Some batch code properties of the simplex code

With algorithmic proof

Ago-Erik Riet
Joint work with Henk D.L. Hollmann, Karan Khathuria, and Vitaly Skachek

Zürich, 14 July, 2022

University of Tartu, Tartu, Estonia
Email: agoerik@ut.ee

Content

Contents of this talk:

- Introduction: data recovery, definitions of batch-type codes
- Simplex code as batch code
- Known results, conjectures, our new result
- A reformulation
- An algorithm for abelian groups
- Application to our problem
- Conclusions

Introduction

In this talk:

- All codes are binary.
- We use row vectors for code words.

A linear code of length n and dimension k is a k-dimensional subspace $C \subseteq \mathbb{F}_2^n$;

$k \times n$ generator matrix $\boldsymbol{G}$: rows are a basis for C.

Encoding: data $\boldsymbol{a} = (a_1, \ldots, a_k) \longrightarrow \boldsymbol{c} = (c_1, \ldots, c_n) = \boldsymbol{aG}$

Data recovery

Recovering data:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{g}_1^\top & \boldsymbol{g}_2^\top & \cdots & \boldsymbol{g}_k^\top \end{bmatrix}$$

If $\sum_{i \in I} \boldsymbol{g}_i = \boldsymbol{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)$,

that is, $\boldsymbol{G} \cdot \chi_I = \boldsymbol{e}_j^\top$, then $\boldsymbol{c} \cdot \chi_I = \boldsymbol{a} \cdot \boldsymbol{G} \cdot \chi_I = \boldsymbol{a} \cdot \boldsymbol{e}_j^\top$,

so $\sum_{i \in I} c_i = a_j$.

Serving a request

Definition

We say that a $k \times n$ matrix $\boldsymbol{G}$ can serve a request $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t$ of (not neccesarily distinct) vectors in $\mathbb{F}_2^k$ if there are **disjoint** *column index sets $I_1, \ldots, I_t$ such that the columns of $\boldsymbol{G}$ in I_j sum up to $\boldsymbol{r}_j$, that is, $\sum_{i \in I_j} \boldsymbol{g}_i = \boldsymbol{r}_j$, for $j = 1, \ldots, t$.*

Every individual request $\boldsymbol{r}$ is a request for the linear combination

$$\boldsymbol{a}\boldsymbol{r}^\top = a_1 r_1 + \cdots + a_k r_k$$

of the data symbols!

