On Duality, Encryption, Sampling and Learning: the power of codes

Kannan Ramchandran
University of California, Berkeley
Shannon’s incredible legacy

- A mathematical theory of communication
- Channel capacity
- Source coding
- Channel coding
- Cryptography
- Sampling theory
- …

(1916-2001)
And many more...

- Boolean logic for switching circuits (MS thesis 1937)

- Juggling theorem:
  \[ H (F+D) = N (V+D) \]

  - \( F \): the time a ball spends in the air,
  - \( D \): the time a ball spends in a hand,
  - \( V \): the time a hand is vacant,
  - \( N \): the number of balls juggled,
  - \( H \): the number of hands.

- ...

(1916-2001)
Story: Shannon meets Einstein

*As narrated by Arthur Lewbel (2001)*

“
The story is that Claude was in the middle of giving a lecture to mathematicians in Princeton, when the door in the back of the room opens, and in walks Albert Einstein.

Einstein stands listening for a few minutes, whispers something in the ear of someone in the back of the room, and leaves. At the end of the lecture, Claude hurries to the back of the room to find the person that Einstein had whispered too, to find out what the great man had to say about his work.

The answer: Einstein had asked directions to the men’s room.
”
Outline

Five “personal” Shannon-inspired research threads:

**Chapter 1:** **Duality** between source coding and channel coding – with side-information (2003)

**Chapter 2:** **Encryption** and **Compression** – swapping the order (2003)

**Chapter 3:** **Sampling** below Nyquist rate and efficient reconstruction (2014)

**Chapter 4:** **Learning** and inference exploiting sparsity – sub-linear time algorithms (2015-Present)

**Chapter 5:** **Codes** for distributed computing & machine learning (2017-Present)
Chapter 1

Duality

- source & channel coding
- with side-information
Shannon’s celebrated 1948 paper

The Bell System Technical Journal

Vol. XXVII  
July, 1948

No. 3

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

The recent development of various methods of modulation such as DSB and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist1 and Hartley2 on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one selected from a set of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonically increasing function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance


Fig. 1—Schematic diagram of a general communication system.
Source coding

Entropy of a random variable
= minimum number of bits required to represent the source

\[ H(X) = \mathbb{E}_X \left[ \log \left( \frac{1}{p(X)} \right) \right] \]
Rate-distortion theory - 1948

- Trade-off between *compression rate* and the *distortion*

**PART V: THE RATE FOR A CONTINUOUS SOURCE**

27. FIDELITY EVALUATION FUNCTIONS

In the case of a discrete source of information we were able to determine a definite rate of generating information, namely the entropy of the underlying stochastic process. With a continuous source the situation is considerably more involved. In the first place a continuously variable quantity can assume an infinite number of values and requires, therefore, an infinite number of binary digits for exact specification. This means that to transmit the output of a continuous source with *exact recovery* at the receiving point requires, in general, a channel of infinite capacity (in bits per second). Since, ordinarily, channels have a certain amount of noise, and therefore a finite capacity, exact transmission is impossible.

This, however, evades the real issue. Practically, we are not interested in exact transmission when we have a continuous source, but only in transmission to within a certain tolerance. The question is, can we assign a definite rate to a continuous source when we require only a certain fidelity of recovery, measured in a suitable way. Of course, as the fidelity require-

\[
R(D) = \min_{P_{Y|X}(y|x)} I(X;Y)
\]

subject to \( \mathbb{E}[d(X,Y)] \leq D \)

Mutual information:
\[
\mathcal{H}(X) - \mathcal{H}(X|Y)
\]

distortion measure
Channel coding

\[ C(W) = \max_{P_X(x)} I(X;Y) \]
subject to \( \mathbb{E}[w(X)] \leq W \)

- For rates \( R < C \), can achieve arbitrary small error probabilities
- Used to be thought one needs \( R \rightarrow 0 \)
Shannon’s breakthrough

• Communication before Shannon:
  – *Linear filtering* (Wiener) at receiver to remove noise

• Communication after Shannon:
  – Designing codebooks
  – *Non-linear estimation* (MLE) at receiver

*Reliable transmission at rates approaching channel capacity*
“There is a curious and provocative duality between the properties of a source with a distortion measure and those of a channel. This duality is enhanced if we consider channels in which there is a cost associated with the different input letters, and it is desired to find the capacity subject to the constraint that the expected cost not exceed a certain quantity.....
Shannon (1959)

...This duality can be pursued further and is related to a duality between past and future and the notions of control and knowledge. Thus, we may have knowledge of the past but cannot control it; we may control the future but not have knowledge of it.”
Functional duality

When is the *optimal encoder* for one problem functionally identical to the *optimal decoder* for the dual problem?
Duality example: Channel coding

You want to send message $m$: how big can you make $R$?

\[
\begin{align*}
\hat{X} & \xrightarrow{1 - p} 0 \\
& \xrightarrow{p} \ast \\
1 & \xrightarrow{1 - p} 1
\end{align*}
\]

\[C_{\text{BEC}} = (1 - p) \text{ bits per channel use}\]

$p = 0.2$

\begin{align*}
\text{Cost (0)} &= 1; \\
\text{Cost (1)} &= 1
\end{align*}

Total budget $\leq 10,000$
What is the Shannon capacity?

Encoder:
- Input: $m$
- Output: $\hat{m}$

Decoder:
- Input: $\hat{m}$
- Output: $\hat{m}$

**Surprise:** the encoder does not need to know which bits are erased!

The decoder knows which bits are erased (channel output).

Suppose the encoder also knows which bits are erased (genie).

- Number of non-erased bits:
  \[ \approx 10,000 \times (1 - p) \]
  \[ = 10,000 \times 0.8 = 8,000 \]

The Shannon capacity is $0.8$ bits/ch. use with a BEC of $0.8$.

Send information in non-erased locations.
Shannon’s prescription: random coding

1) **Encoder & Decoder agree on a random codebook**
   - Shannon’s random coding argument

2) **Encoder encodes message**
   - *Output the codeword corresponding to the index*

3) **Decoder decodes message**
   - *Output the index corresponding to the closest codeword*
Why does it work?

- Decoding successful if the non-erased string is unique
- \( \text{Pr.\{not unique\}} \leq 2^{-n(1-p)} \times 2^{nR} \Rightarrow 0 \) if \( R \leq (1 - p) \)
- 8,000 bits will induce unique match if (random) codebook size is \( \leq 2^{8,000} \) w.h.p.

\[ \begin{align*}
\text{Codebook for channel coding} \\
\begin{array}{c}
1001000010101000...
111011111101110...
1110000111001110...
1101011001001010...
\end{array}
\end{align*} \]
Source Coding Dual to the BEC: BEQ

\( X \in \{0,1\}^{10,000} \)

01*1*00110...

Source Encoder \( m \)

Compressed bit-stream 8,000 bits

Source Decoder \( \hat{X} \)

Want the average distortion to be \( \leq 0.2 \)

\[
d(x, \hat{x}) = \begin{cases} 
0 & \text{if } \hat{x} = x \text{ for } x \in \{0, 1\} \\
\infty & \text{if } \hat{x} \neq x \text{ for } x \in \{0, 1\} \\
1 & \text{if } x = * 
\end{cases}
\]

\( p(0) = p(1) = 0.4; \quad p(*) = 0.2 \)

\( x: 1 \quad 0 \quad * \quad * \quad 0 \quad 1 \)

\( \hat{x}: 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \)

cost: 0 \quad 1 \quad \infty

* is like a “don’t care” symbol (e.g., perceptually masked symbols). How can we exploit this for compression?

Martinian and Yedidia, 2004
Source Coding Dual to the BEC: BEQ

\[
\begin{align*}
X & \quad \text{Source Encoder} \quad m \quad \text{Source Decoder} \quad \hat{X} \\
01*1*00110... & \quad p(0) = p(1) = 0.4 \\
p(*) = 0.2 & \\
\end{align*}
\]

Send the non-* bits: 01100110...

The encoder knows which are the `*' symbols (source attribute)

Suppose the decoder also knows which are the `*' symbols (genie)

\[R_{BEQ}(0.2) \geq 0.8 \text{ bits/symbol}\]

Number of non `*' symbols to send
\[\approx 10,000 \times (1 - p(\ast)) \]
\[= 10,000 \times 0.8 = 8,000\]

Surprise: the decoder does not need to know which symbols are `*!'
Source Coding Dual to the BEC: BEQ

String Length 10,000

Source Encoder

Compressed bitstream 8,000 bits

Source Decoder

Want the average distortion to be \( \leq 0.2 \)

\[ p(0) = p(1) = 0.4; \quad p(*) = 0.2 \]

How would you do it?

Use channel decoder as source encoder

Use channel encoder as source decoder

\[ \begin{array}{c|c|c}
0 & 0.8 & 0 \\
0.2 & 0.2 & * \\
0.8 & 0.2 & 1 \\
\end{array} \]

\[ \begin{array}{c|c|c}
0 & 0.8 & 0 \\
0.2 & 0.2 & * \\
0.8 & 0.2 & 1 \\
\end{array} \]
Shannon’s prescription: random coding

1) Encoder & Decoder agree on a random codebook
   Shannon’s random coding argument

2) Encoder encodes message
   \( \text{Output the codeword corresponding to the index} \)
   \( \text{Output the index corresponding to the closest codeword} \)

3) Decoder decodes message
   \( \text{Output the index corresponding to the closest codeword} \)
   \( \text{Output the codeword corresponding to the index} \)
Why does it work?

- Encoding successful if there exists an **exact match** for the non-* part of input string
- \( \Pr\{\text{no exact match}\} \leq (1 - 2^{-n(1-p)}) \wedge 2^{nR} \Rightarrow 0 \) if \( R \geq (1 - p) \)
- 8,000 source bits will induce an exact match w.h.p. if random codebook size is at least \( 2^{8,000} \)
Knowledge of the erasure pattern

Channel coding

The encoder does not need to know the don’t care locations.
The decoder knows the erasure pattern.

Source coding

The encoder knows the don’t care locations.
The decoder does not need to know the don’t care locations.
Duality between source and channel coding:

Given a source coding problem with source distr. $\bar{p}(X)$, optimal quantizer $p^*(\hat{X} | X)$ distortion measure $d(x, \hat{x})$ and distortion constraint $D$, (left),

∃ a dual channel coding problem with channel $p^*(x | \hat{x})$, cost measure $w(\hat{x})$, and cost constraint $W$ (right) s.t.:

(i) $R(D) = C(W)$;

(ii) $p^*(\hat{x}) = \arg\max_{p(\hat{x}):X|\hat{x} \sim p^*(x | \hat{x}), Ew \leq W} I(X; \hat{X})$,

where $w(\hat{x}) = c_1 D(p^*(x | \hat{x}) \parallel \bar{p}(x)) + \theta$ and $W = E_{p^*(\hat{x})} w(\hat{X})$. 

REVERSAL OF ORDER
Duality between source and channel coding

Given a source coding problem with source distribution $q(x)$, optimal quantizer $p^*(\hat{x}|x)$, distortion measure $d(x, \hat{x})$ and distortion constraint $D$

There is a dual channel coding problem with channel $p^*(x|\hat{x})$ cost measure $w(\hat{x})$ and cost constraint $W$ such that

$R(D) = C(W)$

$w(\hat{x}) = c_1 D(p^*(x|\hat{x}) \parallel q(x)) + \theta$

$W = E_{p^*(\hat{x})} w(\hat{x})$.

