On D-ary Fano Codes

- Ferdinando Cicalese, U. Verona
- Eros Rossi, U. Verona
Prefix code construction

\( X \) source or random variable
\( X = \{x_1, ..., x_n\} \) alphabet of \( X \)
\( X \sim p = (p_1, ..., p_n) \)

\( D \)-ary prefix code for \( X \)

\( c : x \in X \mapsto c(x) \in \{0, ..., D - 1\}^* \)

\( s.t \) there is a \( D \)-ary codetree \( T_c \) with \( n \) leaves

- edges out of same vertex are distinctly labelled with code symbols \( \{0, ..., D-1\} \)
- each \( x \) in \( X \) is distinctly mapped to a leaf \( \ell(x) \)
- \( c(x) \) is the sequence spelled on the path to \( \ell(x) \)
Fano code (D-ary)

Construction of the Prefix code as a codetree $T$

1. Sort $\mathbf{p} : p_1 \geq p_2 \geq ... \geq p_n$
2. Split the prob. vector $\mathbf{p}$ into $\min\{D,n\}$ groups as evenly as possible
3. Recursively apply 2. to build a Fano codetree for each group until groups are sigletons.
4. Label each edge with a codeletter and encode leaves with the labels on the root-to-leaf path

Step 2 (Fano Split)
Determine the goodness of the code
Shannon’s bound and redundancy

• Given \( p = (p_1, ..., p_n) \) and a codetree \( T \) for \( p \)

Average Code Length of \( T \):

\[
L(p) = \sum_{j=1}^{n} h_j p_j
\]

\( h_i \) depth of leaf for \( p_i \)

D-ary Entropy of \( p \):

\[
H_D(p) = \sum_{j=1}^{n} p_j \log_D \frac{1}{p_j}
\]

Shannon classical result

\[
H_D(p) \leq L(p) < H_D(p) + 1
\]
Shannon’s bound and redundancy

- **Given** $p = (p_1, \ldots, p_n)$ and a codetree $T$ for $p$

  **Average Code Length of** $T$:

  $$L(p) = \sum_{j=1}^{n} h_j p_j$$

  $h_i$ depth of leaf for $p_i$

  **D-ary Entropy of** $p$:

  $$H_D(p) = \sum_{j=1}^{n} p_j \log_D \frac{1}{p_j}$$

  **Redundancy of an optimal code**

  $$L(p) - H_D(p) < 1$$
D-ary Fano code in [Krajči et al. ISIT’15]

Construction of the Prefix code as a codetree $T$

1. Sort $p : p_1 \geq p_2 \geq ... \geq p_n$

2. Split the prob. vector $p$ into $\min\{D,n\}$ groups as evenly as possible

Conjecture [Krajči et al, ISIT’15]
For each $D$, a D-ary Fano code (built on Fano split in (*) ) achieves
$L(p) - H_D(p) \leq 1 - p_n$

Proved only for $D = 2,3$ [Krajči et al, ISIT’15]

Step 2 (Fano Split) in Krajči et al [ISIT’15]

minimize $\sum_i \sum_j |q_i - q_j|$  

$q_i$ is the sum of the $i$-th group
Our New D-ary Fano code

Construction of the Prefix code as a codetree $T$

1. Sort $\mathbf{p} : p_1 \geq p_2 \geq ... \geq p_n$

2. Split the prob. vector $\mathbf{p}$ into $\min\{D, n\}$ groups as evenly as possible

Conjecture [Krajči et al, ISIT’15]
For each $D$, a $D$-ary Fano code (built on a Fano split in (*) ) achieves
$$L(\mathbf{p}) - H_D(\mathbf{p}) \leq 1 - p_n$$

Our two Goals:
1. Improve efficiency of the split
2. Prove the conjecture (for a new Fano split)

Step 2 (Fano Split) in Krajči et al [ISIT’15]

minimize $\sum_i \sum_j |q_i - q_j|$ (*)&

$q_i$ is the sum of the $i$-th group
Our New D-ary Fano code

**Conjecture** [Krajči et al, ISIT’15]
For each D, a D-ary Fano code (built on a Fano split in (*) ) achieves
\[ L(p) - H_D(p) \leq 1 - p_n \]

**Our two Goals:**
1. Improve efficiency of the split
2. Prove the conjecture (for a new Fano split)

**Step 2** *(Our new Fano Split)*

\[ \text{maximize } \min_{i=1,...,D} q_i \]

\[ \Theta(nD) \]

minimize \[ \sum_{i} \sum_{j} |q_i - q_j| \] (*

\( q_i \) is the sum of the \( i \)-th group

Holds for every D
Our Results

In the proceedings article

- A new splitting criterion (Fano Split) based on max-min aggregations
  - computable in $O(nD)$
  - resulting D-ary code tree has redundancy $\leq 1-p_n$
    - for $D = 2, 3, 4$ (without other restrictions)
    - for all $D > 1$, for full trees (nodes have 0 or D children)

New result

- There exists a Fano codetree based on max-min aggregation that has redundancy $\leq 1-p_n$
  - Unconditionally for all $D > 1$
  - Proof based on a conjecture in the proceedings
### Max-min aggregations

Given a probability distribution \( p = p_1, ..., p_n \)

A **D-aggregation** \( q \) of \( p \) is the distribution obtained by aggregating contiguously components of \( p \) into \( D \) blocks.

