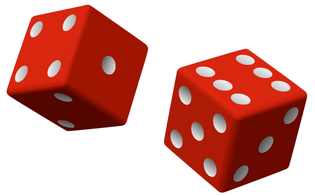
Sampling codes with memory and the insertion list-decoding capacity



João Ribeiro

Inst. Telecomunicações + Técnico-ULisboa

Based on joint work with

Roni Con

Technion → Tel Aviv U

Dean Doron

Ben-Gurion U

Funded by the European Union

European Research Council

Established by the European Commission

Insertions and deletions

Simple models of errors that cause loss of synchronization.

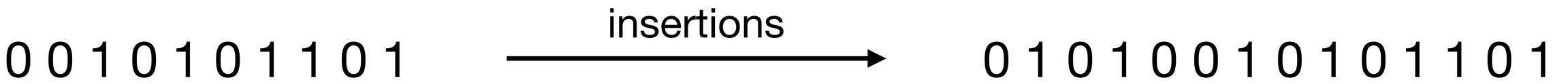
Insertions and deletions

Simple models of errors that cause loss of synchronization.

0 0 1 0 1 0 1 1 0 1

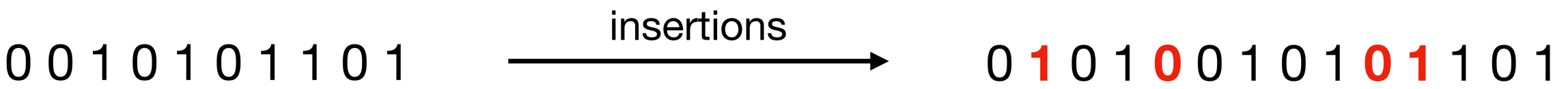
Insertions and deletions

Simple models of errors that cause loss of synchronization.



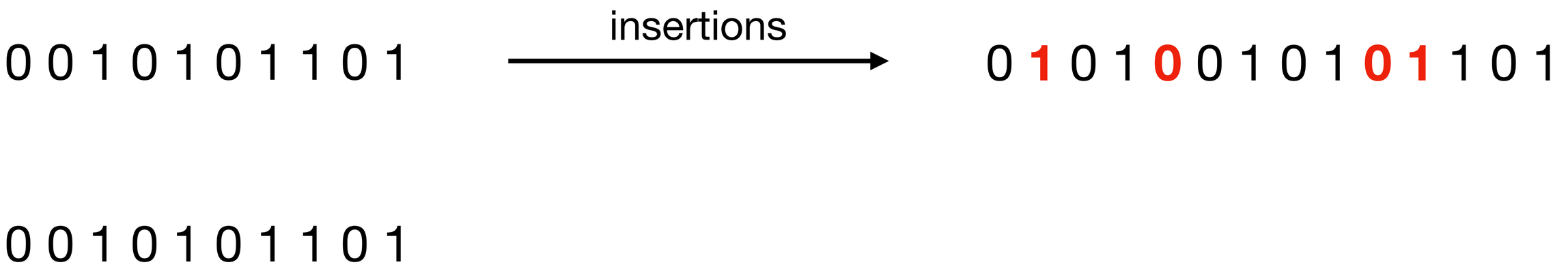
Insertions and deletions

Simple models of errors that cause loss of synchronization.



Insertions and deletions

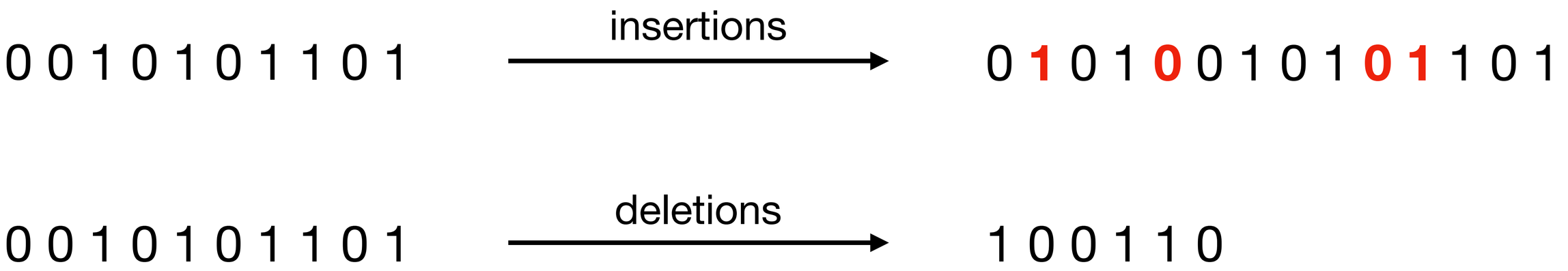
Simple models of errors that cause loss of synchronization.