Pradhan, Chou and Ramchandran, 2003
Interpretation of functional duality

For any given source coding problem, there is a dual channel coding problem such that:

• both problems induce the same optimal joint distribution
• the optimal encoder for one is functionally identical to the optimal decoder for the other
• an appropriate channel-cost measure is associated

Key takeaway

Source coding
  distortion measure is as important as the source distribution
Channel coding
  channel cost measure is as important as the channel conditional distribution
Duality between

source coding with side information

and

channel coding with side information
Source coding with side information (SCSI):

- (Only) decoder has access to side-information $S$
- Studied by Slepian-Wolf ‘73, Wyner-Ziv ’76, Berger ’77
- Applications: sensor networks (IoT), digital upgrade, secure compression.
- No performance loss in some important cases
(Only) encoder has access to "interfering" side-information $S$

- Studied by Gelfand-Pinsker '81, Costa '83, Heegard-El Gamal '85

- Applications: data hiding, watermarking, precoding for known interference, writing on dirty paper, MIMO broadcast.

- No performance loss in some important cases
Encoder (only) has access to "interfering" side-information $S$

- Studied by Gelfand-Pinsker ‘81, Costa ‘83, Heegard-El Gamal ’85
- Applications: data hiding, watermarking, precoding for known interference, writing on dirty paper, MIMO broadcast.
- No performance loss in some important cases
SCSI: binary example of noiseless compression

- X and S=> length-3 binary data (equally likely),
- Correlation: Hamming distance between X and S at most 1
- E.g.: when X=[0 1 0], \( S \) => [0 1 0], [0 1 1], [0 0 0], [1 1 0].

Case 1 (\( S \) at both ends)

Encoder computes \( e=S+X \) (mod 2) and sends using 2 bits

Decoder outputs \( X=S+e \) (mod 2)
Transmission at **2 bits/sample achievable**
- Encoder => send index of the coset containing X.
- Decoder => find a codeword in given coset closest to S

Example: X=010, S=110 => Encoder sends message 10
CCSI: illustrative example *(Binary data-embedding/watermarking)*

- **$S$:** 3-bit (uniformly random) host signal (e.g. binary fax)
- **$m$:** message bits to be embedded in the host signal
- Max. allowed distortion between $S$ and embedded host $X$ is 1: $d_H(X,S) \leq 1$
- Clean channel (no attack) model: ($Z=0$); received signal $Y=X$

**Case: 1:** Both encoder and decoder have access to host signal

- Q) How many bits can $m$ be?
- A) 2 bits
Q) Can we still embed a 2 bit message in $S$ while satisfying $d_H(S, X) \leq 1$?

- **Codebook**: partition $U$ into 4 cosets
- Each of 4 messages indexes a coset in $U$.
- Encoder “nudges” $S$ to closest entry $X$ in desired coset of $U$: $d_H(S, X) \leq 1$
- Decoder receives $Y=X$ and declares coset index of $Y$ as message sent.

Messages index one of 4 cosets of $U$:

- **Coset-1**: $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$
- **Coset-2**: $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$
- **Coset-3**: $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$
- **Coset-4**: $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$

Example: $S=011$, $m=01$; $X=001$ (off in $\leq 1$ bit)
Toy example of duality between SCSI and CCSI

SCSI Encoder

M: coset index

SCSI Decoder

\(X\) reconstr.

Distributed compression (SCSI)

(010) X

source

M

data to be embedded

(10)

Data-hiding Encoder

embedded host

\(S\)

(011)

Data embedding (CCSI)

\(\hat{X}\)

recovered data

Data-hiding Decoder

noisy host

\(M\)

data to be embedded

\((010)\)

(10)

(010)

(010)

(010)

(010)

(010)
<table>
<thead>
<tr>
<th><strong>CCSI</strong></th>
<th><strong>SCSI</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Side information at encoder only</td>
<td>Side info. at decoder only</td>
</tr>
<tr>
<td>Channel code is “partitioned” into a bank of source codes</td>
<td>Source code is “partitioned” into a bank of channel codes</td>
</tr>
<tr>
<td>No performance loss in some important cases w.r.t. presence of side information at both ends</td>
<td>No performance loss in some important cases w.r.t. presence of side information at both ends</td>
</tr>
</tbody>
</table>
Markov chains, duality and rate loss

\[ p(s, x, u, \hat{x}) \]

Duality no rate loss

\[ S \rightarrow X \rightarrow U \]
\[ X \rightarrow U, S \rightarrow \hat{X} \]

SCSI

<table>
<thead>
<tr>
<th>SCSi</th>
<th>Enc.</th>
<th>( \hat{X} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Dec.</td>
<td>( \hat{X} )</td>
</tr>
</tbody>
</table>

CCSI

<table>
<thead>
<tr>
<th>CCSI</th>
<th>Enc.</th>
<th>( \hat{X} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Ch.</td>
<td>( \hat{X} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCSI</th>
<th>Dec.</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Ch.</td>
<td>( \hat{X} )</td>
</tr>
<tr>
<td>( \hat{X} )</td>
<td>Dec.</td>
<td>U</td>
</tr>
</tbody>
</table>

no rate loss
Duality between *source coding* & *channel coding with side information*

*source coding with side information (SCSI)*

Source $\rightarrow$ Encoder $\rightarrow$ Bits $\rightarrow$ Decoder $\rightarrow$ Quantized Source

Internet of Things (IoT), video streaming, multiple description coding, secure compression

*channel coding with side information (CCSI)*

Bits $\rightarrow$ Encoder $\rightarrow$ Channel input $\rightarrow$ Decoder $\rightarrow$ Bits

Side-information $\rightarrow$ Watermarking, data hiding, multi-antenna wireless broadcast

*Pradhan, Chou and Ramchandran, 2003*
Chapter 2

Cryptography

• Compressing encrypted data
• Foundations of *modern cryptography*
• All theoretically unbreakable ciphers must have the properties of one-time pad

*Communication Theory of Secrecy Systems*

By C. E. Shannon

1. Introduction and Summary

The problems of cryptography and secrecy systems furnish an interesting application of communication theory. In this paper a theory of secrecy systems is developed. The approach is on a theoretical level and is intended to complement the treatment found in standard works on cryptography. There, a detailed study is made of the many standard types of codes and ciphers, and of the ways of breaking them. We will be more concerned with the general mathematical structure and properties of secrecy systems.
Compression of Encrypted Data

“Correct” order

Source $\sim iid B(0.11)$

Wrong order?

Johnson, Ishwar, Prabhakaran, Schonberg & Ramchandran, 2004
Example

Original Image → 10,000 bits → Encrypted Image → Compressed Encrypted Image → 5,000 bits → Decoding Compressed Image → Final Reconstructed Image
Key Insight!

- $Y = X + K$ where $X$ is independent of $K$

- **Slepian-Wolf theorem:**
  can send $X$ at rate $H(Y|K) = H(X)$
SCSI: binary example of noiseless compression

(Slepian-Wolf ’73)

- $X$ is uniformly chosen from $\{[000], [001], [010], [100]\}$
- $K$ is a length-3 random key (equally likely in $\{0,1\}^3$)
- Correlation: Hamming distance between $Y$ and $K$ at most 1
- Example: when $K=[0\ 1\ 0]$. $Y \Rightarrow [0\ 1\ 0]. [0\ 1\ 1]. [0\ 0\ 0]. [1\ 1\ 0]$

**Case 1**

Encoder computes $X=Y+K$ (mod 2)
Encoder represents $X$ using 2 bits
Decoder outputs $X$ (mod 2)

00 $\Rightarrow$ 000
01 $\Rightarrow$ 001
10 $\Rightarrow$ 010
11 $\Rightarrow$ 100

$=Y+K$
Transmission at 2 bits/sample

Encoder => send index of the coset containing X.

Decoder => find a codeword in given coset closest to K

Example: Y=010 (K=110) => Encoder sends message 10
Geometric illustration

\[ Y = X + K \]

Signal to decoder

\( Y \) (encrypted)

\( X \) (unencrypted & compressible)

Encoder\( \rightarrow \) \( m \)

\( m \rightarrow \) \( \hat{X} \)
Example: geometric illustration

Encoder $X$ $m$ $m$ $K$ $\hat{X}$

Decoder

Side information $K$
Practical Code Constructions

- Use a linear transformation (hash/bin)
- Design cosets to have maximal spacing
  - State of the art linear codes (LDPC codes)
- Distributed Source Coding Using Syndromes (DISCUS)*

*Pradhan & Ramchandran, ‘03
Chapter 3

Sampling theory
- Sample and compute efficient sampling (and connections to learning)
Sampling theorem

Communication in the Presence of Noise

CLAUDE E. SHANNON, MEMBER, IRE

Theorem 1: If a function $f(t)$ contains no frequencies higher than $W$ cps, it is completely determined by giving its ordinates at a series of points spaced $1/2 W$ seconds apart.

Mathematically, this process can be described as follows. Let $x_n$ be the $n$th sample. Then the function $f(t)$ is represented by

$$f(t) = \sum_{n=-\infty}^{\infty} x_n \frac{\sin \pi(2Wt - n)}{\pi(2Wt - n)}.$$  

(7)

linear interpolation!
Aliasing phenomenon

**Time domain**

**Input signal**

- **Sampling at rate 1**
  - **Bandwidth of 1 Hz**
  - No aliasing
    - can recovery by linear filtering

- **Sampling at rate 1/2**
  - Spectrum is aliased!
But what if the spectrum is sparsely occupied?

Henry Landau, 1967

– Know the frequency support
– Sample at rate “occupied bandwidth” $f_{\text{occ}}$ (Landau rate)

When you do not know the support?
• Feng and Bresler, 1996
• Lu and Do, 2008
• Mishali, Eldar, Dounaevsky and Shoshan, 2011
• Lim and Franceschetti, 2017
Filter bank approach

Input in frequency domain

Know the frequency support, filter and sample

Filtering

Sampling spectrum-blind?
Requires $2f_{\text{occ}}$. Can we design a constructive scheme?

Lu and Do, 2008
Puzzle: Gold thief

- One unknown thief
- Steals unknown but fixed amount from each coin
- What is min. no. of weighings needed?
  - 2 are enough!

100 grams each

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 2 & 3 & 4 & 5
\end{bmatrix}
\begin{bmatrix}
-5 \\
-20
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2
\end{bmatrix}
\]

Ratio-test identifies the location
4-thieves among 12-treasurers

Key Ideas:
1. Randomly group the treasurers.
2. If there is a single thief problem
   - Ratio test
   - Iterate.

Questions:
1. How many groups needed?
2. How to form groups?
3. How to identify if a group has a single thief?
Any bandlimited signal $x(t) \in \mathbb{C}$ whose spectrum has occupancy $f_{occ}$ can be sampled asymptotically at rate $f_s = 2f_{occ}$ by a randomized “sparse-graph-coded filter bank” with probability 1 using $O(f_{occ})$ operations per unit time.

Remarks

- Computational cost $O(f_{occ})$ independent of bandwidth
- Requires mild assumptions (genericity)
- Can be made robust to sampling noise
Key insight for spectrum-blind sampling

• To reduce sampling rate, *subsample judiciously*

  subsampling → aliasing

  “judicious” filtering/subsampling → “good” aliasing

• Introduces aliasing (*structured noise*)

• *Filter bank* derived from *capacity-achieving codes for the BEC*: (irregular LDPC codes)

• *Non-linear recovery* instead of linear interpolation
Filter bank for sampling

• Sample the signal at rate B

• Filter and then sample at rate B
Filter bank for sampling

Aggregate sampling rate: $N \frac{f_M}{N} = f_M = \text{Nyquist rate for } x(t)$
‘Sparse-graph-coded’ filter bank

\[
\begin{align*}
\tilde{Y}(e^{j2\pi f}) &= \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\end{pmatrix}
\times \begin{pmatrix}
X_{0}(f) \\
X_{1}(f) \\
\vdots \\
X_{N-1}(f)
\end{pmatrix}
\end{align*}
\]

where

\[
\tilde{X}(Bf) = \begin{pmatrix}
X_{0}(f) \\
\vdots \\
X_{N-1}(f)
\end{pmatrix}
\]

\(m \times N\) matrix
Example — sparse graph underlying the measurements

Sparse bipartite graph

\[ \tilde{Y}(e^{i2\pi f}) = \tilde{X}(Bf) \]

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
\end{pmatrix}
\]
Example — sparse graph underlying the measurements

visual cleaning for presentation: remove edges that connect to non-active bands
**Example — peeling**

**Measurement classification**

- **zero-ton**: no signal
- **single-ton**: no aliasing
- **multi-ton**: aliasing

**Diagram**

- Bands: $X_0(f), X_1(f), X_2(f), X_3(f), X_4(f), X_5(f), X_6(f), X_7(f), X_8(f), X_9(f)$
- Channels: A, B, C, D, E, F
Example — peeling

Assume a **mechanism**:
identifies which channels have no aliasing (here B and F) and maps them to which bands they came from (here 1 and 4 resp.)