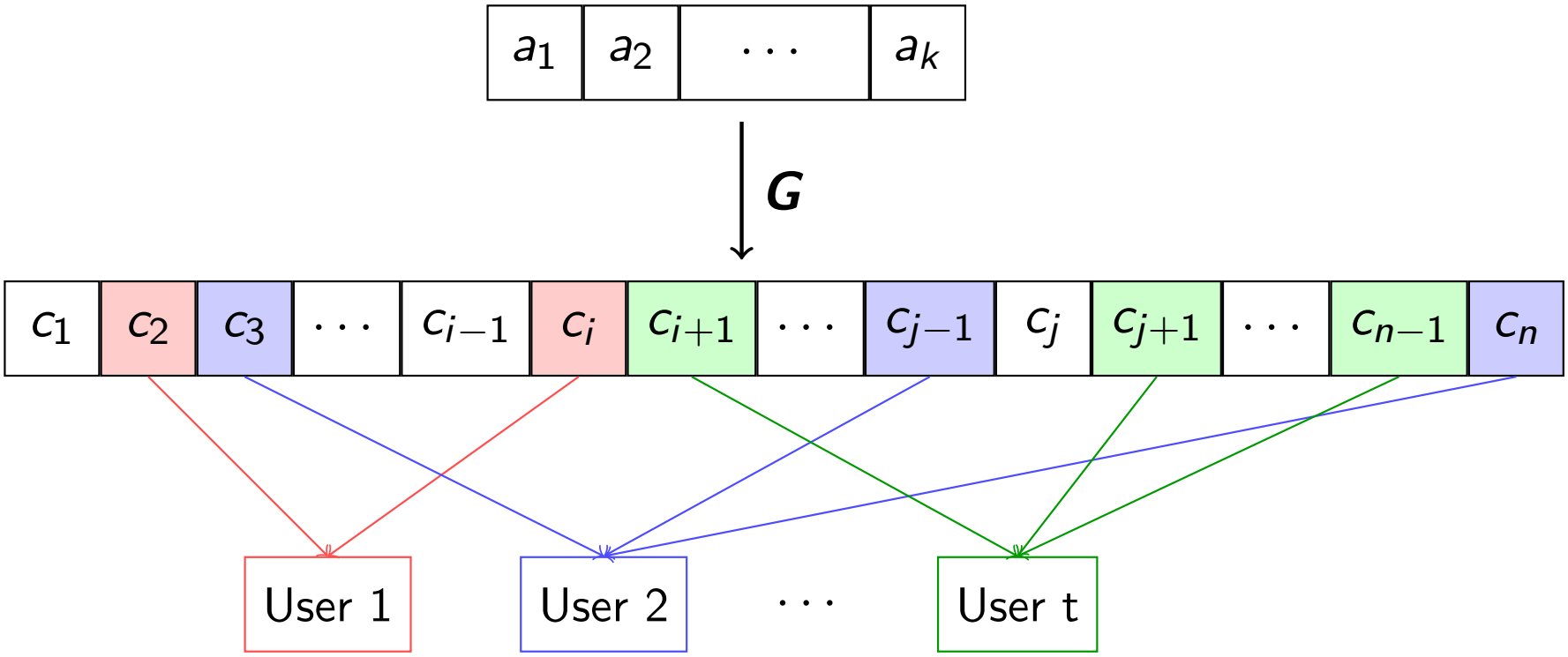
PIR/batch code definitions

An encoder (= a generator matrix $\boldsymbol{G}$) is

1. t-PIR 2. t-batch 3. t-odd-batch 4. t-functional-batch	if $\boldsymbol{G}$ can serve any request sequence consisting of	1. a t-fold repetition of a unit vector 2. t unit vectors 3. t vectors of odd weight 4. t arbitrary vectors

Proposition

A t-PIR code $\boldsymbol{G}$ generates a code with minimum distance at least t.


a_1 a_2 $\cdots$ a_k
G
c_1 c_2 c_3 $\cdots$ c_{i-1} c_i c_{i+1} $\cdots$ c_{j-1} c_j c_{j+1} $\cdots$ c_{n-1} c_n
User 1
User 2
$\cdots$
User t

Simplex code

Binary simplex code of length $n = 2^k - 1$ is the k-dimensional code with $k \times (2^k - 1)$ generator matrix

$$\boldsymbol{G}_k = \begin{bmatrix} 1 & 0 & 1 & 0 & & 0 & 1 \\ 0 & 1 & 1 & 0 & & 1 & 1 \\ 0 & 0 & 0 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1}^\top & \mathbf{2}^\top & \mathbf{3}^\top & \cdots & (\mathbf{2^k - 1})^\top \end{bmatrix}$$

Simplex code has minimal distance 2^{k-1}, so "optimal" is $t = 2^{k-1}$.

Example

$$\boldsymbol{G}_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1}^\top & \mathbf{2}^\top & \mathbf{3}^\top \end{bmatrix}$$

$$\boldsymbol{a} = (a_1, a_2) \longrightarrow \boldsymbol{c} = (c_1, c_2, c_3) = (a_1, a_2, a_1 + a_2).$$

2-PIR:

$\boldsymbol{e}_1, \boldsymbol{e}_1 \longleftarrow \{1\}, \{2, 3\}$

2-batch:

$\boldsymbol{e}_1, \boldsymbol{e}_2 \longleftarrow \{1\}, \{2\}$

2-functional-batch:

$\boldsymbol{e}_1, \mathbf{3} \longleftarrow \{1\}, \{3\}$

$\mathbf{3}, \mathbf{3} \longleftarrow \{1, 2\}, \{3\}$

Greedy does not always work

$$\boldsymbol{G}_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

$t = 2^{k-1} = 4$.

Request: $\boldsymbol{r}_1 = \mathbf{1}, \boldsymbol{r}_2 = \mathbf{2}, \boldsymbol{r}_3 = \mathbf{1}, \boldsymbol{r}_4 = \mathbf{2}$.

Serve $\boldsymbol{r}_1 = \mathbf{1}$ and $\boldsymbol{r}_2 = \mathbf{2}$ by $\{1\}$ and $\{2\}$ (cheapest).

Then $\boldsymbol{r}_3 = \mathbf{1}$ by $\{4, 5\}$ or $\{6, 7\}$;

and $\boldsymbol{r}_4 = \mathbf{2}$ by $\{4, 6\}$ or $\{5, 7\}$.

Not possible!

Use service $\{1\}, \{4, 6\}, \{2, 3\}, \{5, 7\}$.

Known results, conjectures, new result

- $\boldsymbol{G}_k$ is 2^{k-1}-batch [Wang, Kiah, Cassuto (2015)]
  computer proof for $k \leq 8$, then induction algorithm
  better proof wanted!
- $\boldsymbol{G}_k$ is t-functional-batch for $t = \lfloor (5/6)2^{k-1} \rfloor - k$
  [Yohananov, Yaakobi (unpublished)], rather involved proof

Conjecture

$\boldsymbol{G}_k$ is 2^{k-1}-functional-batch. (Probably very hard!)

