Erasures vs. Errors in Local Decoding and Property Testing

Sofya Raskhodnikova*, Nithin Varma*, Noga Ron-Zewi#

*Boston University      #University of Haifa

ITCS 2019
Goals

Study the role of erasures in local decoding
Goals

- Study the role of erasures in local decoding
- Use our understanding to separate erasures and errors in property testing
Overview of Results

• **Local list decoding in the presence of erasures**
  – Local list erasure-decoding **Hadamard Code**
  – **Constant vs.** $\Omega(\log n)$ separation between erasure-resilient testing and tolerant testing
Overview of Results

• Local list decoding in the presence of erasures
  – Local list erasure-decoding Hadamard Code
  – Constant vs. $\Omega(\log n)$ separation between erasure-resilient testing and tolerant testing

• Approximate local list decoding in the presence of erasures
  – Constant vs. $n^{\Omega(1)}$ separation between erasure-resilient testing and tolerant testing
Overview of Results

• Local list decoding in the presence of erasures
  – Local list erasure-decoding Hadamard Code
  – Constant vs. $\Omega(\log n)$ separation between erasure-resilient testing and tolerant testing

• Approximate local list decoding in the presence of erasures
  – Constant vs. $n^{\Omega(1)}$ separation between erasure-resilient testing and tolerant testing

• Relationship between local decoding in the presence of erasures and in the presence of errors
Locally (Unique or List) Decodable Codes

• **Locally decodable codes** [Babai Fortnow Levin Szegedy 91, Gemmel Lipton Rubinfeld Sudan Wigderson 91, Gemmel Sudan 92, Blum Luby Rubinfeld 93, Polishchuk Spielman 94, Beimel Ishai Kushilevitz Raymond 02, Yekhanin 08, Ben-Aroya Efremenko TaShma 10, Dvir Gopalan Yekhanin 11, Efremenko 12, …]

  – Each message bit can be decoded with high probability by querying a few bits of the codeword
Locally (Unique or List) Decodable Codes

• **Locally decodable codes** [Babai Fortnow Levin Szegedy 91, Gemmel Lipton Rubinfeld Sudan Wigderson 91, Gemmel Sudan 92, Blum Luby Rubinfeld 93, Polishchuk Spielman 94, Beimel Ishai Kushilevitz Raymond 02, Yekhanin 08, Ben-Aroya Efremenko TaShma 10, Dvir Gopalan Yekhanin 11, Efremenko 12, …]
  
  – Each message bit can be decoded with high probability by querying a few bits of the codeword

• **Locally list decodable codes** [Goldreich Levin 89, Sudan Trevisan Vadhan 01, Gutfreund Rothblum 08, Gopalan Klivans Zuckerman 08, Ben-Aroya Efremenko Ta-Shma 10, Kopparty Saraf 13, Kopparty 15, Hemenway Ron-Zewi Wootters 17, Goi Kopparty Oliveira Ron-Zewi Saraf 17, Kopparty Ron-Zewi Saraf Wootters 18, …]
  
  – Given oracle access to codeword, with high probability, obtain a list of descriptions of local decoders of each candidate message
  
  – Decodes from a larger fraction of corruptions
# Locally (Unique or List) Decodable Codes

## Locally decodable codes

[Babai Fortnow Levin Szegedy 91, Gemmel Lipton Rubinfeld Sudan Wigderson 91, Gemmel Sudan 92, Blum Luby Rubinfeld 93, Polishchuk Spielman 94, Beimel Ishai Kushilevitz Raymond 02, Yekhanin 08, Ben-Aroya Efremenko TaShma 10, Dvir Gopalan Yekhanin 11, Efremenko 12, …]

- Each message bit can be decoded with high probability by querying a few bits of the codeword

## Locally list decodable codes

[Goldreich Levin 89, Sudan Trevisan Vadhan 01, Gutfreund Rothblum 08, Gopalan Klivans Zuckerman 08, Ben-Aroya Efremenko Ta-Shma 10, Kopparty Saraf 13, Kopparty 15, Hemenway Ron-Zewi Wootters 17, Goi Kopparty Oliveira Ron-Zewi Saraf 17, Kopparty Ron-Zewi Saraf Wootters 18, …]

- Given oracle access to codeword, with high probability, obtain a list of descriptions of local decoders of each candidate message
- Decodes from a larger fraction of corruptions

Can locally decodable codes perform better with erasures than with errors?
Local List Decoding of Hadamard Code

• Hadamard: $\{0,1\}^k \rightarrow \{0,1\}^{2^k}$; Hadamard($x$) = $(\langle x, y \rangle)_{y \in \{0,1\}^k}$
Local List Decoding of Hadamard Code