Measurement classification

- **zero-ton:** no signal
- **single-ton:** no aliasing
- **multi-ton:** aliasing
Example — peeling

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from

**output:**
- channel B: (red, index = 1)
- channel F: (blue, index = 4)
Example — peeling

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from

**output:**
- channel B: (red, index = 1)
- channel F: (blue, index = 4)

*peel from channels they alias into!*
Example — peeling

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from
Example — peeling

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from

**output:**
channel D: (green, index = 8)
channel E: (cyan, index = 5)
Example — peeling

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from

**output:**
channel D: (green, index = 8)
channel E: (cyan, index = 5)

*peel from channels they alias into!*
**Example — peeling**

<table>
<thead>
<tr>
<th>bands</th>
<th>channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_0(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_1(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_2(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_3(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_4(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_5(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_6(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_7(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_8(f)$</td>
<td></td>
</tr>
<tr>
<td>$X_9(f)$</td>
<td></td>
</tr>
</tbody>
</table>

**mechanism:**
identifies which channels have no aliasing and maps them to which bands they came from

*signal is completely recovered!*
Realizing the *mechanism*

Identify which channels have no aliasing and map them to bands

same magnitude response
‘*stairs*’ phase response

identifies dark blue band as a singleton
Construction of the sparse-graph code

- Designed through capacity-approaching sparse-graph codes
- Connect each band to channels at random according to a carefully chosen degree distribution.
- Asymptotically, number of channels is \((1 + \epsilon)\) times the number of active bands

\[
P(\text{degree} = j) \propto \frac{1}{j(j-1)}, \text{ for } j = 2, \ldots, D + 1
\]

\[
D > 1/\epsilon
\]

Degree distribution for \(\epsilon = 1/20\)

Luby et al. 2001
Construction of the sparse-graph code

Regular graph construction:
Connect every variable node to d check nodes chosen uniformly at random
Density evolution

Regular graph construction:
Connect every variable node to \( d \) check nodes chosen uniformly at random

example: \( d = 4 \)
• Pick an arbitrary edge in the graph \((c, v)\).

Regular graph construction: 
Connect every variable node to \(d\) check nodes chosen uniformly at random.
• Examine its directed neighborhood at depth-2\(\ell\)
• Examine its directed neighborhood at depth-2$\ell$
Density evolution

The variable node $v$ can be resolved if any of these check nodes can be resolved.

A check node is resolved from below if all of the variable nodes connected to it from below are resolved.

$\mathbb{P}_\ell$ Probability of being present at depth $2\ell$
Example with a depth-2 neighborhood

\[ p' = \left( 1 - p' \right)^3 \left( 1 - p' \right)^2 \left( 1 - p' \right) \]

In the general setting where each variable has \( d \) edges, the degree distribution of the graph results in the following:

\[ p' = \left( \frac{1}{e} p' \right) \frac{1}{d} \]

Density evolution

The variable node \( v \) can be resolved if any of these check nodes can be resolved.

A check node is resolved from below if all of the variable nodes connected to it from below are resolved.
Example with a depth-2 neighborhood

\[ p_\ell = \mathbf{1} \times (1 - p_{\ell - 1})^3 \times (1 - p_{\ell - 1})^2 \times (1 - p_{\ell - 1}) \]

In the general setting where each variable has \( d \) edges, the degree distribution of the graph results in the following:

\[ p_\ell = \binom{d}{1} \frac{p_\ell}{\binom{d}{1} p_{\ell - 1}} \]

The variable node \( v \) can be resolved if any of these check nodes can be resolved.

A check node is resolved from below if all of the variable nodes connected to it from below are resolved.

Power 3 is because the check node has 3 variable nodes as children.
Example with a depth-2 neighborhood

\[ p_{\ell} = \tau (1 - p_{\ell}^1) \]

In the general setting where each variable has \( d \) edges, the degree distribution of the graph results in the following:

\[ p_{\ell} = (1 - (1 - p_{\ell-1})^3) \times (1 - (1 - p_{\ell-1})^2) \times \]

The variable node \( v \) can be resolved if any of these check nodes can be resolved.

A check node is resolved from below if all of the variable nodes connected to it from below are resolved.
Density evolution

$P_\ell$ Probability of being present at depth $2\ell$

The variable node $v$ can be resolved if \textit{any} of these check nodes can be resolved.

A check node is resolved from below if \textit{all} of the variable nodes connected to it from below are resolved.

$$p_\ell = \left[ 1 - (1 - p_{\ell-1})^3 \right] \times \left[ 1 - (1 - p_{\ell-1})^2 \right] \times \left[ 1 - (1 - p_{\ell-1})^2 \right]$$
Example with a depth-2 neighborhood

\[ p_\ell = [1 - (1 - p_{\ell-1})^3] \times [1 - (1 - p_{\ell-1})^2] \times [1 - (1 - p_{\ell-1})^2] \]

It generalizes to left $d$-regular graphs, we have

\[ p_\ell = \sum_{\text{left } d\text{-regular graphs}} \]
Density evolution

Example with a depth-2 neighborhood

\[ p_\ell = \left(1 - (1 - p_{\ell-1})^3\right) \times \left(1 - (1 - p_{\ell-1})^2\right) \times \left[1 - (1 - p_{\ell-1})^2\right] \]

It generalizes to left $d$-regular graphs, we have

\[ p_\ell = \left(1 - \frac{e^{p_{\ell-1}}}{1 + e^{p_{\ell-1}}} \right) \]

**Regular graph construction:**
Connect every variable node to $d$ check nodes chosen uniformly at random

**Number of children of check nodes has Poisson distribution with mean $Kd/M$**

\[ \Pr\{\text{a check node is resolved}\} = \sum_{c} e^{(Kd/M)(Kd/M)^c/c!} (1 - p_{\ell-1})^c = e^{Kd/M p_{\ell-1}} \]

Example: $d = 4$
Density evolution

\[ p_\ell = [1 - (1 - p_{\ell-1})^3] \times [1 - (1 - p_{\ell-1})^2] \times [1 - (1 - p_{\ell-1})^2] \]

\[ p_\ell = \left(1 - e^{-\frac{Kd}{M}p_{\ell-1}}\right)^{d-1} \]

- Need \( p_\ell \to 0 \) as \( \ell \to \infty \).
- Choose \( K, M \) and \( d \) so that \( p_\ell \) goes to zero!

Example: \( d = 4 \)
Density evolution

\[ d = 3 \]
\[ M = 1.23K \]

- \( K = \# \text{ of active bands} \)
- \( M = \# \text{ of channels} \)
- \( d = \text{ left degree (\# of edges from bands to channels)} \)

\[ p_\ell = \left( 1 - e^{-\frac{Kd}{M}p_{\ell-1}} \right)^{d-1} \]
Density evolution

Set:
- $M = (1 + \epsilon)K$
- $D > 1/\epsilon$
- Node degree distribution $P(\text{degree} = j) = \frac{D+1}{D} \frac{1}{j(j-1)!}$, for $j = 2, ..., D + 1$

\[
p_{\ell} = \frac{1}{H(D)} \sum_{j=2}^{D+1} \frac{1}{j-1} \left(1 - e^{-\frac{\bar{d}}{1+\epsilon} p_{\ell-1}}\right)
\]

$p_{\ell}$ goes to zero!
Density evolution

\[ M = (1 + \epsilon)K \]
\[ \epsilon = 0.1 \]
Density evolution

EXIT chart

$p_{\ell+1}$ vs $p_{\ell}$

$M = (1 + \epsilon)K$

$\epsilon = 0.1$
Algorithm analysis

- **Density Evolution**
  - assumes that the directed neighborhood is a tree
  - tree-based average analysis
    Density evolution equations

\[ p_\epsilon \text{ can be made arbitrarily small with } O(1) \text{ number of iterations} \]
• Density Evolution
  – assumes that the directed neighborhood is a tree
  – tree-based average analysis

Density evolution equations

\[ p_\varepsilon \text{ can be made arbitrarily small with } O(1) \text{ number of iterations} \]

\[ Kd(1 - p_\varepsilon) \text{ edges removed} \]
Algorithm analysis

Performance concentration:
- Actual performance concentrated around the density evolution
- $P(|\text{# of actual remaining edges} - Kdp_{\ell}| > \varepsilon_2) \rightarrow 0, \forall \varepsilon_2 > 0$

$Kd(1 - p_{\ell})$ edges removed
Algorithm analysis

\[ K_d(1 - p_e) \text{ edges removed} \]

\[ K_d p_e \text{ edges remain} \]

\[ K_d \text{ edges to be removed} \]
Algorithm analysis

- Expander Graph
  - the remaining $Kd p_\ell$ edges form an expander graph
  - expander graphs guarantee steady supplies of single-toms

ALL non-zero coefficients recovered w.h.p.
Back to sub-Nyquist sampling: Numerical experiment

- Lebesgue measure $f_L = 0.1$
- Number of slices $N = 1000$
- Number of channels $M = 284$
- Sampling rate $f_S = 0.284$
Interesting connection

• **Minimum-rate spectrum-blind sampling**

• **Coding theory** and **sampling theory**
  – Capacity-approaching codes for erasure channels
  – Filter banks that approach Landau rate for sampling
"Peeling-based" turbo engine

Sparse-Graph Code

Divide

Concur

"Solve-if-trivial" sub-engine
Broad scope of applications

Sparse-graph codes

- Compressive phase retrieval
- Sparse mixed linear regression
- Compressed sensing
- Sub-Nyquist sampling theory
- Sparse Spectrum (DFT/WHT)
- Fast neighbor discovery for IoT (group testing)

Pedarsani, Lee, R., 2014
Ocal, Li, R., 2016
Pawar, R., 2013
Li, Pawar, R., 2014
Lee, Pedarsani, R., 2015
Li, Pawar, R., 2014
Yin, Pedarsani, Chen, R., 2016
Broad scope of applications

- Compressive phase retrieval
  - Pedarsani, Lee, R., 2014

- Sparse mixed linear regression
  - Yin, Pedarsani, Chen, R., 2016

- Sparse-graph codes

- Sub-Nyquist sampling theory
  - Ocal, Li, R., 2016

- Sparse Spectrum (DFT/WHT)
  - Pawar, R., 2013
  - Li, Pawar, R., 2014

- Fast neighbor discovery for IoT (group testing)
  - Lee, Pedarsani, R., 2015

- Compressed sensing
  - Li, Pawar, R., 2014
Chapter 4

Speeding up learning and sparse recovery

Sameer Pawar
Simon Li
Orhan Ocal
Motivation

- Given training data points \((x, y)\), our goal is to learn

\[(x : \text{feature}, y : \text{label})\]
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \xrightarrow{\text{Dataset}} \quad x \xrightarrow{f(\cdot)} y = f(x)
\]
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \rightarrow \quad x \quad \rightarrow \quad f(\cdot) \quad \rightarrow \quad y = f(x)
\]

Dataset

(e.g. area, bedrooms)

(e.g. house prices)
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - **a certain rule** \(f\) that explains the label \(y\) based on features \(x\):

  \[
  (x : \text{feature, } y : \text{label}) \quad \rightarrow \quad x \quad \rightarrow \quad f(\cdot) \quad \rightarrow \quad y = f(x)
  \]

  (e.g. area, bedrooms) \quad \rightarrow \quad (e.g. house prices)

- Questions of interest
  - **Sample complexity**: how many data points do we need?
  - **Computational complexity**: how much time does it take?
Motivation

• Given training data points \((x, y)\), our goal is to learn
  
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[(x : \text{feature}, y : \text{label}) \quad \xrightarrow[]{\text{Dataset}} \quad x \quad \xrightarrow[]{\text{\(f(\cdot)\)}} \quad y = f(x) + \epsilon \quad \text{(e.g. house prices)}\]

• Questions of interest
  
  - Sample complexity: how many data points do we need?
  - Computational complexity: how much time does it take?
  - Robustness: how accurate and stable is it?
Motivation

- Given training data points \((x, y)\), our goal is to learn
  
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \xrightarrow{\text{Dataset}} \quad x \xrightarrow{f(\cdot)} y = f(x) + \epsilon
\]

  (e.g. area, bedrooms)  
  (e.g. house prices)

  e.g. \(f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b\)

  Problem Dimension \(N = 3\)
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

  \[(x : \text{feature}, y : \text{label})\]

  \[\begin{align*}
  x &\rightarrow f(\cdot) \\
  (\text{e.g. area, bedrooms}) &\rightarrow y = f(x) + \epsilon \\
  (\text{e.g. house prices})
  \end{align*}\]

  e.g. \(f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b\)

  EASY!