Our new result:

$\boldsymbol{G}_k$ is 2^{k-1}-odd-batch (with algorithmic proof).

Strongest conjecture: $\boldsymbol{G}_k$ can serve every request of 2^{k-1} vectors with disjoint column sets of size at most two.

Note: Requests for $\mathbf{0}$ are for free (take $I = \emptyset$).

Our result: $\boldsymbol{G}_k$ can serve every request of 2^{k-1} vectors of odd weight with disjoint column sets of size at most two.

Reformulations

Observation:

$$
\begin{array}{ccccccc}
 & & & \text{in } \mathbb{F}_2^k & & & \\
 & & \text{(odd weight)} & & \text{(odd weight)} & & \text{(even weight)} \\
(v_1, v_2, \ldots, v_{k-1}, \boxed{v_k}) & & \boldsymbol{r} & = & \boldsymbol{x} & + & \boldsymbol{y} \\
\downarrow \uparrow & & \updownarrow & & \updownarrow & & \updownarrow \\
(v_1, v_2, \ldots, v_{k-1}) & & \boldsymbol{r'} & = & \boldsymbol{x'} & + & \boldsymbol{y'} \\
 & & & \text{in } \mathbb{F}_2^{k-1} & & &
\end{array}
$$

Reformulation: Given arbitrary vectors
$\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{2^{k-1}}$ in $\mathbb{F}_2^{k-1}$,
there are mutually distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{2^{k-1}}$ in $\mathbb{F}_2^{k-1}$
(a permutation of $\mathbb{F}_{2^{k-1}}$)
such that the non-zero vectors among
$\boldsymbol{y}_1 = \boldsymbol{x}_1 + \boldsymbol{r}_1, \ldots, \boldsymbol{y}_{2^{k-1}} = \boldsymbol{x}_{2^{k-1}} + \boldsymbol{r}_{2^{k-1}}$
are also mutually distinct in $\mathbb{F}_2^{k-1}$.

Approach: Consider this as a problem in abelian groups.

A problem in abelian groups

We will show (with an **extension** algorithm):

Let $(G, +)$ be a finite abelian group, of order $n = |G|$. Given a sequence $r_1, \ldots, r_{n-1}$ in G, we can find distinct $x_1, \ldots, x_{n-1}$ in G such that $y_1 = x_1 + r_1, \ldots, y_{n-1} = x_{n-1} + y_{n-1}$ are also distinct.

Extension algorithm $m \longleftarrow m + 1$ (add r_0)

At the start:

$$\begin{array}{lccccc} Y: & y & \cdots & y' & & y'' \\ & \uparrow r_1 & & \uparrow r_{m-1} & & \uparrow r_m \\ X: & x & \cdots & x' & & x'' \end{array}$$

A single service: a **triple** (x, y, r) with $x + r = y$:
pictures as an arc from x to y labeled r.

The extension algorithm

Initially: $t = 0$.

Set X (size $m \leq |G| - 2$) matched by $r_1, \ldots, r_m$ into set Y.

New: r_0. Take distinct y_{-1}, y_0 outside Y, and set $x_0 = y_0 - r_0$.

No triples ($t = 0$) and $c := x_0 + y_{-1}$. To construct: $x = x_1$.

After step t (where $t \geq 0$):

t old triples (after renumbering r_i's): $(x_1, y_1, r_1), \ldots, (x_t, y_t, r_t)$;

t new triples: $y_{-1} = r_0 + x_1, \ldots, y_{t-2} = r_{t-1} + x_t$;

Constant: $c = x_0 + y_{-1} = \cdots = x_t + y_{t-1}$ on diagonals.