We call \( q_{j,\text{First}} \) and \( q_{j,\text{Last}} \) the first and last component in the \( j \)th block.

A D-aggregation \( q \) is **max-min** if it maximizes the minimum component w.r.t to all D aggregation of \( p \).

\( q \) is **balanced** if, letting \( j^* = \arg\min_j q_j \)

- For \( j < j^* \) \( q_j - q_{j,\text{Last}} \leq q_{j^*} \)
- For \( j > j^* \) \( q_j - q_{j,\text{First}} \leq q_{j^*} \)

<table>
<thead>
<tr>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( p_5 )</th>
<th>( p_6 )</th>
<th>( p_7 )</th>
<th>( p_8 )</th>
<th>( p_9 )</th>
<th>( p_{10} )</th>
<th>( p_{11} )</th>
<th>( p_{12} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 ) = ( p_1 + p_2 )</td>
<td>( q_2 ) = ( p_3 + p_4 + p_5 )</td>
<td>( q_3 ) = ( p_6 + p_7 + p_8 )</td>
<td>( q_4 ) = ( p_9 + p_{10} + p_{11} + p_{12} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( q_{2,\text{First}} = p_3 \) \( q_{2,\text{Last}} = p_5 \)
Max-min aggregations

Given a probability distribution \( p = p_1, ..., p_n \)

A **D-aggregation** \( q \) of \( p \) is the distribution obtained by aggregating contiguously components of \( p \) into \( D \) blocks.

We call \( q_{j,First} \) and \( q_{j,Last} \) the first and last component in the \( j \)th block.

A D-aggregation \( q \) is **max-min** if it maximizes the minimum component w.r.t to all D aggregation of \( p \)

\( q \) is **balanced** if, letting \( j^* = \arg\min_j q_j \)

- For \( j < j^* \) \( q_j - q_{j,Last} \leq q_{j^*} \)
- For \( j > j^* \) \( q_j - q_{j,First} \leq q_{j^*} \)

**Properties**

1. A balanced D-aggregation can be computed in \( O(nD) \) [dynamic programming]
   - \( O(nD \log n) \) for a whole balanced codetrees

2. \( 1 - H_D(q) \leq \sum_{j=1}^{D-1} q_{j,Last} \)
Our New D-ary Fano code

1. Sort \( p : p_1 \geq p_2 \geq \ldots \geq p_n \)
2. Split the prob. vector \( p \) into \( \min\{D,n\} \) groups as evenly as possible using a balanced max-min \( D \)-aggregation
3. Recursively apply 2. to build a Fano codetree for each group

**Theorem:**
If in the codetree every node has 0 or \( D \) children, then \( L(p) - H_D(p) \leq 1 - p_n \)

\[
L(p) - H_D(p) = \sum_{\text{node } v} Q_v \left( 1 - H_D(q[v]) \right)
\]

\[
\sum_{\text{node } v} \sum_{i=1}^{D-1} q_i^{[v]} \leq 1 - p_n
\]

when all non-leaf \( v \) have \( D \) children or \( D \leq 4 \)
Our New D-ary Fano code

1. Sort \( p : p_1 \geq p_2 \geq \ldots \geq p_n \)
2. Split the prob. vector \( p \) into \( \min\{D,n\} \) groups as evenly as possible using a balanced max-min D-aggregation
3. Recursively apply 2. to build a Fano codetree for each group

Conjecture (in the proceedings):
The contribution to \( H_D(p) \) and \( L(p) \) of \( w \) and its non-full children (like \( v \)) is bounded by

\[
H_D(q[w]) + \sum_{v \text{ child not full}} Q_v H_D(q[w]) \geq \sum_{\text{ith child not full}} q_i[w] + \sum_{\text{ith child is full}} (q_i[w] - q_{i,\text{Last}}) + q_{D,\text{Last}}
\]

\[
Q_w = \sum q_i[w]
\]
Open Problems

• Can we construct balanced max-min aggregations faster
  – greedy approach
  – $O(D \log n)$?
    • codetree construction in $O(nD \log n)$
• what about other Fano splits
  – min-max aggregations?
  – max-entropy aggregations?
Thank You for your attention