  Problem Dimension \(N = 3\)
Motivation

- Given training data points \((x, y)\), our goal is to learn
  
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, \ y : \text{label}) \quad \Rightarrow \quad \begin{array}{c}
\text{Dataset} \\
\text{(e.g. area, bedrooms)}
\end{array} \quad \xrightarrow{f(\cdot)} \quad \begin{array}{c}
y = f(x) + \epsilon \\
\text{(e.g. house prices)}
\end{array}
\]

\[
e.g. \ f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b
\]

However…

Problem Dimension \(N = 3\)
Motivation

- Given training data points \((x, y)\), our goal is to learn
  
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x \, : \text{feature}, y \, : \text{label}) \quad \xrightarrow{\text{Dataset}} \quad x \quad \xrightarrow{f(\cdot)} \quad y = f(x) + \epsilon
\]

  (e.g. area, bedrooms)  

  (e.g. house prices)

  \[f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b\]

  \text{in reality...}

  Problem Dimension \(N = 3\)
Motivation

- Given training data points \((x, y)\), our goal is to learn

  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \xrightarrow{\text{Dataset}} \quad x \xrightarrow{f(\cdot)} y = f(x) + \epsilon
\]

\[
e.g. \ f(x) = \frac{D}{D\epsilon} w_2 \bigg| \bigg( w_1 v + w_2 \bigg) \bigg| - w_2 - \frac{1}{\beta} \epsilon \bigg( w_2 \bigg) \bigg| - \frac{\lambda}{2} \frac{D^2}{Dx^2} \bigg| - \frac{\lambda}{2} \frac{D^2}{Dx^2} \bigg| w_2 = -\epsilon, \\
\left[1 + \epsilon \frac{D}{D\epsilon} \left( \frac{D x}{D\epsilon} \right) \bigg| - 2 \left( \frac{D x}{D\epsilon} \right) - 2 \left( \frac{D x}{D\epsilon} \right) D_\epsilon - 2 \left( \frac{D x}{D\epsilon} \right) D_\epsilon + \frac{1}{\beta} \epsilon \bigg( \frac{D x}{D\epsilon} \bigg) - \frac{D^2}{D\epsilon^2} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| - \frac{\lambda}{2} \frac{D^2}{Dx^2} \bigg| - \frac{\lambda}{2} \frac{D^2}{Dx^2} \bigg| w_2 = -\epsilon, \\
\left[1 + \epsilon \frac{D}{D\epsilon} \frac{x}{D\epsilon} \bigg| + \frac{x}{D\epsilon} \bigg| \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| w_2 = -\epsilon, \\
\left[1 + \epsilon \frac{D}{D\epsilon} \frac{x}{D\epsilon} \bigg| + \frac{x}{D\epsilon} \bigg| \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| w_2 = -\epsilon, \\
\left[1 + \epsilon \frac{D}{D\epsilon} \frac{x}{D\epsilon} \bigg| + \frac{x}{D\epsilon} \bigg| \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D x}{D\epsilon} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| - \frac{D^2}{D\epsilon^2} \bigg| w_2 = -\epsilon.
\]

in reality…

Problem Dimension \(N \to \infty\)
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - **a certain rule** \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \xrightarrow{f(\cdot)} \quad y = f(x) + \epsilon
\]

\[
e.g. \quad f(x) = \frac{D}{\partial x} \frac{v \cdot w}{T} + \frac{\nabla v \cdot w}{T} + \frac{v \cdot \nabla w}{T} - \left(\frac{v \cdot w}{T} + \frac{v \cdot \nabla w}{T}\right) \left(v + \frac{\partial w}{\partial x}\right)
\]

- Sample cost
- Run-time

\[
\text{Problem Dimension } N \rightarrow \infty
\]
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[
(x : \text{feature}, y : \text{label}) \quad \xrightarrow{\text{f(\cdot)}} \quad y = f(x) + \epsilon
\]
Motivation

- Given training data points \((x, y)\), our goal is to learn
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):

\[(x : \text{feature}, y : \text{label}) \rightarrow f(\cdot) \rightarrow y = f(x) + \epsilon\]

- What if — we can actively choose training data
  - the model has sublinear d.o.f
Motivation

- Given training data points \((x, y)\), our goal is to learn
  
  - a certain rule \(f\) that explains the label \(y\) based on features \(x\):
    
    \[(x : \text{feature}, y : \text{label}) \rightarrow f(\cdot) \rightarrow y = f(x) + \epsilon\]

- What if — we can actively choose training data — the model has sublinear d.o.f.?

Can we achieve fast & robust learning with active sampling + coding theory?
Applications

MRI
Sub-Nyquist Sampling

Machine Learning
Computational Imaging
IoT
Sparse Spectrum (DFT/WHT)

Sparse-graph codes

Sameer Pawar
Xiao (Simon) Li
Orhan Ocal
Learning polynomials: HS algebra edition

• Given \( f(x) = \sum_{n=0}^{N-1} F_n x^n \)
• Find coefficients \( \{F_n\}_{n=0}^{N-1} \)

\[ f(x) \quad \rightarrow \quad f(x_i) \]

Q. How many evaluations do we need?

A. \( N \) evaluations

\[ y = a + bx + cx^2 + dx^3 + ex^4 + fx^5 + gx^6 + hx^7 \]
Recovering the coefficients

- Given \( f(x) = \sum_{n=0}^{N-1} F_n x^n \)
- Find coefficients \( \{F_n\}_{n=0}^{N-1} \)

\[
\begin{bmatrix}
  f(X_0) \\
  f(X_1) \\
  f(X_2) \\
  \vdots \\
  f(X_{19})
\end{bmatrix}
= 
\begin{bmatrix}
  1 & X_0 & \cdots & X_{19}^0 \\
  1 & X_1 & \cdots & X_{19}^1 \\
  1 & X_2 & \cdots & X_{19}^2 \\
  \vdots \\
  1 & X_{19} & \cdots & X_{19}^{19}
\end{bmatrix}
\begin{bmatrix}
  F_0 \\
  F_1 \\
  F_2 \\
  \vdots \\
  F_{19}
\end{bmatrix}
\]

inverse Discrete Fourier Transform (DFT)

\[ X_m = e^{i \frac{2\pi}{N} m} \]
What if only $K$ of $N$ coeffs. non-zero?

Example:
Degree $N = 1$ million
Sparsity $K = 200$

(spoiler alert)
# evaluations $= 616$ ($\approx 3K$)
computations $= O(K \log K)$
Discrete Fourier Transform (DFT)

Compute the DFT of $x \in \mathbb{C}^N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} kn}, \quad n = 0, \cdots, N - 1$$

What if only $K$ out of $N$ Fourier coefficients are non-zero?

**Example:**
Length $N = 1$ million
Sparsity $K = 200$

*(spoiler alert)*

Sample complexity: $N$
Computational cost: $O(N \log N)$

# evaluations = $616 \ (\approx 3K)$
computations = $O(K \log K)$
Compute the $K$-sparse DFT of $x \in \mathbb{C}^N$ with $K \ll N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k \in \mathcal{K}} X[k] e^{i \frac{2\pi k}{N} n} \quad n = 0, \cdots, N - 1$$

Support $\mathcal{K}$ chosen from $[N]$ uniformly at random

**FFAST** (Fast Fourier Aliasing-Based Sparse Transform)

- **Noiseless**: For $K$ sublinear in $N$
  - Uses fewer than $4K$ samples
  - $O(K \log K)$ computation time
- **Robust to noise**: $O(K \log^{4/3} N)$ samples in $O(K \log^{7/3} N)$ time

*Sub-linear time recovery when d.o.f. sublinear!*

Pawar, R, IEEE Trans. Inf. Theory, 2018
Aliasing

Signal and its spectrum

Sampling

Sub-sampling
Insights

Sub-sampling below Nyquist rate

Aliasing in the frequency domain

Clever sub-sampling (for sparse case)

Good “alias” code

Chinese-Remainder-Theorem guided subsampling

Sparse graph codes

We use coding-theoretic tools

Design:

• Randomized constructions of good sparse-graph codes

Analysis:

• Density evolution, Martingales, Expander graph theory...
Main idea

time-domain $x[n]$, length $N = 20$

frequency-domain $X[k]$, sparsity $K = 5$

$X[1] = 1$

$X[3] = 4$

$X[5] = 1$

$X[10] = 3$

$X[13] = 7$

$\text{DFT} \quad \text{(length = 20)}$
Main idea

time-domain $x[n]$, length $N = 20$

frequency-domain $X[k]$, sparsity $K = 5$

DFT

↓5

subsample by 5
Main idea

**time-domain** \( x[n] \), length \( N = 20 \)

**frequency-domain** \( X[k] \), sparsity \( K = 5 \)

\[
\begin{array}{c}
X[3] = 4 \\
X[10] = 3 \\
X[13] = 7 \\
\end{array}
\]

\[ \iff \text{DFT} \iff \text{DFT} \]

**Our Measurements**

\[
\begin{array}{cccc}
\end{array}
\]

↓5

subsample by 5
**Main idea**

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

$\downarrow 5$

subsample by 5

Aliasing

$\iff \text{DFT} \iff$

(length = 20)

$\iff \text{DFT} \iff$

(length = 4)

$U[0] \quad U[1] \quad U[2] \quad U[3]$
Main idea

time-domain $x[n]$, length $N = 20$

frequency-domain $X[k]$, sparsity $K = 5$

↓5

subsample by 5

$\downarrow$ DFT
(length = 20)

Aliasing

$\leftarrow$ DFT $\rightarrow$
(length = 4)

$U[0] \quad U[1] \quad U[2] \quad U[3]$
Main idea

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

\[ X[3] = 4 \]
\[ X[10] = 3 \]
\[ X[13] = 7 \]

\[ X[1] = 1 \]
\[ X[5] = 1 \]

↓5

subsample by 5

\[ \downarrow \](length = 20)

\[ \downarrow \](length = 4)

Aliasing
Main idea

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

$\downarrow 5$

subsample by 5

$\leftrightarrow$ DFT $\rightarrow$ (length = 20)

$\leftrightarrow$ DFT $\rightarrow$ (length = 4)

zero-ton


Main idea

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

↓ 5

subsample by 5

DFT (length = 20)