0 0 1 0 1 0 1 1 0 1

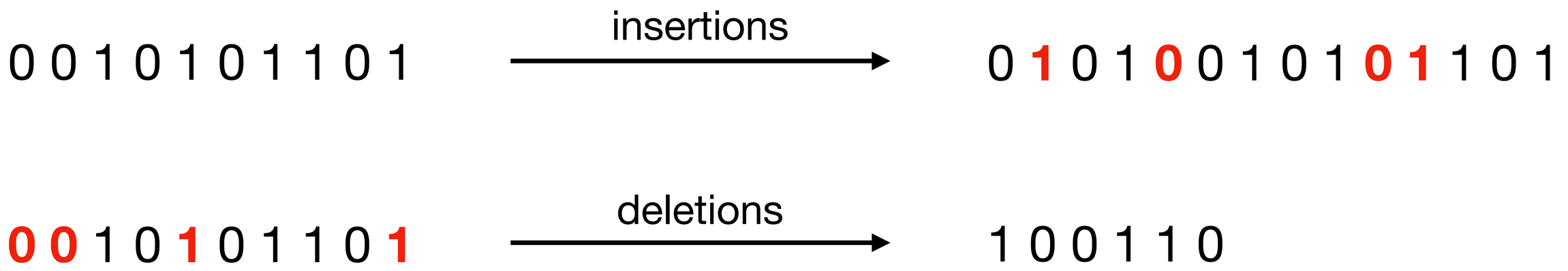
Insertions and deletions

Simple models of errors that cause loss of synchronization.



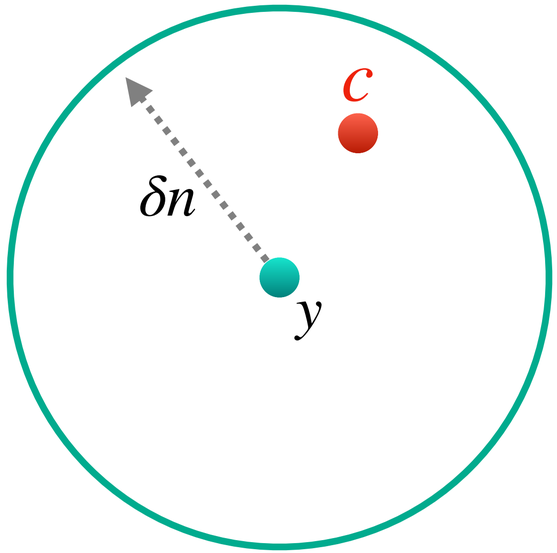
Insertions and deletions

Simple models of errors that cause loss of synchronization.



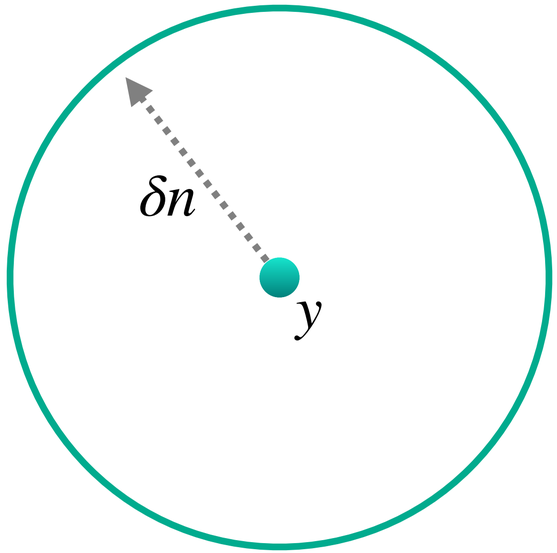
List-decoding

A useful relaxation of unique decoding.