- Hadamard: $\{0,1\}^k \rightarrow \{0,1\}^{2^k}$; $\text{Hadamard}(x) = (\langle x, y \rangle)_{y \in \{0,1\}^k}$

<table>
<thead>
<tr>
<th>Type of corruptions</th>
<th>Corruption tolerance $\alpha$</th>
<th>List size, $\ell$</th>
<th>Number of queries, $q$</th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>$\alpha \in \left(0, \frac{1}{2}\right)$</td>
<td>$\Theta\left(\frac{1}{\left(\frac{1}{2} - \alpha\right)^2}\right)$</td>
<td>$\Theta\left(\frac{1}{\left(\frac{1}{2} - \alpha\right)^2}\right)$</td>
<td>[Goldreich Levin 89]</td>
<td>[Blinovsky 86, Guruswami Vadhan 10, Grinberg Shaltiel Viola 18]</td>
</tr>
</tbody>
</table>
Local List Decoding of Hadamard Code

- Hadamard: \( \{0,1\}^k \rightarrow \{0,1\}^{2^k} \); Hadamard\((x) = (\langle x, y \rangle)_{y \in \{0,1\}^k} \)

<table>
<thead>
<tr>
<th>Type of corruptions</th>
<th>Corruption tolerance ( \alpha )</th>
<th>List size, ( \ell )</th>
<th>Number of queries, ( q )</th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>( \alpha \in (0, \frac{1}{2}) )</td>
<td>( \Theta \left( \frac{1}{(\frac{1}{2} - \alpha)^2} \right) )</td>
<td>( \Theta \left( \frac{1}{(\frac{1}{2} - \alpha)^2} \right) )</td>
<td>[Goldreich Levin 89]</td>
<td>[Blinovsky 86, Guruswami Vadhan 10, Grinberg Shaltiel Viola 18]</td>
</tr>
<tr>
<td>Erasures</td>
<td>( \alpha \in (0,1) )</td>
<td>( O \left( \frac{1}{1 - \alpha} \right) )</td>
<td>( \Theta \left( \frac{1}{1 - \alpha} \right) )</td>
<td>new</td>
<td>Implicit in [Grinberg Shaltiel Viola 18]</td>
</tr>
</tbody>
</table>

An improvement in dependence on \( \alpha \) was suggested by Venkat Guruswami
Local List Decoding of Hadamard Code

• Hadamard: \( \{0,1\}^k \to \{0,1\}^{2^k} \); Hadamard\( (x) = (\langle x, y \rangle)_{y \in \{0,1\}^k} \)
• Impossible to decode when fraction of errors is \( \geq 1/2 \).

<table>
<thead>
<tr>
<th>Type of corruptions</th>
<th>Corruption tolerance ( \alpha )</th>
<th>List size, ( \ell )</th>
<th>Number of queries, ( q )</th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>( \alpha \in \left(0, \frac{1}{2}\right) )</td>
<td>( \Theta \left( \frac{1}{(\frac{1}{2} - \alpha)^2} \right) )</td>
<td>( \Theta \left( \frac{1}{(\frac{1}{2} - \alpha)^2} \right) )</td>
<td>[Goldreich Levin 89]</td>
<td>[Blinovsky 86, Guruswami Vadhan 10, Grinberg Shaltiel Viola 18]</td>
</tr>
<tr>
<td>Erasures</td>
<td>( \alpha \in (0,1) )</td>
<td>( O\left( \frac{1}{1 - \alpha} \right) )</td>
<td>( \Theta\left( \frac{1}{1 - \alpha} \right) )</td>
<td>new</td>
<td>Implicit in [Grinberg Shaltiel Viola 18]</td>
</tr>
</tbody>
</table>

**Erasure-decoding better than error-decoding**

An improvement in dependence on \( \alpha \) was suggested by Venkat Guruswami
Property Testing

Property Tester
[Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

Property = Set of all YES instances

1 1 3 3 5 5 7 7 9 9
sorted array

2 1 4 3 6 5 8 7 9 0

1/2-far from sorted

Two objects are at distance $\varepsilon = \text{they differ in an } \varepsilon \text{ fraction of places}$
Erasure-Resilient and Tolerant Property Testing

**Erasure-Resilient Property Tester**

[Dixit Raskhodnikova Thakurta Varma 16]

\[ \leq \alpha \] fraction of the input is erased adversarially

Can be completed to YES

Any completion is far from YES

Accept w.h.p.

Don’t care

Reject w.h.p.

