New possibilities of codings – information theory in practice

3 new families of codings:

1) **Asymmetric Numeral Systems** – widely used new entropy coding
   - **Huffman** coding: fast, but suboptimal (compression ratio)
   - **Arithmetic** coding: slow, nearly optimal (→ Shannon entropy)
   - **ANS**: fast and nearly optimal (build automaton)

Asymmetric knowledge of sender-receiver:

2) **Generalizations of Kuznetsov-Tsybakov problem**
   - e.g. for more subtle steganography/watermarking, like image-like QR codes

3) **Joint Reconstruction Codes** – enhancement of Fountain codes
   - When sender doesn’t know noise levels (receiver does), e.g. data storage

Jarosław Duda, Warszawa, 6.II.2016
1. **Entropy coding** – the heart of data compressors

(prefix,) **Huffman coding** (also unary, Golomb, Elias, etc.)

- **fast** (>300MB/s/core)
- no multiplication, needs sorting
- but **inaccurate**: \( \Pr(s) \sim 2^{-r} \)
- e.g. for \( \Pr(a)=0.01, \Pr(b)=0.99 \)
- uses **1 bit/symbol**

**Past: compromise**

**Arithmetic/range coding**

- **slow** (<< 100MB/s/core)
- uses multiplication
- uses nearly **accurate** \( \Pr(s) \)
- e.g. for \( \Pr(a)=0.01, \Pr(b)=0.99 \)
- uses \(~0.08\) bits/symbol

**Or?**

**Asymmetric Numeral Systems (ANS)**

- **fast** (>500MB/s/core)
- no multiplication, no sorting
- uses nearly **accurate** \( \Pr(s) \)
- e.g. for \( \Pr(a)=0.01, \Pr(b)=0.99 \)
- uses \(~0.08\) bits/symbol

**Now: ANS**

Also allows for simultaneous encryption

**Some ANS decoding:**

- \( X \rightarrow s, \text{ new } X \)
- \( 0 \rightarrow a, 2 + d_1 \)
- \( 1 \rightarrow b, 0 + 2d_2 + d_1 \)
- \( 2 \rightarrow a, 0 \)
- \( 3 \rightarrow a, 1 \)

\( newX, nbBits, decodingTable \)
**Huffman vs ANS in compressors** (LZ + entropy coder):

from Matt Mahoney benchmarks [http://mattmahoney.net/dc/text.html](http://mattmahoney.net/dc/text.html)

<table>
<thead>
<tr>
<th>Compressor</th>
<th>LZA 0.82b –mx9 –b7 –h7</th>
<th>lzturbo 1.2 –39 –b24</th>
<th>WinRAR 5.00 –ma5 –m5</th>
<th>WinRAR 5.00 –ma5 –m2</th>
<th>lzturbo 1.2 –32</th>
<th>zhuff 0.97 –c2</th>
<th>gzip 1.3.5 –9</th>
<th>pkzip 2.0.4 –ex</th>
<th>ZSTD 0.0.1</th>
<th>WinRAR 5.00 –ma5 –m1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enwiki8 100,000,000B</td>
<td>26,396,613</td>
<td>26,915,461</td>
<td>27,835,431</td>
<td>29,758,785</td>
<td>30,979,376</td>
<td>34,907,478</td>
<td>36,445,248</td>
<td>36,556,552</td>
<td>40,024,854</td>
<td>40,565,268</td>
</tr>
<tr>
<td>encode time [ns/byte]</td>
<td>449</td>
<td>582</td>
<td>1004</td>
<td>228</td>
<td>19</td>
<td>24</td>
<td>101</td>
<td>171</td>
<td>7.7</td>
<td>54</td>
</tr>
<tr>
<td>decode time [ns/byte]</td>
<td>9.7</td>
<td>2.8</td>
<td>31</td>
<td>30</td>
<td>2.7</td>
<td>3.5</td>
<td>17</td>
<td>50</td>
<td>3.8</td>
<td>31</td>
</tr>
</tbody>
</table>

**zhuff, ZSTD** (Yann Collet): LZ4 + tANS (switched from Huffman)

**lzturbo** (Hamid Bouzidi): LZ + tANS (switched from Huffman)

**LZA** (Nania Francesco): LZ + rANS (switched from range coding)

e.g. lzturbo vs gzip: **better compression, 5x faster encoding, 6x faster decoding**

**saving time and energy in extremely frequent task**
**Apple LZFSE** = Lempel-Ziv + Finite State Entropy
Default in iOS9 and OS X 10.11
“matching the compression ratio of ZLIB level 5, but with much higher energy efficiency and speed (between 2x and 3x) for both encode and decode operation”

**Finite State Entropy** is Yann Collet’s (known from e.g. LZ4) implementation of **tANS**

Default **DNA compression**: **CRAM 3.0** of European Bioinformatics Institute

<table>
<thead>
<tr>
<th>Format</th>
<th>Size</th>
<th>Encoding(s)</th>
<th>Decoding(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM</td>
<td>5 579 036 306</td>
<td>46</td>
<td>-</td>
</tr>
<tr>
<td>CRAM v2 (LZ77+Huff)</td>
<td>1 053 744 556</td>
<td>183</td>
<td>27</td>
</tr>
<tr>
<td><strong>CRAM v3 (order 1 rANS)</strong></td>
<td>869 500 447</td>
<td>75</td>
<td>31</td>
</tr>
</tbody>
</table>
2. Generalizations of Kuznetsov-Tsybakov problem (KT)

Imagine we can send \( n \) bits, but \( k = p_f n \) of them are damaged (fixed)

For example QR-like codes with fixed some fraction of pixels (e.g. as contour of a plane):

\[
\begin{align*}
160 \times 160 & \quad p_f = 0.139 \\
80 \times 80 & \quad p_f = 0.157 \\
40 \times 40 & \quad p_f = 0.189
\end{align*}
\]

If the **receiver would know positions** of these strong constraints, we could just use the remaining \( n - k = (1 - p_f)n \) bits

**Kuznetsov-Tsybakov problem**: what if **only the sender knows the constraints**?

Surprisingly, we can still **approach** the same capacity.