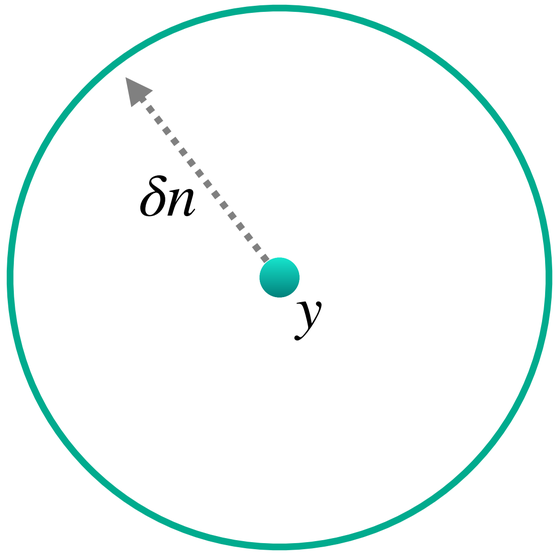
List-decoding

A useful relaxation of unique decoding.



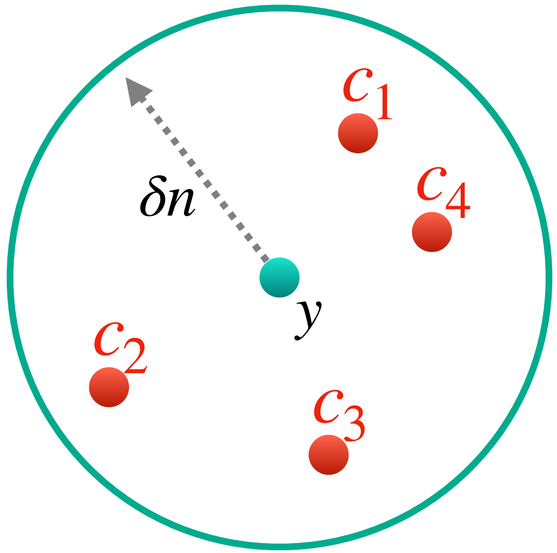
List-decoding

A useful relaxation of unique decoding.



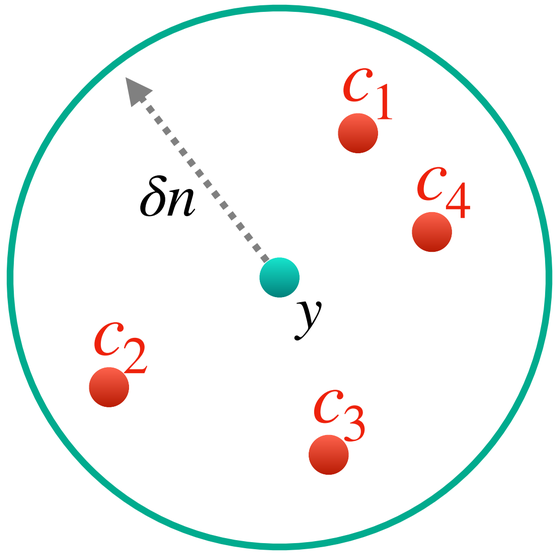
List-decoding

A useful relaxation of unique decoding.



List-decoding

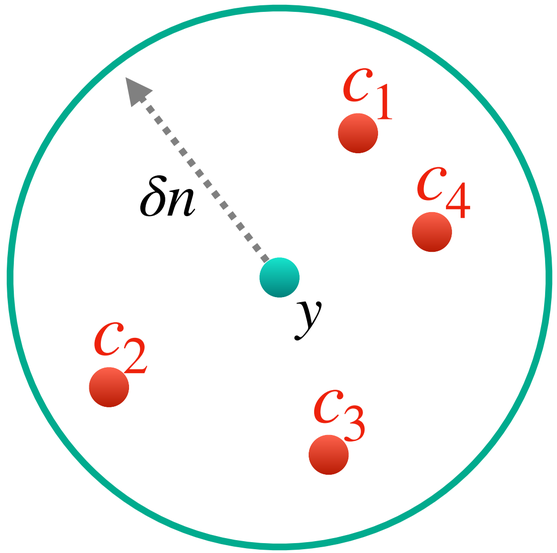
A useful relaxation of unique decoding.



Why care about list-decoding?

List-decoding

A useful relaxation of unique decoding.