$X[3] = 4$

$X[10] = 3$

$X[1] = 1$

$X[5] = 1$

DFT (length = 4)

$U[0]$

$U[1]$

$U[2]$

$U[3]$

zero-ton multi-ton
Main idea

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

$\downarrow 5$

subsample by 5

$\iff \text{DFT} \implies \text{(length = 20)}$

$\iff \text{DFT} \implies \text{(length = 4)}$

$U[0] \quad U[1] \quad U[2] \quad U[3]$

zero-ton multi-ton single-ton single-ton
Main idea

**time-domain** $x[n]$, length $N = 20$

**frequency-domain** $X[k]$, sparsity $K = 5$

$\xrightarrow{\text{DFT}}$ (length = 20)

↓ 5

subsample by 5

$\xrightarrow{\text{DFT}}$ (length = 4)

zero-ton multi-ton single-ton single-ton
Main idea

time-domain $x[n]$, length $N = 20$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x[n]$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

frequency-domain $X[k]$, sparsity $K = 5$

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| $X[k]$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |


$(\text{length } = 20)$

Our Measurements

|----------------|---------|---------|---------|---------|

subscript $U_S$ suggests shift

shift and subsample by 5

$\downarrow 5$

$(\text{length } = 4)$
Main idea

time-domain \( x[n] \), length \( N = 20 \)

\[
\begin{align*}
\omega &= e^{-j \frac{2\pi}{20}} \\
\end{align*}
\]

\( \iff \text{DFT} \iff \) (length = 20)

\( \downarrow 5 \)
shift and subsample by 5

frequency-domain \( X[k] \), sparsity \( K = 5 \)

\[
\begin{align*}
\end{align*}
\]

\( \iff \text{DFT} \iff \) (length = 4)

zero-ton multi-ton single-ton single-ton
Main Idea

Stage 1

downsampling by 5

$$x[n]$$

$$X[13]$$

$$X[10]$$

$$X[5]$$

$$X[3]$$

$$X[1]$$

$$U[0]$$

$$U[1]$$

$$U[2]$$

$$U[3]$$

DFT
Main Idea

Stage 1
downsample by 5

\[
\begin{align*}
X[13] & \rightarrow U[0] \quad U_s[0] \\
X[1] & \rightarrow DFT \quad \downarrow \\
\end{align*}
\]

Stage 1

DFT

\[
\begin{align*}
U[0] & \rightarrow U_s[0] \\
U[1] & \rightarrow U_s[1] \\
\end{align*}
\]

DFT

\[
\begin{align*}
\text{shift} & \rightarrow \quad \downarrow \\
x[n] & \rightarrow \quad \downarrow \\
\end{align*}
\]
Stage 1
downsampling by 5

Stage 2
downsampling by 4

Main Idea

DFT

Stage 1

DFT

shift

Stage 2

x[n]
Main Idea

Stage 1

downsampling by 5

Stage 2
downsampling by 4

... 

Stage d

DFT

\[ \downarrow \]

shift

\[ x[n] \]
Main Idea

Stage 1
downsample by 5

Stage 2
downsample by 4

$X[13]$  
$X[10]$  
$X[5]$  
$X[3]$  
$X[1]$

single-ton  
single-ton  
multi-ton  
zero-ton  
zero-ton  
multi-ton  
multi-ton  
zero-ton  
zero-ton  
multi-ton

peeling decoder
Main Idea

Stage 1
downsample by 5

Stage 2
downsample by 4

- multi-ton
- zero-ton
- single-ton

peeling decoder
Main Idea

Stage 1
downsampling by 5

$X[13]$  
$X[10]$  
$X[5]$  
$X[3]$  
$X[1]$

Stage 2
downsampling by 4

multi-ton  
zero-ton  
multi-ton  
zero-ton  
single-ton  
single-ton

peeling decoder
Main Idea

Stage 1
downsample by 5

Stage 2
downsample by 4

X[13]
X[10]
X[5]
X[3]
X[1]

peeling
decoder

single-ton
multi-ton
zero-ton
single-ton
zero-ton
Main Idea

Stage 1
downsample by 5

Stage 2
downsample by 4

X[13]
X[10]
X[5]
X[3]
X[1]
How do we induce good graphs that will work?
Sparse DFT Computation = Decoding over Sparse Graphs

- Erased symbols/packets
- Parity checks
- Non-zero DFT Coefficients
- Aliased frequency bins
- Explicit graph: design well-understood.
- Implicit graph induced by subsampling.
- $(N - K)$ correctly received packets.
- $K$ erased packets.
- Peeling decoder recovers values & locations.
Chinese-Remainder-Theorem: A number between 0-19 is uniquely represented by its remainders modulo (4,5).

> The two graph ensembles are identical.
$N = 100 \times 103 \times 107; K \approx 200; M \approx 600$

- Subsample by $100 \times 103$
- Shift & Subsample by $100 \times 103$
- DFT 107-length
- DFT 107-length

Peeling Decoder
$N = 100 \times 103 \times 107; K \approx 200; M \approx 600$

Subsample by $100 \times 103$

Shift & Subsample by $100 \times 103$

Subsample by $100 \times 107$

Shift & Subsample by $100 \times 107$

Subsample by $103 \times 107$

Shift & Subsample by $103 \times 107$

DFT $107$-length

DFT $107$-length

DFT $103$-length

DFT $103$-length

DFT $100$-length

DFT $100$-length

Peeling Decoder
Sparse polynomial learning

What if only (very few) $K$ of the $N$ polynomial coeffs. $\{F_n\}$ are non-zero?

E.g. deg. $N=1$ million
Sparsity $K=200$

$# \text{ evals. } M = 616$

$N=100 \times 103 \times 107$
$K \approx N^{1/3}$
$M=2*(100+103+107)-4$
Noiseless setting: Theory vs. practice

Theory is by using *density evolution equations*
From Noiseless to Noisy

Noiseless - FFAST

Noisy - R-FFAST

Use more shifts
Magnetic resonance imaging

Fourier Transform

FFT/IFFT
Numerical Phantoms for Cardiovascular MR

http://www.biomed.ee.ethz.ch/research/bioimaging/cardiac/mrxcat

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

336 = 16 \times 21
323 = 17 \times 19
Numerical Phantoms for Cardiovascular MR

temporal difference across different frames of the phantom
Real Time Reconstruction in MATLAB on a Macbook

Measurements: 35.33% of Nyquist rate
MRI Viewfinder

Kodak, 1975

Viewing the photograph

Canon, 2000

Real-time MRI with viewfinder
Chapter 4 (part 2)

Speeding up learning and recovery of pseudo-Boolean functions

Sameer Pawar
Simon Li
Orhan Ocal
Walsh-Hadamard Transform (WHT)

• N-point Discrete Fourier Transform (DFT)
  \[ f[m] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] e^{\frac{2\pi i k}{N} m}, \quad m = 0, \cdots, N - 1 \]

• N-point Walsh-Hadamard (WHT) with \( N = 2^n \)
  \[ f[m] = \sum_{k \in \{0,1\}^n} F[k] (-1)^{\langle k, m \rangle}, \quad m \in \{0,1\}^n \]
  Equivalent to a high-dim. DFT over the hyper-cube

• \( F[k] \) is sparse in many machine learning applications:
  – Decision tree and regression tree
  – Evolutionary biology
  – Hypergraph sketching
WHT: polynomial interpretation

\[ f[m] = \sum_{k \in \{0,1\}^n} F[k](-1)^{\langle k, m \rangle}, \quad m \in \{0,1\}^n \]

• Set \( x_i = (-1)^{m_i} \) to get a multilinear polynomial \( f : \{-1,1\}^n \to \mathbb{R} \)

**Ex. \( n = 2 \):**
\[ f(x_1, x_2) = F_0 1 + F_1 x_1 + F_2 x_2 + F_3 x_1 x_2 \]

**Ex. \( n = 3 \):**
\[ f(x_1, x_2, x_3) = F_0 1 + F_1 x_1 + F_2 x_2 + F_3 x_1 x_2 + F_4 x_3 + F_5 x_1 x_3 + F_6 x_2 x_3 + F_7 x_1 x_2 x_3 \]

1-1 mapping between WHT coeffs. \( F_i \)'s and the evaluations of \( f(x_1, x_2, x_3) \) at \( x_i = (-1)^{m_i} \)
Recovering the function

Example for

\[
\begin{bmatrix}
  f(1, 1) \\
  f(-1, 1) \\
  f(1, -1) \\
  f(-1, -1)
\end{bmatrix}
= \begin{bmatrix}
  1 & 1 & 1 & 1 \\
  1 & -1 & 1 & -1 \\
  1 & 1 & -1 & -1 \\
  1 & -1 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
  F_{\{\}} \\
  F_{\{1\}} \\
  F_{\{2\}} \\
  F_{\{1,2\}}
\end{bmatrix}
\]
Polynomial recovery

Recover the polynomial $f : \{-1, 1\}^n \rightarrow \mathbb{R}$

**Example for $n = 3$:**

$f(x_1, x_2, x_3) = F_0 1 + F_1 x_1 + F_2 x_2 + F_3 x_1 x_2 + F_4 x_3 + F_5 x_1 x_3 + F_6 x_2 x_3 + F_7 x_1 x_2 x_3$

<table>
<thead>
<tr>
<th>input</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1,1,...,1)$</td>
<td>$f(1,1,...,1)$</td>
</tr>
<tr>
<td>$(1,1,...,-1)$</td>
<td>$f(1,1,...,-1)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Evaluate the function at every point

**Sample complexity:** $N = 2^n$

**What if only $K$ out of $N$ WHT coeffs. are non-zero?**

Ex:  
- No. of variables $n = 30$
- No. of input combinations $N = 1$ billion
- Sparsity $K = 64$

# evaluations $= 2600 \ (\approx 1.23Kn)$
Main Result

Example for $n = 3$:

$$f(x_1, x_2, x_3) = F_0 1 + F_1 x_1 + F_2 x_2 + F_3 x_1 x_2 + F_4 x_3 + F_5 x_1 x_3 + F_6 x_2 x_3 + F_7 x_1 x_2 x_3$$

We can learn $f: \{-1,1\}^n \rightarrow \mathbb{R}$ whose spectrum is $K$-sparse:

• with a sample complexity of $O(nK)$
• with a computational complexity of $O(nK \log K)$
• can be made robust to noise

$\kappa = \log(N)$

Insights:

- Sub-sampling
- Clever sub-sampling (for sparse case)

Aliasing in the WHT domain

Good “alias” code ($Sparse$ graph codes)
Walsh-Hadamard Transform

Equivalent to a high-dim. DFT over the hyper-cube
Walsh-Hadamard Transform

“time” domain

WH domain
Walsh-Hadamard Transform

“time” domain

WH domain
Walsh-Hadamard Transform

“time” domain

WH domain
WHT – Hypergraph Sketching

\[ n = \# \text{ of books} \]
\[ s = \# \text{ of sale patterns} \]

\[ 2^n \approx 10^9 \text{ possible hyperedges if } n = 30 \]

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the cuts of the graph instead!
WHT – Hypergraph Sketching

\[ n = \# \text{ of books} \]
\[ s = \# \text{ of sale patterns} \]

2\(^n\) \approx 10^9 \text{ possible hyperedges if } n = 30

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the cuts of the graph instead!

consider a cut:

\[ x_1 = \cdots = x_5 = +1 \quad \implies \text{cut value } f(x) = 0 \]
\[ x_6 = \cdots = x_{25} = -1 \]
WHT – Hypergraph Sketching

- $n = \# \text{ of books}$
- $s = \# \text{ of sale patterns}$

$2^n \approx 10^9$ possible hyperedges if $n = 30$

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the cuts of the graph instead!

consider a cut:

$x_1 = \cdots = x_{10} = +1 \implies \text{cut value } f(x) = 1$

$x_{11} = \cdots = x_{25} = -1$
• recover all sale patterns (hyperedges) without logging every transaction?
• sketch the cuts of the graph instead!

