Compressive Sampling for Energy Efficient Event Detection

Zainul Charbiwala, Younghun Kim, Sadaf Zahedi, Jonathan Friedman, and Mani B. Srivastava
Event Detection in Wireless Sensor Networks

\[ R_{ij} \]
When noise is non-Gaussian:
Event Detection in Wireless Sensor Networks

When noise is non-Gaussian:
- Need to detect some unique feature of the signal
When noise is non-Gaussian:

- Need to detect some unique feature of the signal
- Computing features locally is expensive, so is transmitting raw data
Event Detection in Wireless Sensor Networks

When noise is non-Gaussian:

- Need to detect some unique feature of the signal
- Computing features locally is expensive, so is transmitting raw data
- Feature computation = compression in the feature space
Compressive Sampling
Compressive Sampling

Physical Signal -> Sampling -> Compression -> Communication -> Detection

Shifts computation to fusion center

Physical Signal -> Compressive Sampling -> Communication -> Decoding -> Detection
Compressive Sampling

\[ x \in \mathbb{R}^n \quad \Rightarrow \quad z = \Phi x \]

\[ x \in \mathbb{R}^n \quad \Rightarrow \quad z = \mathbf{I}_n x \]
Compressive Sampling

\[ x \in \mathbb{R}^n \]

Physical Signal

Sampling

Compressive Sampling

Compression

Compressed domain samples

\[ z = I_n x \]

\[ y = \Psi z \]

Decoding

Compressed domain samples

\[ y = \arg \min_{\tilde{y}} \| \tilde{y} \|_{\ell_1} \]

\[ \text{s.t. } z = \Phi \Psi^{-1} \tilde{y} \]
Compressive Sampling

How do we generate $\Phi$ and compute incoherent samples?

$\mathbf{x} \in \mathbb{R}^n$

$\mathbf{z} = \mathbf{I}_n \mathbf{x}$

$\mathbf{y} = \Psi \mathbf{z}$

$\mathbf{y} = \arg \min_{\tilde{\mathbf{y}}} \| \tilde{\mathbf{y}} \|_{\ell_1}$

$s.t. \quad \mathbf{z} = \Phi \Psi^{-1} \tilde{\mathbf{y}}$
Taking Incoherent Projections

- Gaussian: independent realizations of $\mathcal{N}(0, \frac{1}{n})$
Taking Incoherent Projections

- Gaussian: independent realizations of $\mathcal{N}(0, \frac{1}{n})$

\[ z = \Phi x \]

- Needs a priori Nyquist sampling and storage for $x$ and $\Phi$

- Needs $kn^2$ FPU multiplications, $k(n-1)$ additions and $kn^2$ memory accesses
Taking Incoherent Projections

- Realizations of equiprobable Bernoulli RV $\left\{ \frac{+1}{\sqrt{n}}, \frac{-1}{\sqrt{n}} \right\}$

\[ z = \Phi \]

\[ x \]
Taking Incoherent Projections

- Realizations of equiprobable Bernoulli RV \( \left\{ \frac{+1}{\sqrt{n}}, \frac{-1}{\sqrt{n}} \right\} \)

\[ z = \Phi x \]

- Needs a priori Nyquist sampling and storage for \( x \) and \( \Phi \)

- Needs \( k(n-1) \) additions and \( kn^2 \) memory accesses but no multiplications
Taking Incoherent Projections

- Uniform Random Sampling

\[ z = \Phi x \]
Taking Incoherent Projections

- Uniform Random Sampling

\[ z = \Phi x \]

- Needs a priori Nyquist sampling & storage for \( x \) but \( \Phi \) can be generated on-the-fly

- Needs \( k \) memory accesses but no multiplications or additions

- Need higher \( k \) because matrix is sparse
Taking Incoherent Projections

- Causal Uniform Random Sampling (sorted $\Phi$ from URS)

\[ z = \Phi \]

\[ \chi \]
Taking Incoherent Projections

- Causal Uniform Random Sampling (sorted $\Phi$ from URS)

- Need to generate, **sort** and store $\Phi$ (but no a priori sampling or storage for $x$)

- Needs $k$ memory accesses but no multiplications or additions

- May violate ADC hold time
Causal Uniform Random Sampling on-the-fly

- Use an additive random process

\[
\begin{align*}
  t_0 &= 0 \\
  t_{i+1} &= t_i + \tau_i \\
  \tau_i &\sim \mathcal{N}(\mu, r^2 \mu^2)
\end{align*}
\]
Causal Uniform Random Sampling on-the-fly

- Use an additive random process

\[ t_0 = 0 \]
\[ t_{i+1} = t_i + \tau_i \]
\[ \tau_i \sim \mathcal{N}(\mu, r^2 \mu^2) \]
Causal Uniform Random Sampling on-the-fly

- Use an additive random process

\[ t_0 = 0 \]
\[ t_{i+1} = t_i + \tau_i \]
\[ \tau_i \sim \mathcal{N}(\mu, r^2 \mu^2) \]

- Need to truncate the negative tail of Gaussian PDF to ensure causality and feasibility
Causal Uniform Random Sampling on-the-fly

- Use an additive random process

\[ z = \Phi \]
Causal Uniform Random Sampling on-the-fly

- Use an additive random process

\[ z = \Phi \]

- Can generate \( \Phi \) on-the-fly and do not need to sample or store \( x \) a priori