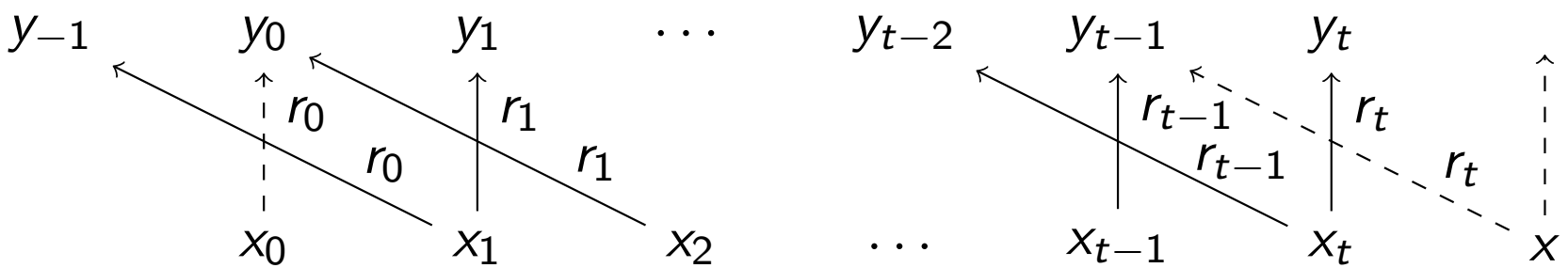


Figure 1: The extension algorithm $[x := y_{t-1} - r_t]$

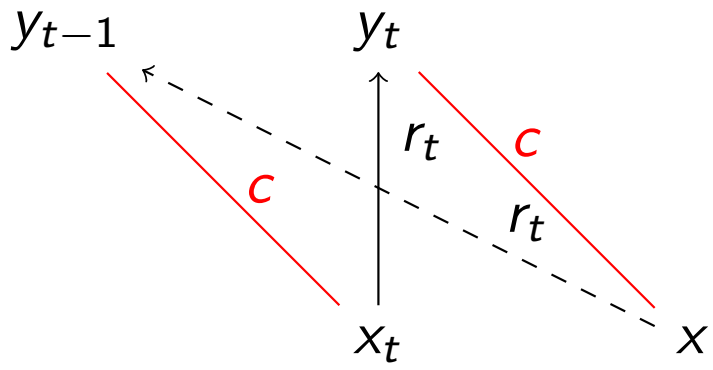


Figure 2: Crossing arcs

$$\begin{aligned} x_t + r_t &= y_t \\ x + r_t &= y_{t-1} \end{aligned} \qquad \text{hence} \qquad c = x_t + y_{t-1} = x + y_t.$$

So the "diagonal" relations keep on holding!

Cases

1. $x \notin X$. Then the t new triples

$$(x_1, y_{-1}, r_0), \ldots, (x_t, y_{t-2}, r_{t-1}),$$

the triple

$$(x, y_{t-1}, r_t),$$

and the $(m - t)$ old triples

$$(x_{t+1}, y_{t+1}, r_{t+1}), \ldots, (x_m, y_m, r_m)$$

together are a service for $r_0, r_1, \ldots, r_m$ and we are done.

2. $x \in X \setminus \{x_1, \ldots, x_t\}$. After renumbering: $x = x_{t+1}$, new triple $(x_{t+1}, y_{t+1}, r_{t+1})$, and the "diagonal" relation $c = x_{t+1} + y_t$.

3. $x \in \{x_1, \ldots, x_t\}$. This cannot occur!
   Proof: If $x = x_j$, $1 \leq j \leq t$, then $x_j + y_{j-1} = c = x_j + y_t$ (diagonals), hence $y_t \in \{y_0, y_1, \ldots, y_{t-1}\}$, contradiction.

Algorithm (continued)

Since $t \leq m$ always holds, the algorithm must end in case 1.

Application to our problem

- Take $(G, +) = (\mathbb{F}_2^m, +)$ where $\boxed{m = k - 1}$.
- Given $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{2^m}$.
- Use algorithm to construct
  distinct $\mathbf{x}_1, \ldots, \mathbf{x}_{2^m-1}$ in $\mathbb{F}_2^m$ such that
  $\boldsymbol{y}_1 = \boldsymbol{x}_1 + \boldsymbol{r}_1, \ldots, \boldsymbol{y}_{2^m-1} = \boldsymbol{x}_{2^m-1} + \boldsymbol{r}_{2^m-1}$ are **also distinct**.
- Let $\boldsymbol{x}_{2^m}$ be such that $\mathbb{F}_2^m = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{2^m}\}$ (the *missing* one).
- For $i = 1, \ldots, 2^m$, replace $\boldsymbol{x}_i, \boldsymbol{y}_i$ by $\boldsymbol{x}_i + \boldsymbol{a}, \boldsymbol{y}_i + \boldsymbol{a}$;
  take $\boldsymbol{a}$ such that $\boldsymbol{x}_{2^m} = \boldsymbol{r}_{2^m}$, set $\boldsymbol{y}_{2^m} = \mathbf{0}$, and add the triple $(\boldsymbol{r}_{2^m}, \mathbf{0}, \boldsymbol{r}_{2^m})$
  to complete the service for $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{2^m}$.

Conclusions

- We related batch-code properties of the simplex code to problems in finite additive groups.
- Simple computer-free proof-by-algorithm that the simplex code $\boldsymbol{G}_k$ is (optimal) 2^{k-1}-[odd]-batch, with column sets of size ≤ 2.
- The conjecture that $\boldsymbol{G}_k$ is even (optimal) 2^{k-1}-functional-batch is very interesting, and needs everyone's attention!

See arXiv: "On some batch code properties of the simplex code"
Conjectures on abelian groups; application of Alon's Nullstellensatz