$n = \# \text{ of books}$
$s = \# \text{ of sale patterns}$

$2^n \approx 10^9$ possible hyperedges if $n = 30$
WHT – Hypergraph Sketching

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the cuts of the graph instead!
- Generally speaking, we have the cut function

\[ f(x) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23} \]

\[ n = \# \text{ of books} \]
\[ s = \# \text{ of sale patterns} \]

\[ 2^n \approx 10^9 \text{ possible hyperedges if } n = 30 \]
$n = \# \text{ of books}$

$s = \# \text{ of sale patterns}$

$2^n \approx 10^9$ possible hyperedges
if $n = 30$

- small # of sale patterns $s \ll n$
- small # of items per sale $d \ll n$

- $K$-sparse polynomial
  - $K \leq s2^{d-1}$

- Generally speaking, we have the **cut function**

\[
f(x) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23}
\]
WHT – Hypergraph Sketching

$n = \# \text{ of books}$

$s = \# \text{ of sale patterns}$

$2^n \approx 10^9$ possible hyperedges if $n = 30$

• Generally speaking, we have the cut function

$$f(x) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23}$$
$n = 50$ books
$d = 2$ items/sale
$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$
WHT – Hypergraph Sketching

- $n = 50$ books
- $d = 2$ items/sale
- $s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$
\[ n = 50 \text{ books} \]
\[ d = 2 \text{ items/sale} \]
\[ s = 250 \text{ sale patterns} \]

- total cut values \(2^n = 2^{50}\)
- sparsity \(K \leq s2^{d-1} = 500\)
- \# of cut queries \(O(Kn) \approx 25000\)
$n = 50$ books

$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$
WHT – Hypergraph Sketching

\[ n = 50 \text{ books} \]
\[ d = 2 \text{ items/sale} \]
\[ s = 250 \text{ sale patterns} \]

- total cut values \( 2^n = 2^{50} \)
- sparsity \( K \leq s 2^{d-1} = 500 \)
- # of cut queries \( O(Kn) \approx 25000 \)
$n = 50$ books
$d = 2$ items/sale
$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$
Open source implementations

- **Sparse FFT and Sparse WHT** implemented in C++
- Publicly available on **GitHub**
  [https://github.com/ucbasics](https://github.com/ucbasics)
- Hardware implementation of sparse FFT

---

**IEEE JSSC, 2019**  
**A Real-Time, 1.89-GHz Bandwidth, 175-kHz Resolution Sparse Spectral Analysis RISC-V SoC in 16-nm FinFET**

A. Wang, W. Bae, J. Han, S. Bailey, O. Ocal, P. Rigge, Z. Wang, K. Ramchandran, E. Alon, B. Nikolic
“Peeling-based” turbo engine

Sparse-Graph Code

Divide

Concur

“Solve-if-trivial” sub-engine
Broad scope of applications

Sparse Spectrum (DFT/WHT)
- Fast neighbor discovery for IoT (group testing) [Lee, Pedarsani, R., 2015]
- Sparse mixed linear regression [Yin, Pedarsani, Chen, R., 2016]
- Compressive phase retrieval
- Compressed sensing [Ocal, Li, R., 2016]

Sparse-graph codes
- Sub-Nyquist sampling theory
- Li, Pawar, R., 2014
- Compressed sensing
- Li, Pawar, R., 2014
- Pawar, R., 2013
- Li, Pawar, R., 2014
- Fast neighbor discovery for IoT (group testing) [Lee, Pedarsani, R., 2015]
Compressed sensing

Estimate the $K$-sparse signal $x \in \mathbb{C}^N$, which has only $K \ll N$ non-zero coefficients, from linear measurements in the presence of noise

$$y = Ax + w$$

Methods based on convex relaxation

- Measurement matrix $A$ has random design (e.g., random Gaussian matrix)
- Solve the the convex optimization problem
  Minimize $\|Ax - y\| + \lambda \|x\|_1$
- Measurements: $O \left( K \log \frac{N}{K} \right)$
- Computations: $O(\text{poly}(N))$

Candes 2006, Donoho 2006
Compressed sensing

Estimate the $K$-sparse signal $\mathbf{x} \in \mathbb{C}^N$, which has only $K \ll N$ non-zero coefficients, from linear measurements in the presence of noise

$$\mathbf{y} = \mathbf{A} \mathbf{x} + \mathbf{w}$$

Recovery w.h.p. using

**Noiseless:** $O(K)$ samples, $O(K)$ computations

**Noisy:** $O(K \log N)$ samples, $O(K \log N)$ computations

Li, Pawar, R., 2014 – Yin et al., 2019
Generic method to make algorithm robust to noise

Recall how we find locations and values of singletons in the noiseless setting. Ex.: a singleton with non-zero element $b$ at index 4

\[
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix} =
\begin{bmatrix}
r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 \\
r_1 & r_2W & r_3W^2 & r_4W^3 & r_5W^4 & r_6W^5 & r_7W^6 & r_8W^7
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
b
\end{bmatrix} =
\begin{bmatrix}
r_4 \\
r_4W^3b
\end{bmatrix}
\]

Location information is encoded in the **relative phase** between $y_2$ and $y_1$.

- What if we have $y_1 = r_4 + w_1$ and $y_2 = r_4W^3b + w_2$?
- $\angle \left( \frac{y_2}{y_1} \right) =$?
Generic method to make algorithm robust to noise

It is **not** robust to encode the **location** information in the relative phase! Alt. choice?

1. Represent each element by its binary index string: \((\log N)\)
2. Encode it using an error correcting code matched to the noise of the channel: \((C_1 \log N)\)
3. Add a unique random signature vector to each column to identify the element the column represents: \((C_2 \log N)\).
4. Total cost (per measurement bin) is \(O(\log N)\).
5. No. of measurement bins is \(O(K)\) (using sparse graph codes).
6. Total measurement cost is \(O(K \log N)\).

Guess-and-check algorithm:

A. **Guess** that a received bin measurement corresponds to a **singleton**.
B. Find ML estimate of singleton **value and location** index (using coded representation).
C. **Verify** using signature vector if singleton hypothesis is correct.
D. If **yes**, “peel” singleton node from the other measurement bins it belongs to, and continue.
E. If **no**, continue to next measurement bin.
Broad scope of applications

Sparse-graph codes

- Sparse Spectrum (DFT/WHT)  [Pawar, R., 2013] [Li, Pawar, R., 2014]
- Fast neighbor discovery for IoT (group testing)  [Lee, Pedarsani, R., 2015]
- Compressive phase retrieval  [Pedarsani, Lee, R., 2014]
- Sparse mixed linear regression  [Yin, Pedarsani, Chen, R., 2016]
- Sub-Nyquist sampling theory  [Ocal, Li, R., 2016]

Compressed sensing  [Li, Pawar, R., 2014]
Compressive Phase Retrieval (CPR)

Recover a **$K$-sparse** signal $x \in \mathbb{C}^n$ from **magnitude** measurements:

$$y = |Ax| + w,$$

where $A \in \mathbb{C}^{m \times n}$ is the measurement matrix.

Nonlinearity makes peeling challenging.
Main Results

• **Sparse-graph** codes for Compressive Phase Retrieval: **PhaseCode**

• **Fast & efficient:** first ‘capacity-approaching’ results

<table>
<thead>
<tr>
<th></th>
<th>Sample complexity</th>
<th>Computational complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noiseless</td>
<td>$4K$ (or $14K$)</td>
<td>$O(K)$</td>
</tr>
<tr>
<td>Noisy (almost-linear)</td>
<td>$O(K \log n)$</td>
<td>$O(N \log n)$</td>
</tr>
<tr>
<td>Noisy (sub-linear)</td>
<td>$O(K \log^3 n)$</td>
<td>$O(K \log^3 n)$</td>
</tr>
</tbody>
</table>

• Design can be made ‘**Optics-Friendly**’

• Extensive **simulations** validate close tie between theory & practice

*Pedarsani, Yin, Lee, R., 2014*
Simulation Results

\[ x = 2D \text{ FFT coefficients of} \]

\[
\begin{array}{c}
\text{A} \quad Ax \\
\text{A} \quad Ax \\
\text{Decoder} \quad \hat{x}
\end{array}
\]
Simulation Results

Iteration 0

IFFT of recovered FFT coefficients

Color of balls

RED = Not colored
Simulation Results

Iteration 1

Some balls are colored!
Simulation Results

Iteration 2

More balls are colored!
Simulation Results

Iteration 3
Simulation Results

Iteration 4
Simulation Results

Iteration 5
Simulation Results

Iteration 6
Simulation Results

Iteration 7
Simulation Results

Iteration 8
Simulation Results

Iteration 9
Simulation Results

Iteration 10

GREEN becomes dominant?
Simulation Results

Iteration 11

Most balls are **GREEN**
Simulation Results

Iteration 12

All but 1 ball are GREEN
Broad scope of applications

Sparse graphs

- Compressive phase retrieval
- Sparse mixed linear regression
- Compressed sensing
- Compressed sensing
- Sparse Spectrum (DFT/WHT)
- Fast neighbor discovery for IoT (group testing)
- Sub-Nyquist sampling theory

Contributions:

- Pawar, R., 2013
- Li, Pawar, R., 2014
- Ocal, Li, R., 2016
- Li, Pawar, R., 2014
- Pedarsani, Lee, R., 2014
- Lee, Pedarsani, R., 2015
- Yin, Pedarsani, Chen, R., 2016
Find $K$ defective from $n$ items using ‘group’ measurements

[85] Principles of group testing and an application of the design and analysis of multi-access protocols
[85] Born again group testing: multi-access communications
[84] Random multiple-access communications and group testing
Group testing

Find **K defective** from **n items** using ‘group’ measurements
Group Testing for Neighbor Discovery

node1

node2

node3
Group Testing for Neighbor Discovery

node1
node2
node3

<table>
<thead>
<tr>
<th></th>
<th>node1</th>
<th>node2</th>
<th>node3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

* 0.9 * 0.5 * 0.1
Group Testing for Neighbor Discovery

node1

node2

node3

Energy detection

1 0 1 0 1

node1 1 0 1 0 0 * 0.5
node2 1 0 1 0 0 * 0.9
node3 0 0 1 0 1 * 0.1
Group Testing for Neighbor Discovery

Node1, Node2, and Node3 are neighbors!
SAFFRON
(Sparse-grAph codes Framework For gROup testiNg)

Thm: With $6C(\epsilon)K \log_2 n$ tests, SAFFRON recovers at least $(1 - \epsilon)K$ defective items with probability $1 - O\left(\frac{K}{n^2}\right)$ by performing $O(K \log n)$ computations.

Example: SAFFRON ($\epsilon = 10^{-6}, C(\epsilon) = 11.3$)

With $68K \log_2 n$ tests, SAFFRON recovers at least $(1 - 10^{-6})K$ defective items with probability $1 - O\left(\frac{K}{n^2}\right)$ with a decoding time complexity of $O(K \log n)$.

Lee, Pedarsani, Chandrasekher, R., 2015
SAFFRON
(Sparse-grAph codes Framework For gROup testiNg)

Simulation done on a regular MacBook Air laptop

Run-time with $K = 2^5$ and varying $n$.