\( \alpha \)-erasure-resilient \( \varepsilon \)-testing
### Erasure-Resilient and Tolerant Property Testing

<table>
<thead>
<tr>
<th>Erasure-Resilient Property Tester</th>
<th>Tolerant Property Tester</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Dixit Raskhodnikova Thakurta Varma 16]</td>
<td>[Parnas Ron Rubinfeld 06]</td>
</tr>
<tr>
<td>( \leq \alpha ) fraction of the input is erased adversarially</td>
<td>( \leq \alpha ) fraction of the input is erroneous</td>
</tr>
<tr>
<td>Can be completed to YES</td>
<td>YES</td>
</tr>
<tr>
<td>Any completion is far from YES</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>far from YES</td>
<td>(( \alpha ), ( \varepsilon ))-tolerant testing</td>
</tr>
</tbody>
</table>

#### \( \alpha \)-erasure-resilient \( \varepsilon \)-testing

- Accept w.h.p.
- Don’t care
- Reject w.h.p.

#### (\( \alpha \), \( \varepsilon \))-tolerant testing

- Accept w.h.p.
- Don’t care
- Reject w.h.p.
Relationships Between Models

- $\varepsilon$-testing
- $\alpha$-erasure-resilient $\varepsilon$-testing
- $(\alpha, \alpha + \varepsilon)$-tolerant testing
Relationships Between Models

Containments are strict:

- [Fischer Fortnow 05]: standard vs. tolerant
- [Dixit Raskhodnikova Thakurta Varma 16]: standard vs. erasure-resilient
- **Our Result**: erasure-resilient vs. tolerant
Our First Separation Result

**First Separation Theorem**

There is a property of $n$-bit strings that

- can be erasure-resiliently tested with constant query complexity,
- but requires $\Omega(\log n)$ queries for tolerant testing.
Our First Separation Result

First Separation Theorem

There is a property of $n$-bit strings that

- can be erasure-resiliently tested with constant query complexity,
- but requires $\Omega(\log n)$ queries for tolerant testing.

Main Tools

- Separation between erasures and errors in local list decoding Hadamard codes
- PCPs of proximity [BenSasson Goldreich Harsha Sudan Vadhan 06, Dinur Reingold 06, Dinur 07]

(~ PCPs for property testing problems)
Our First Separation Result

First Separation Theorem
There is a property of $n$-bit strings that
• can be erasure-resiliently tested with constant query complexity,
• but requires $\Omega(\log n)$ queries for tolerant testing.

Main Tools

Separation between erasures and errors in local list decoding Hadamard codes

PCPs of proximity [BenSasson Goldreich Harsha Sudan Vadhan 06, Dinur Reingold 06, Dinur 07]
(~ PCPs for property testing problems)

Error-tolerant testing is harder than erasure-resilient testing in general.
Strengthened Separation Result

Strengthened Separation Theorem

There is a property of $n$-bit strings that
- can be erasure-resiliently tested with constant query complexity,
- but requires $\Omega(\log n)$ $n^{\Omega(1)}$ queries for tolerant testing.

Main Tools

- Separation between erasures and errors in approximate local list decoding Hadamard codes
- PCPs of proximity [BenSasson Goldreich Harsha Sudan Vadhan 06, Dinur Reingold 06, Dinur 07]
  ($\sim$ PCPs for property testing problems)

Error-tolerant testing is much harder than erasure-resilient testing in general.
Errors and Erasures in Local Decoding

• Local decoding implies local erasure-decoding
  – locally decodable from at most an $\alpha$ fraction of errors $\Rightarrow$
    locally decodable from at most an $2\alpha$ fraction of erasures
  – Also holds for local list decoding and approximate local list decoding
Errors and Erasures in Local Decoding

• **Local decoding implies local erasure-decoding**
  – locally decodable from at most an $\alpha$ fraction of errors $\Rightarrow$
    locally decodable from at most an $2\alpha$ fraction of erasures
  – Also holds for local list decoding and approximate local list decoding

• **Local erasure-decoding implies local decoding** (up to some parameters)
  – locally decodable from at most an $\alpha$ fraction of erasures using $q$ queries $\Rightarrow$
    locally decodable from at most an $\alpha/O(q^2 \cdot 9^q)$ fraction of errors using $O(q \cdot 3^q)$ queries
Open Questions

• Even stronger separation between erasure-resilient and tolerant testing -- constant vs. linear?
• Separation between errors and erasures for a "natural" property?
• Constant-query, constant list size, local list erasure-decodable codes with inverse polynomial rate?

Thank you!