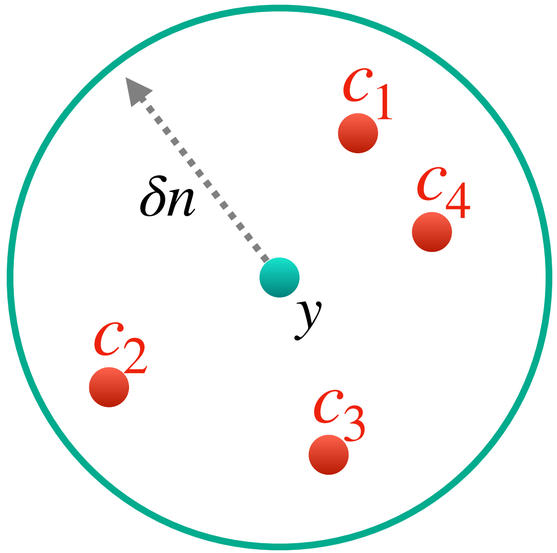


Why care about list-decoding?

- Natural geometric notion;

List-decoding

A useful relaxation of unique decoding.

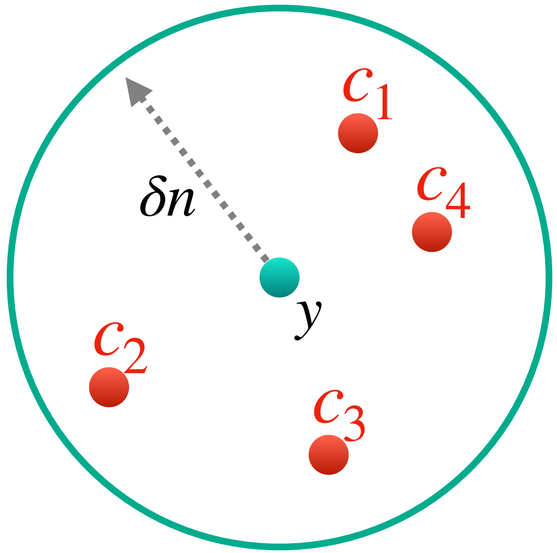


Why care about list-decoding?

- Natural geometric notion;
- Can achieve much higher rate with small list size;

List-decoding

A useful relaxation of unique decoding.

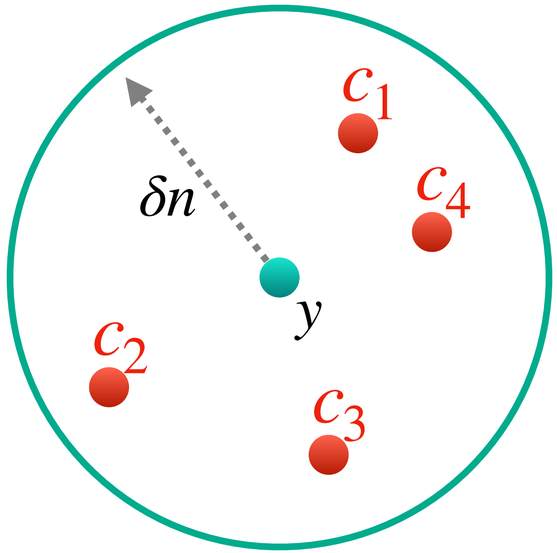


Why care about list-decoding?

- Natural geometric notion;
- Can achieve much higher rate with small list size;
- Small lists may be OK in practice (pruned with contextual information);

List-decoding

A useful relaxation of unique decoding.

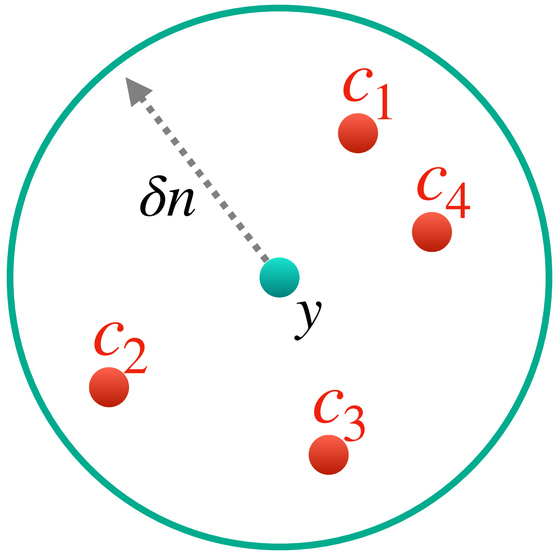


Why care about list-decoding?