Finding 32 defective items from a population of size 1 trillion can be done with SAFFRON using ~87,000 tests in 0.3 second on a regular MacBook Air laptop!
Peeling with OR operation

Under field arithmetic

1  
1  

Under OR

1 OR 1 = 1

1  
1  

1  
1  

1  
1  

1  
1  

Challenge: Peeling with OR operation

Find singleton measurement/test and recover the value
Peeling with OR operation

Under field arithmetic

\[2 - 1 = 1\]

Under OR

1 OR 1 = 1

depends on other neighbors

Nonlinearity makes peeling challenging
Solution – high level idea

Binary expansion of subject index

subject 2 = (10)_2

subject 3 = (11)_2
Solution – high level idea

Binary expansion of subject index

subject 2 = (10)_2

subject 3 = (11)_2

Complement
Solution – high level idea

Binary expansion of subject index

subject 2 = (10)₂

subject 3 = (11)₂

Complement
Broad scope of applications

Sparse-graph codes

- Compressive phase retrieval
- Sparse mixed linear regression
- Compressed sensing
- Sub-Nyquist sampling theory
- Fast neighbor discovery for IoT (group testing)

Pawar, R., 2013
Li, Pawar, R., 2014
Lee, Pedarsani, R., 2015
Ocal, Li, R., 2016
Li, Pawar, R., 2014

Pedarsani, Lee, R., 2014
Yin, Pedarsani, Chen, R., 2016
Motivation

- Compressive sensing: a powerful tool for sparse recovery.
- What if we have a mixture of sparse signals?
- Applications: Neuroscience, experiment design in biology…
Problem Formulation

- \( L \)-class mixture of sparse linear regressions.
- Sparse parameter vectors \( \beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(L)} \in \mathbb{C}^n \)
- Total number of non-zero elements \( K (\ll Ln) \)
- Design query vectors \( x_1, x_2, \ldots, x_m \in \mathbb{C}^n \).
- Obtain measurements \( y_i = \langle x_i, \beta^{(\ell)} \rangle + w_i \) with probability \( q_\ell \).
- No knowledge of which \( \beta^{(\ell)} \) is associated with each measurement.

Simultaneous \textit{de-mixing} and sparse parameter \textit{estimation} problem!
Problem Formulation
Problem Formulation

query vectors
Problem Formulation
Problem Formulation
Problem Formulation

Goal:
- Output accurate estimates $\hat{\beta}^{(1)}, \hat{\beta}^{(2)}, \ldots, \hat{\beta}^{(L)}$.
- Minimize sample and time complexities.

Can we get sample and time complexities sublinear in $n$?
Related Work (incomplete list)


Sparse mixed linear regression: main results

**Mixed-Coloring algorithm**

For any fixed $p^* \in (0,1)$, for $m = \Theta(K)$, the Mixed-Coloring algorithm satisfies these properties for each $\ell \in [L]$:

- No false discovery
- Recover $1 - p^*$ fraction of the support of each $\beta^{(\ell)}$ w.p. $1 - O(1/K)$.
- Recovered support is uniform
- Time complexity: $\Theta(K)$ (optimal)

*Yin, Pedarsani, Chen, R., 2017*
Main Results

- Precise characterization of the constants in the sample complexity.

<table>
<thead>
<tr>
<th>$L$</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p^*$</td>
<td>$5.1 \times 10^{-6}$</td>
<td>$8.8 \times 10^{-6}$</td>
<td>$8.1 \times 10^{-6}$</td>
</tr>
<tr>
<td>$m = CK$</td>
<td>$33.39K$</td>
<td>$37.80K$</td>
<td>$40.32K$</td>
</tr>
</tbody>
</table>

- Time complexity: $\Theta(K)$ (optimal)

- $C = \Theta(\log \frac{1}{\epsilon})$. 

$L$: # of parameter vectors
$K$: sparsity
$p^*$: error floor
$m$: # of measurements
Primitives

Summation check:
- Goal: find measurements generated by the same $\beta^{(\ell)}$
- Generate $x_1, x_2 \in \mathbb{C}^n$ from some continuous distribution
- Generate the third vector of the form $x_3 = x_1 + x_2$
- Get measurements $y_1, y_2, y_3$
- $y_3 = y_1 + y_2$?
- If so, the three measurements come from the same $\beta^{(\ell)}$
- Consistent pair $(y_1, y_2)$

Ratio test
- Find location of a singleton

Peeling
- Remove contribution to other measurements
Decoding Algorithm

• Find consistent pairs
Singleton balls
At this stage, we have got some non-zero elements but we don’t know which parameter vectors they belong to.

Decoding Algorithm

- Find consistent pairs
- Find singletons
Strong doubletons: consistent pairs that are only associated with two singleton balls found in the first stage.
Can be found by guess-and-check.
The two singleton balls must be in the same parameter vector.
Decoding Algorithm

- Find consistent pairs
- Find singletons
- Find strong doubletons

**Theorem:** As long as $M/K > \text{const.}$, the $L$ largest connected components of the graph are of size $O(K)$, and correspond to different parameter vectors. Other connected components are of size $O(\log K)$.

*[Follows from E-R $(n,p)$ random graphs: if $np>1$, then component size is $O(n)$, else it is $O(\log n)$.]*
Decoding Algorithm

- Find consistent pairs
- Find singletons
- Find strong doubletons
- Recover a subset of size $\Theta(K)$
Decoding Algorithm

- Find consistent pairs
- Find singletons
- Find strong doubletons
- Recover a subset of size $\Theta(K)$
- Iterative decoding
Decoding Algorithm

Iterative decoding:

Non-zero elements from two parameter vectors, either blue or red.

Consistent pairs (bins). Each bin is either blue or red.
Decoding Algorithm

Iterative decoding:
Decoding Algorithm

Iterative decoding:

By finding strong doubletons and largest connected components, we have already **recovered** a fraction of non-zero elements. Say $a, b$ (blue) and $u, v$ (red).
Decoding Algorithm

Iterative decoding:

By finding strong doubletons and largest connected components, we have already *recovered* a fraction of non-zero elements. Say $a, b$ (blue) and $u, v$ (red).
Decoding Algorithm

Iterative decoding:

Guess-and-check: try to subtract $a$ and $b$ from bin 1, and $v$ from bin 3.
Decoding Algorithm

Iterative decoding:

The remaining measurements pass ratio test. Recover $c$ using bin 1 and recover $w$ using bin 3.
Decoding Algorithm

Iterative decoding:

Iterate this procedure and recover all the non-zero elements.
Decoding Algorithm

Density evolution:

- Consider one particular parameter vector $p_j$.
- $p_j$: the fraction of non-zero elements that are not recovered after the $i$-th iteration.

- Summation starts from 2 because singletons are not useful for iterative decoding as we don’t know their “color.”

$$p_{j+1} = f(p_j) = \left(1 - \sum_{i=2}^{\infty} \rho_i (1 - p_j)^{i-1}\right)^{d-1}$$

- $p_j$ can converge to an arbitrarily small constant.

- Bad news: $p_0 = 1$ is a fixed point!
- Good news: If we can start at $p_0 = 1 - \delta$, we are good to go!
Experimental Results

Noiseless setting: sample and time complexities:

- Optimal parameters \((d, R, V)\) computed from density evolution.
- Success: exact recovery of all non-zero elements.
- Empirical success probability/average running time over 100 trials.

<table>
<thead>
<tr>
<th>(L)</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p^*)</td>
<td>(5.1 \times 10^{-6})</td>
<td>(8.8 \times 10^{-6})</td>
<td>(8.1 \times 10^{-6})</td>
</tr>
<tr>
<td>(m = CK)</td>
<td>(33.39K)</td>
<td>(37.80K)</td>
<td>(40.32K)</td>
</tr>
</tbody>
</table>

\[n = 10^5\]

\[\text{Matches our theory}\]
Generic method to make algorithm robust to noise

Recall how we find locations and values of singletons in the noiseless setting. Ex.: a singleton with non-zero element $b$ at index 4

\[
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix} =
\begin{bmatrix}
r_1 & r_2 & r_3 & r_4 \circled{r_4 W^3} & r_5 & r_6 & r_7 & r_8 \\
r_1 & r_2 W & r_3 W^2 & r_4 W^3 & r_5 W^4 & r_6 W^5 & r_7 W^6 & r_8 W^7
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
b \\
0 \\
0 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
r_4 \\
0 \\
0 \\
0 \\
r_4 W^3 b
\end{bmatrix}
\]

Location information is encoded in the \textit{relative phase} between $y_2$ and $y_1$.

- What if we have $y_1 = r_4 + w_1$ and $y_2 = r_4 W^3 b + w_2$?
- $\angle \left( \frac{y_2}{y_1} \right) =$?
Robust Mixed-Coloring Algorithm

It is not robust to encode the location information in the relative phase! Alternative choice?

We can also encode the location information in the relative phase! What if

\[
\begin{bmatrix}
y_1 \\
y_2 \\
y_3
\end{bmatrix} = \begin{bmatrix}
0 + w_1 \\
b + w_2 \\
b + w_3
\end{bmatrix}
\]

- It is still possible to recover the binary pattern of the measurements by a simple thresholding.
- Of course we may make mistakes.
- This procedure can be robustified by simply repeating each bit or using an error correcting code.
Experimental Results

Noisy setting: sample and time complexities:

- $\Delta = 1, \sigma = 0.2$.
- Record the minimum number of measurements to achieve 100 consecutive success.
- Sublinear sample and time complexities.

![Graphs showing sample and time complexities for different $K$ values.](image-url)
Chapter 5

Speeding up distributed computing on the cloud
System Noise

Network bottlenecks

HW failures

Maintenance, etc.
System Noise = Latency Variability

Computing $f(A)$...
Completed in 1s.
System Noise = Latency Variability

Computing $f(A)$…
Completed in 1s.

Computing $f(A)$…
Still computing…
Still…
Completed in 3s.
Distributed Matrix-Vector Multiplication

\[ A \times b \]
Distributed Matrix-Vector Multiplication

\[ A \times b = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b \]
Distributed Matrix-Vector Multiplication

\[ A \times b = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b = \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix} \]

Master

Worker 1

Worker 2

Worker 3
Distributed Matrix-Vector Multiplication

\[ A \times b \]
\[ = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b \]
\[ = \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix} \]
\[ := \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \]
Distributed Matrix-Vector Multiplication

\[ A \times b \]
\[ = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b \]
\[ = \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix} \]
\[ := \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \]
Straggler Problem

CDF of time to collect results from k workers

The slowest worker can be 6 times slower

[Chen et al., 2016]

Google Brain
Why Do We Have Stragglers?

