm is shown to converge to the optimal function value with the faster, rate of O(1/k2), k being the iteration number. So the overall working of FISTA

is given by
FISTA with constant step size
Input: $L = L(f)$ - A Lipschitz constant of ∇f

- Step 0 : Take $y_1 = x_0 \in \mathbb{R}^n, t_1 = 1$
- Step k : $k \geq 1 compute$

$$x_k = \text{Prox}_{t_k}(g)\left(y_k - \frac{1}{L}\nabla f(y_k)\right) \tag{11}$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \tag{12}$$

$$y_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1}) \tag{13}$$

The usage of FISTA differs from that of ISTA only at the shrinkage step. The point selected for iterative shrinkage in FISTA is specific linear combination of the previous two points x_{k-1}, x_k.

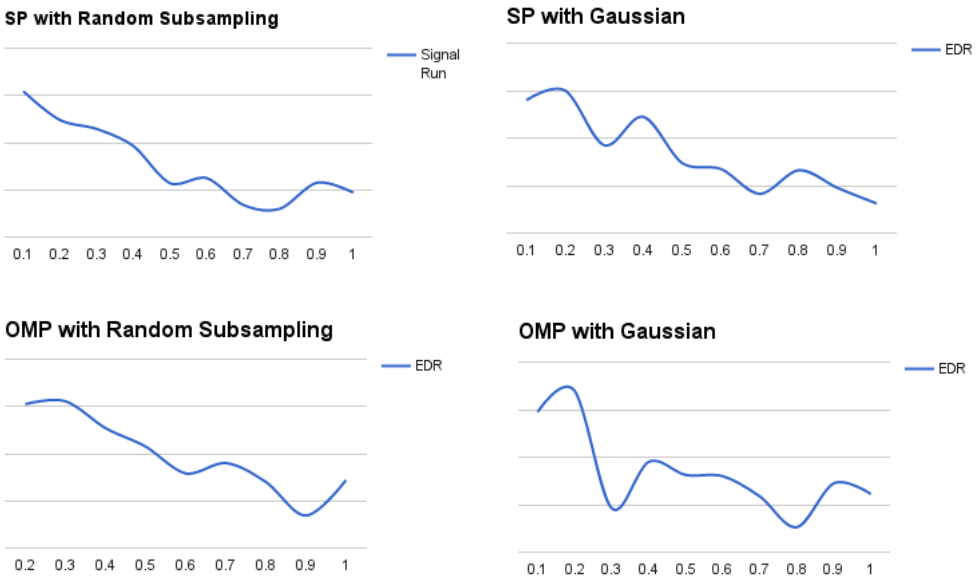
5.3 Sampling Methods

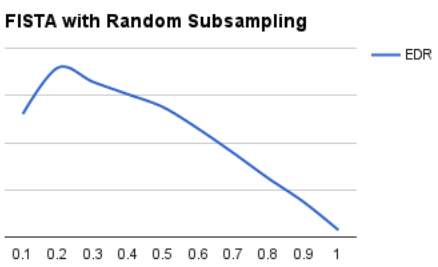
The sampling methods we have used for construction of sensing matrix are:

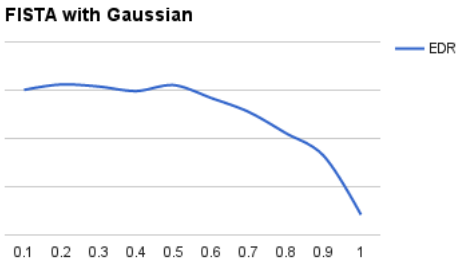
- Random SubSampling : It is achieved by using DCT on the signal and randomly permute them to form sensing matrix.
- Random Gaussian Sampling : Here the randomness is achieved by permutation on the signal using Matrix norm.

6 Results

We have used EDR (edit distance) between original signal spikes and recovered signal spikes to find if a method is giving us good results or not. Below given graphs are showing our results with different techniques over different values of M.







7 Conclusion

We noticed that FISTA algorithm was the fastest and giving consistently good results. We also noticed that Random Subsampling Matrix was giving good results as a sampling method. Gaussian was unpredictable and did not give good results even when values of M were increased to sampling 0.8 of the values.

8 Future Work

We have started working on clustering the spikes using GMM unsupervised learning technique to design a sparsifying matrix that can be used during sampling process and recovery. It has been worked on by some other researchers and proven to be good techniques.

Acknowledgments

Prof Peter Chin has been an advisor during this project and his input has been invaluable. 3 members of our team had distributed project work shown below.
Aswin has been responsible for Code for Filtering and execution of L0-minimization algorithms, preparation of presentation slides and write-up for the report (Filtering, Algorithm Details and sampling methods).
Jacob has been responsible for preparation of presentation slides, write-up for the report and collection and visualization of result data.
Pratik has been responsible for code for Spike Detection Algorithm and execution of L1-minimization algorithms preparation of presentation slides, write-up and formatting for the report.

References

Rachna Arya and Shiva Jaiswal
Design of Low pass FIR Filters using Kaiser Window Function with variable parameter Beta
International Journal of Multidisciplinary and Current Research, March/April 2015
http://ijmcr.com/wp-content/uploads/2015/03/Paper6220-224.pdf.

Matt Gaidica
EXTRACTING SPIKES FROM NEURAL ELECTROPHYSIOLOGY IN MATLAB
http://gaidi.ca, November 2, 2015
http://gaidi.ca/weblog/extracting-spikes-from-neural-electrophysiology-in-matlab.

Kim and Kim (2003).

Mark A. Davenport, Marco F. Duarte, Yonina C. Eldar, Gitta Kutyniok
Introduction to Compressed Sensing
http://statweb.stanford.edu/ markad/publications/ddek-chapter1-2011.pdf.

Wei Dai and Olgica Milenkovic
Subspace Pursuit for Compressive Sensing Signal Reconstruction
https://arxiv.org/pdf/0803.0811.pdf.

Amir Beck and Marc Teboulle
A FAST ITERATIVE SHRINKAGE-THRESHOLDING ALGORITHM WITH APPLICATION TO WAVELET-BASED IMAGE DEBLURRING
https://pdfs.semanticscholar.org/2584/b154a1f7743438119c7cd8ed56f556e7abe9.pdf.