But at a cost of searching for satisfying a code (paid only by the sender!)
KT: strong constraints – enforce values, **weak – enforce densities**

**Homogeneous contrast case:** use \((c, 1 - c)\) or \((1 - c, c)\) probability dist.

<table>
<thead>
<tr>
<th>Rate: 7/8</th>
<th>Rate: 3/4</th>
<th>Rate: 1/2</th>
<th>Rate: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contrast:</strong> 0.7051</td>
<td><strong>Contrast:</strong> 0.7855</td>
<td><strong>Contrast:</strong> 0.8900</td>
<td><strong>Contrast:</strong> 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>40 x 40 (1600b)</th>
<th>160 x 160 (25600b)</th>
<th>80 x 80 (6400b)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
</tbody>
</table>
General constraints: grayness of a pixel in picture is probability of using ‘1’ in code
**Constrained Coding**: find the closest $Y = T(payload\_bits, freedom\_bits)$, send $Y$

**Rate Distortion**: find the closest $T(0, freedom\_bits)$, send $freedom\_bits$

Because $T$ is bijection, for the same constraints/message to fulfill/resemble,

\[
density\ of\ freedom\ bits + density\ of\ payload\ bits = 1
\]

rate of RD + rate of CC = 1

we can translate previous calculations by just using $1 - \text{rate}$

B&W picture: 1 bit per pixel = visual aspect (RD) + hidden message (CC)

<table>
<thead>
<tr>
<th>K-T, $p_f = 0.157$</th>
<th>HC, contrast = 0.7855</th>
<th>general, average $h(g) \sim 0.82$</th>
</tr>
</thead>
</table>

**Constrained Coding** - we could **hide within** such halftone picture **at most**:

1 - 0.157 = 0.843 bits/pixel \quad h(0.7855) \sim 0.75 bits/pixel \quad 0.82 bits/pixel

**Rate Distortion** - we could **store/compress** such picture alone using **at least**:

0.157 bits/pixel \quad 0.25 bits/pixel \quad 0.18 bits/pixel
3. Joint Reconstruction Codes (JRC)

**Fountain codes**: multiple universal packets, any large enough subset of undamaged ones is sufficient … what about *noise*: damaged packets?

**JRC**: the *sender doesn’t need to know (final) noise levels* (receiver does)

The decoder adapts to the actual final noise levels as individual trust levels.
Some packets are **lost**, some are **damaged**

Adding redundancy (sender) requires **knowing the damage level**

**Not available** in many situations (JRC...):

- The final damage of **storage medium** depends on its history,
- **broadcaster** doesn’t know individual damage levels,
- damage of **watermarking** depends on capturing conditions
- damage of packet in **network** depends on its route
- rapidly **varying conditions** can prevent adaptation
JRC, sequential decoding: **undamaged case** for received $N + 1$ packets

Large $N$: **Gaussian distribution**, on average 1.5 more nodes to test

**Damaged case** ($N = 3, 2: \epsilon = 0, d: \epsilon = 0.2$): \approx **Pareto distribution**
Sequential decoding: choose and expand most promising candidate (largest weight)

Rate is

\[ R_c(\varepsilon) = 1 - h_{1/(1+c)}(\varepsilon) \]

for Pareto coefficient \( c \)

where \( h_u(\varepsilon) = \frac{\varepsilon^u + (1-\varepsilon)^u}{1-u} \)

is Renyi entropy, \( h_1 \) is Shannon entropy

\[ \Pr(\# \text{ steps} > s) \propto s^{-c} \]

\[ R_0(\varepsilon) = 1 - h(\varepsilon) \]

Shannon bound

\[ R_1(\varepsilon) = \lg \left( 1 + 2\sqrt{\varepsilon(1-\varepsilon)} \right) \]

“cut-off” bound

\[ R_0(\varepsilon) = 1 - h(\varepsilon) \]

Shannon bound

\[ R_1(\varepsilon) = \lg \left( 1 + 2\sqrt{\varepsilon(1-\varepsilon)} \right) \]

“cut-off” bound
1. **Asymmetric Numeral Systems** – currently replaces Huffman and AC
   Articles, implementations, compressors, benchmarks:

2. **Generalizations of Kuznetsov-Tsybakov problem**
   Image-like QR codes, more subtle steganography/watermarking
   Also: telecommunication settings with dynamically adapting sender

3. **Joint Reconstruction Codes** enhancement of Fountain Codes
   When encoder doesn’t know the final noise levels,
   Like in data storage, broadcasting, networking, watermarking
   Also: data compression with encoder not knowing prior information of decoder