“… infeasible to eliminate all latency variability.”
[Dean, Barroso, Comm. ACM'2013]

1. Data Locality
2. Shared Resources
3. Network Latency

Worker 1: \( A_1 b \)
Worker 2: \( A_2 b \)
Worker 3: \( A_3 b \)
Coded Matrix-Vector Multiplication

\[ A \times b \]
\[ = \left( \begin{array}{c}
A_1' \\
A_2'
\end{array} \right) \times b \]
\[ = \left( \begin{array}{c}
A_1' \times b \\
A_2' \times b
\end{array} \right) \]
\[ := \left( \begin{array}{c}
y_1' \\
y_2'
\end{array} \right) \]

\[ A'_3 := A'_1 + A'_2 \]
\[ y'_3 := y'_1 + y'_2 \]
Coded Computation for Linear Operations

Assumptions:
- n workers
- k subtasks
- **Computing time of each worker: constant + exponential RV (i.i.d.)**
- Average computing time is proportional to $1/k$

**Theorem:**

\[
E[T_{\text{uncoded}}] = \Theta \left( \frac{\log n}{n} \right)
\]

\[
E[T^{*}_{\text{replication}}] = \Theta \left( \frac{\log n}{n} \right)
\]

\[
E[T^{*}_{\text{MDS-coded}}] = \Theta \left( \frac{1}{n} \right)
\]

Lee, Lam, Pedarsani, Papailiopoulos, R. 2015
MDS-Coded Matrix-Vector Multiplication

\[ \Theta \left( \frac{\log n}{n} \right) \quad \Theta \left( \frac{1}{n} \right) \]

opt. rep. \quad \text{coded} \quad \text{vector multiplication}

Under exponential latency model

On Amazon AWS

Codes provide 30% speedup compared uncoded and replicated jobs for fixed number of workers

[LLPPR, NIPS workshop ’15]
[LLPPR, T-IT’18]
Applications

• Distributed linear regression
• Distributed non-linear function computation
• Reducing communication in data shuffling by network coding

Has attracted lots of interest:

• Coded Matrix Multiplication in MapReduce setup
• Coded Computation for Logistic Regression
• Coded Computation + Distributed Gradient Computing
• Approximation: SVD + Coded Matrix Multiplication, Sketching, Second order methods...
Coded Computation [LLPPR, NIPS W’15]
[LLPPR, ToIT’18]

• A new interface between ML systems and information & coding theory

• Codes can be used to speed up distributed computation & distributed ML
  • Matrix-vector multiplication [LLPPR, ToIT’18],
  • Matrix-matrix multiplication [LSR, ISIT’17], [BLOR, ISIT’18], [GWCR, BG’19]
  • Gradient accumulation [LPPR, ISIT’17], [GKCMR, ICML Workshop’19]
  • Data shuffling [CLPPR, NeurIPS W’17], [CLPPR, SysML’18]

• Works in practice (Amazon EC2 experiments on real data)
Coded Computation

- **Matrix-vector multiplication** [LLPPR, ToIT’18]
  - [Ferdinand and Draper, Allerton’16]
  - [Reisizadeh et al., ISIT’17]
  - [Malick, Chaudhari, Joshi, '18]
  - [Wang, Liu, Shroff, ICML’18]
  - [Maity, Rawat, Mazumdar, SysML’18]
  - …

- **Matrix-matrix multiplication** [LSR, ISIT’17], [BLOR, ISIT’18] [GWCR, BG’19]
  - [Yu, Maddah-Ali, Avestimehr, NIPS’17]
  - [Dutta et al., ’18]
  - …

- **Gradient accumulation** [LLPPR, ISIT’17] [GKCMR, ICML Workshop’19]
  - [Dutta, Cadambe, Grover, NIPS’16]
  - [Tandon, Lei, Dimakis, Karampatziakis, ICML’17]
  - [Raviv, Tamo, Tandon, Dimakis, ’17]
  - [Halbawi, Azizan, Salehi, Hassibi, ISIT’18]
  - [Ye and Abbe, ICML’18]
  - [Charles and Papailiopoulos, ISIT’18]
  - …

- **Data shuffling** [CLPPR, NeurIPS W’17], [CLPPR, SysML’18]
  - [Song et al., ISIT’17]
  - [Attia and Tandon, Globecom’16]
  - …
Scalable computing: **Serverless** platform!

- A decade ago, **cloud servers** abstracted away **physical servers**.
- Future: “**serverless**” computing will abstract away **cloud servers**.

- “Function as a Service (FaaS)”
  - Run my function “somewhere”
  - AWS, Google, IBM, Microsoft, etc.

**Why Serverless computing?**

- Simple abstraction for user
  - Cluster management hidden
- Tremendous scale
  - 16,000 machines in 10 seconds
  - Cloud storage as infinite RAM
- Reduced Costs
  - Pay only for the time you use
- Significant interest from the cloud computing community

---

Serverless Systems: Characteristics

- Massive scale of low quality workers
- Workers do not communicate
  - Read/write data through a single data storage entity
- Workers are short-lived
- Stragglers and faults!

A single run snapshot

Average Runtimes over 10 trials

Can have up to 16,000 workers on AWS Lambda
What are we optimizing for?

- Matrix multiplication is a black box
- MDS is beneficial, but target only $T_{comp}$
- *End to end latency* is desired metric

**Product Codes:** a good tradeoff between near-MDS and local enc./dec.

**G-LDPC codes** [Tanner ‘81, Lentmaier-Z’99, Boutros et al. ‘99], **Product codes** [Elias ‘54, Justeson ’07, JENR ‘15]

\[
\begin{pmatrix}
A_1 \\
A_2 \\
A_1 + A_2
\end{pmatrix}
\times
\begin{pmatrix}
B_1 & B_2 & B_1 + B_2
\end{pmatrix}
\]

\[
= \begin{pmatrix}
A_1 B_1 & A_1 B_2 \\
A_2 B_1 & A_2 B_2 \\
(A_1 + A_2)B_1 & (A_1 + A_2)B_2
\end{pmatrix}
\]

1. Near-MDS
2. Low ENC/DEC cost
3. DEC is parallelizable
4. N-dim product codes...
Product Code Decoding

- Peeling decoder is very simple and parallelizable
Product Code Decoding

- Peeling decoder is very simple and parallelizable
Product Code Decoding

- Peeling decoder is very simple and parallelizable
Product Code Decoding

- Peeling decoder is very simple and parallelizable
Product Code Decoding

- Peeling decoder is very simple and parallelizable
Product-Coded MM: Performance

**Result:** (Baharav & R’18) In a $d$-dimensional product-coded matrix multiplication scheme with $(n, k, r+1)$ component codes, the output will be decodable w.h.p. after $K' = N - \frac{N-K}{\eta(d,r)}$ nodes have completed their subtasks.

- Can tolerate $\frac{N-K}{\eta(d,r)}$ stragglers

<table>
<thead>
<tr>
<th>Table II: Thresholds: $\eta(d,r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
Kernel Ridge Regression using Conjugate Gradient on AWS Lambda

On a real-world dataset with $n = 0.4$ million examples and 400 workers

Problem:
Solve for $x$ in $(K + \lambda I)x = y$, $K$: Kernel matrix of dim. $n$

Initialization:
$x_0 = 1^{n \times 1}, r_0 = y - (K + \lambda I)x_0, p_0 = r_0, k = 0$

YES

Compute in parallel: $h_k = (K + \lambda I)x_k$

$k = k + 1$

NO

Update locally

$k_0 = \frac{v_k^T r_k}{h_k^T h_k}$

$x_{k+1} = x_k + k_0 p_k$

$r_{k+1} = r_k - k_0 h_k$

$\beta_k = \frac{r_{k+1}^T r_{k+1}}{h_k^T r_k}$

$p_{k+1} = r_{k+1} + \beta_k p_k$

Reduced variation in iteration times, improved reliability

(First iteration includes the one-time encoding cost)
Power Iteration on serverless AWS Lambda

- Goal: Find the largest eigenvalue and eigenvector of a diagonalizable matrix $A$ of dimension 0.5 million with 1000 workers
- Applications: PCA, PageRank; Twitter recos. on whom to follow
- Each iteration: a matrix-vector multiplication $b_{k+1} = \frac{A b_k}{\|A b_k\|}$

~50% savings in total time! (1 hour 6 min. less)
Matrix Multiplication: Sketching

- Exact computation is not necessary, especially if input data has redundancies.
- Randomized sketching is an important technique to reduce computation complexity.
- To compute $AA^T$
  - Sketch the input matrix: $\tilde{A} = AS$
    
    \[
    \begin{array}{ccc}
    \mathbf{A} & \times & \mathbf{S} \\
    d \times n & & n \times m \\
    \end{array}
    \]
    \[
    = \begin{array}{c}
    \tilde{A} \\
    d \times m \\
    \end{array}
    \]
    (m ≪ n)
  - $S$ is a random matrix such that $SS^T$ is close to identity.
  - Multiply the smaller matrices $\tilde{A}$ and $\tilde{A}^T$

Mahoney, M. W., 2011; Woodruff, D. P., 2014; Drineas et al., 2016; .......
Recall the challenges in serverless systems:

- Slow communication
- Ephemeral workers
- Persistent stragglers

Hence, reducing the number of iterations is paramount

- Second-order methods are a natural fit for serverless systems
  - Reduce the number of iterations considerably
  - Exploit the tremendous compute power per iteration
- OverSketched Newton: Tailored to serverless systems
OverSketched Newton

Key Observation: For many common convex optimization problems
• Gradient can be written as a few large matrix-vector mults.
• Hessian can be written as a large matrix-matrix multiplication

Example problems:
- Logistic and linear regression,
- Softmax regression,
- SVMs,
- Linear program,
- Semidefinite programs,
- Lasso (in dual formulation), etc.
Example: Logistic Regression

\[
\min_{w \in R^d} \left\{ f(w) = \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i w^T x_i} \right) + \frac{\lambda}{2} \|w\|^2 \right\}
\]

- \( X = [x_1, \ldots, x_n] \in R^{d \times n} \) is the matrix containing training examples
- \( y = [y_1, \ldots, y_n] \in R^n \) is vector containing training labels

- Gradient is given by
  \[
  \nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i}{1 + e^{y_i w^T x_i}} + \lambda w
  \]

- Can be written as matrix-vector products
  \[
  \nabla f(w) = X \beta + \lambda w, \text{ where } \beta_i = \frac{-y_i}{1 + e^{y_i \alpha_i}}, \text{ and } \alpha = X^T w
  \]

- Hessian is given by
  \[
  H^T = \frac{1}{n} X \Lambda X^T + \lambda I_d \in R^{d \times d}
  \]
  - \( \Lambda \) is diagonal, \( \Lambda(i, i) = \frac{-y_i}{1 + e^{y_i \alpha_i}} \)

- Requires computation of \( A A^T \), where
  \[
  A = X \sqrt{\Lambda} \in R^{d \times n}, n \gg d
  \]
OverSketched Newton

- Compute the gradient using classical coded computing
- Compute the Hessian approximately by “over sketching”

\[ A \times S^{T} \]

- Model update: \( w^{t+1} = w^{t} - \hat{H}^{-1} g \)
  - Can be done locally if \( d \) small enough

We prove convergence guarantees for OverSketched Newton when the objective is both strongly and weakly convex
Comparison with existing second-order methods

Experiments with $n = 0.3$ million examples and $d = 3000$ features on AWS Lambda

- GIANT: Linear-quadratic convergence when $n \gg d$
- 60 workers used for Gradient
- 3600 workers used to compute the exact Hessian
- 600 workers used to compute the sketched Hessian

Coded computing vs Recomputing Stragglers

Experiments on logistic regression with n = 0.4 million and d = 2000

**Newton-type methods on EPSILON dataset**

- Exact Hessian (with recomputed gradient)
- Exact Hessian (with coded gradient)
- OverSketched Newton (with recomputed gradient)
- OverSketched Newton (with coded gradient)

**Codes used?**

<table>
<thead>
<tr>
<th>Gradient</th>
<th>Hessian</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

**Training Error vs Time (seconds)**

- 0.3
- 0.35
- 0.4
- 0.45
- 0.5
- 0.55
- 0.6
- 0.65

- 0
- 100
- 200
- 300
- 400
- 500
- 600
- 700
First order vs Second order on AWS Lambda

Experiments on a EPSILON dataset with $n = 0.4$ million ex. and $d = 2000$ features

- 100 workers used for Gradient computation
- 1500 workers used to compute the sketched Hessian
Experiments on logistic regression with $n = 0.3 \text{ million}$ and $d = 3000$

MPI (server-based) vs Serverless computing

GIANT on Amazon EC2 (server-based) vs OveSketched Newton on AWS Lambda
Concluding Remarks

Shannon-inspired research threads on the power of codes in:

• **Duality:**
  – “exchangability” of enc. and dec. functions in source/channel coding

• **Encryption:**
  – “exchangability” of encryption & compression modules w/o perf. loss

• **Sampling:**
  – unexplored connections between sampling theory and coding theory

• **Learning:**
  – sparse-graph code based “peeling” core powerful in many sparse
    learning settings with sub-linear time complexity

• **Distributed computing:**
  – straggler-proofing with codes speeds up distributed machine learning
Conclusion: Shannon’s incredible legacy

- A mathematical theory of communication
- Channel capacity
- Source coding
- Channel coding
- Cryptography
- Sampling theory
- ...

His legacy will last many more centuries!

(1916-2001)
Thank you!