- Natural geometric notion;
- Can achieve much higher rate with small list size;
- Small lists may be OK in practice (pruned with contextual information);
- Stepping stone towards unique decoding;

List-decoding

A useful relaxation of unique decoding.

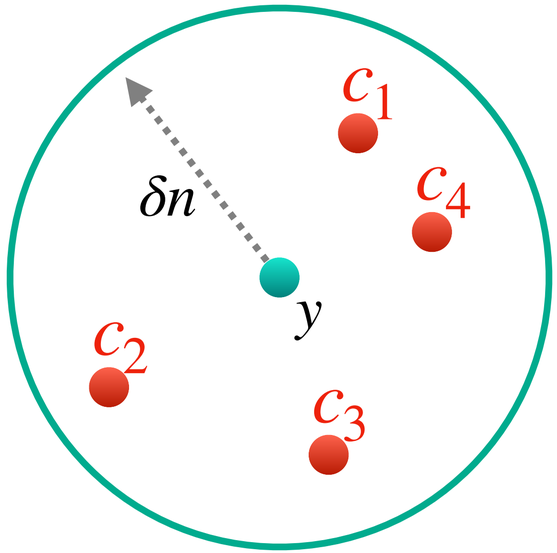


Why care about list-decoding?

- Natural geometric notion;
- Can achieve much higher rate with small list size;
- Small lists may be OK in practice (pruned with contextual information);
- Stepping stone towards unique decoding;
- Curious connections to channel capacity!

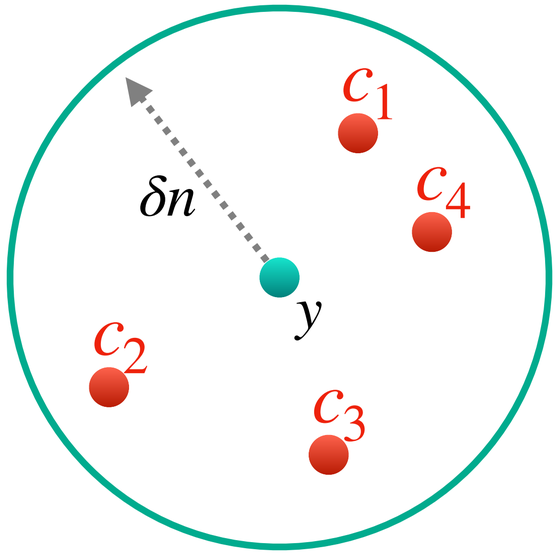
List-decoding

A useful relaxation of unique decoding.



List-decoding

A useful relaxation of unique decoding.

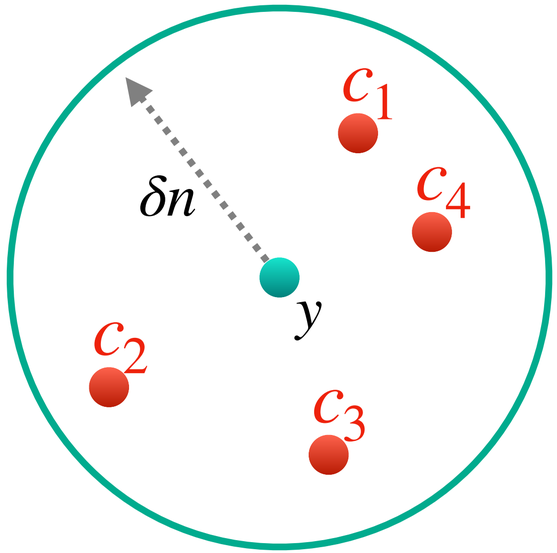


$C \subseteq \{0,1\}^n$ is (δ, L)-**list-decodable from insertions** if for all $y \in \{0,1\}^{(1+\delta)n}$

$$|\{c \in C : y \text{ is supersequence of } c\}| \leq L$$

List-decoding

A useful relaxation of unique decoding.



$C \subseteq \{0,1\}^n$ is (δ, L)-**list-decodable from insertions** if for all $y \in \{0,1\}^{(1+\delta)n}$

$$|\{c \in C : y \text{ is supersequence of } c\}| \leq L$$

$C \subseteq \{0,1\}^n$ is (δ, L)-**list-decodable from deletions** if for all $y \in \{0,1\}^{(1-\delta)n}$

$$|\{c \in C : y \text{ is subsequence of } c\}| \leq L$$

List-decoding capacity

Supremum of rates at which we can list-decode from a δ-fraction of errors with “small” list size (constant or polynomial in the block length).