- Needs \( k \) memory accesses but no multiplications or additions
Modifying CS for Detection
Modifying CS for Detection

- CS is geared toward sparse signal reconstruction
- What if we wanted to detect a signal feature instead?
Modifying CS for Detection

- CS is geared toward sparse signal reconstruction
- What if we wanted to detect a signal feature instead?

\[ y = \arg \min_{\tilde{y}} \| \tilde{y} \|_{\ell_1} \quad \text{s.t.} \quad z = \Phi \Psi^{-1} \tilde{y} \]

\[ y = \arg \min_{\tilde{y}} \| W \tilde{y} \|_{\ell_1} \quad \text{s.t.} \quad z = \Phi \Psi^{-1} \tilde{y} \]

- Proposed solution: Weighted Basis Pursuit
  - Intuitive idea: Reduce penalty on features of interest
Example: Detecting 450Hz Tone

(a) FFT of original 450Hz tone at -10dB SNR

(b) BP recovery with 300Hz sampling

(c) BP recovery with 30Hz sampling

(d) Weighted BP recovery with 30Hz sampling

The reconstruction is performed in the Fourier (frequency) domain from randomly collected samples at different rates. When no weighting is applied, the average sampling rate needs to be as high as 300Hz to detect the tone – the red dot in Figure 1.4b is just above the noise floor. While this is below the Nyquist rate of 900Hz, the gains are not impressive. If the sampling rate is lowered to 30Hz, no detection is possible (1.4c). However, if weighting is applied, the frequency tones immediately stand out (1.4d), implying a near $30 \times$ benefit over the Nyquist rate. A detailed evaluation of both simulated and experimental performance for different sampling rates in various noisy environments is deferred until Section 3.4.
Evaluation Process

Generate Tone+Noise

Sensing Mote

BASE Station

Collect Samples

Weighted Basis Pursuit

Tone Detection

$z = \Phi x + \eta'$

$\hat{y}$

$\mathcal{H}$

simulate Random Projections

$X + \eta$

Acoustic Tone Detection
Per-Module Energy Consumption

- FFT computation higher than transmission cost
- Highest consumer in CS is the random number generator
Detection Performance - Simulation

- Detection near perfect above 0dB with 30Hz sampling rate
- Improvement in performance due to weighting at high SNR
Detection Performance - Experiments

- Some deviation from simulation results due to ambient conditions
Detection Performance - Interference

- Weighting does not help much against narrowband interference

-30dB SINR
-20dB SINR
-10dB SINR

Sampling Rate (Hz)

P_e

0.5
0.4
0.3
0.2
0.1
0
10
20
30
50
100
10
20
30
50
100
10
20
30
50
100

BP
WBP
Related Work

- Compressive Sensing
  - [Donoho98], [Candes06], [Candes08], etc.

- Compressive Detection
  - [Duarte06], [Khajehnejad09]

- Random Sampling
  - [Bilinskis92], [Boyle07], [Dang08], etc.
Summary

- Prudence at the sampling stage can deliver energy efficiency gains through the entire sensing chain.
- Computing random projections on the fly is feasible by paying an extra cost in number of measurements.
- Good pseudo-random number generation has non-trivial computation cost.
- Detection performance with CS at par with Nyquist rate with a 30x rate reduction.
Cost/Performance of RNGs

- Linear Feedback Shift Register
- Multiplicative Linear Congruential Generator
- Mersenne-Twister $2^{19937}-1$
Cost/Performance of RNGs

- Linear Feedback Shift Register
- Multiplicative Linear Congruential Generator
- Mersenne-Twister \(2^{19937} - 1\)

- RIP constant indicates CS performance (lower is better)
- MLCG a little worse than MT
- Variation with LFSR very high
Thank You!

zainul@ee.ucla.edu


Slides: http://nesl.ee.ucla.edu/document/show/302
Geometric Interpretation of WBP

2.2 An iterative algorithm

The question remains of how a valid set of weights may be obtained without first knowing $x_0$. As Figure 1 shows, there may exist a range of favorable weighting matrices $W$ for each fixed $x_0$, which suggests the possibility of constructing a favorable set of weights based solely on an approximation $x$ to $x_0$ or on other side information about the vector magnitudes.

We propose a simple iterative algorithm that alternates between estimating $x_0$ and redefining the weights. The algorithm is as follows:

1. Set the iteration count $\ell$ to zero and $w^{(0)}_i = 1, i = 1, \ldots, n$.
2. Solve the weighted $\ell_1$ minimization problem
   
   $x^{(\ell)} = \text{arg min} \|W^{(\ell)} x\|_{\ell_1}$
   
   subject to $y = \Phi x$. (2.4)
3. Update the weights: for each $i = 1, \ldots, n$,
   
   $w^{(\ell+1)}_i = 1/|x^{(\ell)}_i| + \epsilon$. (2.5)
4. Terminate on convergence or when $\ell$ attains a specified maximum number of iterations $\ell_{\text{max}}$. Otherwise, increment $\ell$ and go to step 2.

We introduce the parameter $\epsilon > 0$ in step 3 in order to provide stability and to ensure that a zero-valued component in $x^{(\ell)}$ does not strictly prohibit a nonzero estimate at the next step. As empirically demonstrated in Section 3, $\epsilon$ should be set slightly smaller than the expected nonzero magnitudes of $x_0$.

In general, the recovery process tends to be reasonably robust to the choice of $\epsilon$.