List-decoding capacity

Supremum of rates at which we can list-decode from a δ-fraction of errors with “small” list size (constant or polynomial in the block length).

Analogous question for unique decoding ($L = 1$) is a major open problem in coding theory, even for bit-flips!

Capacity of list-decoding from bit-flips

This is easy to work out!

Capacity of list-decoding from bit-flips

This is easy to work out!

The capacity of list-decoding from a δ-fraction of bit-flips is $1 - h(\delta)$ when $\delta < 1/2$.

Capacity of list-decoding from bit-flips

This is easy to work out!

The capacity of list-decoding from a δ-fraction of bit-flips is $1 - h(\delta)$ when $\delta < 1/2$.

This matches the capacity of the binary symmetric channel. (Not a coincidence!)

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

$$|\{(c, y) : c \in C, d(c, y) \leq \delta n\}|$$

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

$$L \cdot 2^n \geq \quad |\{(c, y) : c \in C, d(c, y) \leq \delta n\}|$$

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

$$L \cdot 2^n \geq \quad |\{(c, y) : c \in C, d(c, y) \leq \delta n\}| \quad = \sum_{c \in C} |B_\delta(c)|$$

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

$$L \cdot 2^n \geq \quad |\{(c, y) : c \in C, d(c, y) \leq \delta n\}| \; = \sum_{c \in C} |B_\delta(c)| \; \approx 2^{Rn} \cdot 2^{h(\delta)n}$$

Capacity of list-decoding from bit-flips

(Converse — exponential list size):
If C has rate $R = 1 - h(\delta) + \varepsilon$ and has list size L, then

$$L \cdot 2^n \geq \quad |\{(c, y) : c \in C, d(c, y) \leq \delta n\}| \; = \sum_{c \in C} |B_\delta(c)| \; \approx 2^{Rn} \cdot 2^{h(\delta)n}$$

$$\implies L \geq 2^{\varepsilon n}$$

Capacity of list-decoding from bit-flips

(Achievability — constant list size):
Take a uniformly random code C of rate $R = 1 - h(\delta) - \varepsilon$.

Capacity of list-decoding from bit-flips

(Achievability — constant list size):
Take a uniformly random code C of rate $R = 1 - h(\delta) - \varepsilon$.

The probability that $L + 1$ codewords land in the radius-δn Hamming ball around y is $\leq (2^{h(\delta)n}/2^n)^{L+1}$.

Capacity of list-decoding from bit-flips

(Achievability — constant list size):
Take a uniformly random code C of rate $R = 1 - h(\delta) - \varepsilon$.

The probability that $L+1$ codewords land in the radius-δn Hamming ball around y is $\leq (2^{h(\delta)n}/2^n)^{L+1}$.

By a union bound, the probability that C is **not** (δ, L)-list-decodable is at most

$$\binom{2^{Rn}}{L+1} \cdot 2^n \cdot 2^{-(1-h(\delta))n(L+1)}$$

Capacity of list-decoding from bit-flips

(Achievability — constant list size):
Take a uniformly random code C of rate $R = 1 - h(\delta) - \varepsilon$.

The probability that $L+1$ codewords land in the radius-δn Hamming ball around y is $\leq (2^{h(\delta)n}/2^n)^{L+1}$.

By a union bound, the probability that C is **not** (δ, L)-list-decodable is at most

$$\binom{2^{Rn}}{L+1} \cdot 2^n \cdot 2^{-(1-h(\delta))n(L+1)} \leq 2^{n(L+1)\left(R-(1-h(\delta))+\frac{1}{L+1}\right)}.$$

Capacity of list-decoding from bit-flips

(Achievability — constant list size):
Take a uniformly random code C of rate $R = 1 - h(\delta) - \varepsilon$.

The probability that $L+1$ codewords land in the radius-δn Hamming ball around y is $\leq (2^{h(\delta)n}/2^n)^{L+1}$.

By a union bound, the probability that C is **not** (δ, L)-list-decodable is at most

$$\binom{2^{Rn}}{L+1} \cdot 2^n \cdot 2^{-(1-h(\delta))n(L+1)} \leq 2^{n(L+1)\left(R-(1-h(\delta))+\frac{1}{L+1}\right)}.$$

This is < 1 when $L \geq 1/\varepsilon$.

Capacity of list-decoding from bit-flips

Capacity of list-decoding from bit-flips

Summarizing:

Capacity of list-decoding from bit-flips

Summarizing:

- At rate $R = 1 - h(\delta) - \varepsilon$ we can achieve list size $1/\varepsilon$ with a uniformly random code;

Capacity of list-decoding from bit-flips

Summarizing:

- At rate $R = 1 - h(\delta) - \varepsilon$ we can achieve list size $1/\varepsilon$ with a uniformly random code;
- At rate $R = 1 - h(\delta) + \varepsilon$, every code has list size at least $2^{\varepsilon n}$.

Capacity of list-decoding from insertions — the story so far

Much less is known!

Capacity of list-decoding from insertions — the story so far

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1]$$ [Haeupler-Shahrasbi-Sudan '18]

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$|\{(c, y) : c \in C, y \text{ is supersequence of } c\}|$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$L \cdot 2^{(1+\delta)n} \geq |\{(c, y) : c \in C, y \text{ is supersequence of } c\}|$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$L \cdot 2^{(1+\delta)n} \geq |\{(c,y) : c \in C, y \text{ is supersequence of } c\}|$$

$$= \sum_{c \in C} |B_\delta^{\text{ins}}(c)|$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \leq (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$L \cdot 2^{(1+\delta)n} \geq |\{(c,y) : c \in C, y \text{ is supersequence of } c\}|$$
$$= \sum_{c \in C} |B_\delta^{\text{ins}}(c)| \approx 2^{Rn} \cdot 2^{(1+\delta)h\left(\frac{\delta}{1+\delta}\right)n}$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \le (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$L \cdot 2^{(1+\delta)n} \ge |\{(c,y) : c \in C, y \text{ is supersequence of } c\}|$$

$$= \sum_{c \in C} |B_\delta^{\text{ins}}(c)| \approx 2^{Rn} \cdot 2^{(1+\delta)h\left(\frac{\delta}{1+\delta}\right)n}$$

number of length-$(1+\delta)n$ supersequences of any length-n string

$$= \sum_{i=0}^{\delta n} \binom{(1+\delta)n}{i} \approx 2^{(1+\delta)h\left(\frac{\delta}{1+\delta}\right)n}$$

Capacity of list-decoding from insertions — the story so far

Upper bound mimicking bit-flips approach:

$$C_{\text{ins}}(\delta) \le (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \text{ for } \delta \in [0,1] \qquad \text{[Haeupler-Shahrasbi-Sudan '18]}$$

If C has rate $R = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) + \varepsilon$ and has list size L, then

$$L \cdot 2^{(1+\delta)n} \ge |\{(c,y) : c \in C, y \text{ is supersequence of } c\}|$$

$$= \sum_{c \in C} |B_\delta^{\text{ins}}(c)| \approx 2^{Rn} \cdot 2^{(1+\delta)h\left(\frac{\delta}{1+\delta}\right)n}$$

$$\implies L \ge 2^{\varepsilon n}$$

number of length-$(1+\delta)n$ supersequences of any length-n string

$$= \sum_{i=0}^{\delta n} \binom{(1+\delta)n}{i} \approx 2^{(1+\delta)h\left(\frac{\delta}{1+\delta}\right)n}$$

Capacity of list-decoding from insertions — the story so far

To get a lower bound… it’s natural to try a uniformly random code!

Capacity of list-decoding from insertions — the story so far

To get a lower bound… it's natural to try a uniformly random code!

Performance of uniformly random code is characterized by size of largest "deletion ball"

$$B_\delta^{\mathrm{del}}(y) = \{x \in \{0,1\}^n : x \text{ is subsequence of } y\}$$

Capacity of list-decoding from insertions — the story so far

To get a lower bound… it's natural to try a uniformly random code!

Performance of uniformly random code is characterized by size of largest "deletion ball"

$$B_\delta^{\mathrm{del}}(y) = \{x \in \{0,1\}^n : x \text{ is subsequence of } y\}$$

[Hirschberg-Régnier '02]: Every $y \in \{0,1\}^{(1+\delta)n}$ has at most $\sum_{i=0}^{\delta n} \binom{n}{i}$ length-n subsequences,

and equality is achieved when $y = 0101\ldots01$ or $y = 1010\ldots10$.

Capacity of list-decoding from insertions — the story so far

To get a lower bound… it's natural to try a uniformly random code!

Performance of uniformly random code is characterized by size of largest "deletion ball"

$$B_\delta^{\mathrm{del}}(y) = \{x \in \{0,1\}^n : x \text{ is subsequence of } y\}$$

[Hirschberg-Régnier '02]: Every $y \in \{0,1\}^{(1+\delta)n}$ has at most $\sum_{i=0}^{\delta n} \binom{n}{i}$ length-n subsequences,

and equality is achieved when $y = 0101\ldots01$ or $y = 1010\ldots10$.

$\Longrightarrow$ size of largest deletion ball is $\approx 2^{h(\delta)n}$ when $\delta < 1/2$ and $\geq 2^{n-1}$ when $\delta \geq 1/2$.

Capacity of list-decoding from insertions — the story so far

To get a lower bound… it's natural to try a uniformly random code!

Performance of uniformly random code is characterized by size of largest "deletion ball"

$$B_\delta^{\mathrm{del}}(y) = \{x \in \{0,1\}^n : x \text{ is subsequence of } y\}$$

[Hirschberg-Régnier '02]: Every $y \in \{0,1\}^{(1+\delta)n}$ has at most $\sum_{i=0}^{\delta n} \binom{n}{i}$ length-n subsequences,

and equality is achieved when $y = 0101\ldots01$ or $y = 1010\ldots10$.

$\Longrightarrow$ size of largest deletion ball is $\approx 2^{h(\delta)n}$ when $\delta < 1/2$ and $\geq 2^{n-1}$ when $\delta \geq 1/2$.

$$\Longrightarrow \quad C_{\mathrm{ins}}(\delta) \geq \begin{cases} 1 - h(\delta), & \text{if } \delta < 1/2, \\ 0, & \text{if } \delta \geq 1/2. \end{cases}$$

with constant list size [Haeupler-Shahrasbi-Sudan '18]*

Capacity of list-decoding from insertions — the story so far

Uniformly random codes don’t give any non-trivial lower bound when $\delta \geq 1/2$…

Capacity of list-decoding from insertions — the story so far

Uniformly random codes don't give any non-trivial lower bound when $\delta \geq 1/2$…

Not obvious whether $C_{\text{ins}}(\delta) > 0$ when $\delta \geq 1/2$.

Capacity of list-decoding from insertions — the story so far

Uniformly random codes don't give any non-trivial lower bound when $\delta \geq 1/2$…

Not obvious whether $C_{\text{ins}}(\delta) > 0$ when $\delta \geq 1/2$.

Johnson bound for insdels
+ Bukh-Guruswami-Håstad

[Wachter-Zeh '17, Hayashi-Yasunaga '18, Liu-Tjuawinata-Xing '23, Liang '26]

$\Longrightarrow$ $C_{\text{ins}}(\delta) > 0$ for all $\delta < 0.707$.

Capacity of list-decoding from insertions — the story so far

Uniformly random codes don't give any non-trivial lower bound when $\delta \geq 1/2$…

Not obvious whether $C_{\text{ins}}(\delta) > 0$ when $\delta \geq 1/2$.

Johnson bound for insdels
+ Bukh-Guruswami-Håstad
[Wachter-Zeh '17, Hayashi-Yasunaga '18, Liu-Tjuawinata-Xing '23, Liang '26]

$\Longrightarrow$ $C_{\text{ins}}(\delta) > 0$ for all $\delta < 0.707$.

intricate explicit construction
[Guruswami-Haeupler-Shahrasbi '20]

$\Longrightarrow$ $C_{\text{ins}}(\delta) > 0$ for all $\delta < 1$.

Capacity of list-decoding from insertions — the story so far

Uniformly random codes don't give any non-trivial lower bound when $\delta \geq 1/2$…

Not obvious whether $C_{\text{ins}}(\delta) > 0$ when $\delta \geq 1/2$.

Johnson bound for insdels
+ Bukh-Guruswami-Håstad
[Wachter-Zeh '17, Hayashi-Yasunaga '18, Liu-Tjuawinata-Xing '23, Liang '26]

$\Longrightarrow$ $C_{\text{ins}}(\delta) > 0$ for all $\delta < 0.707$.

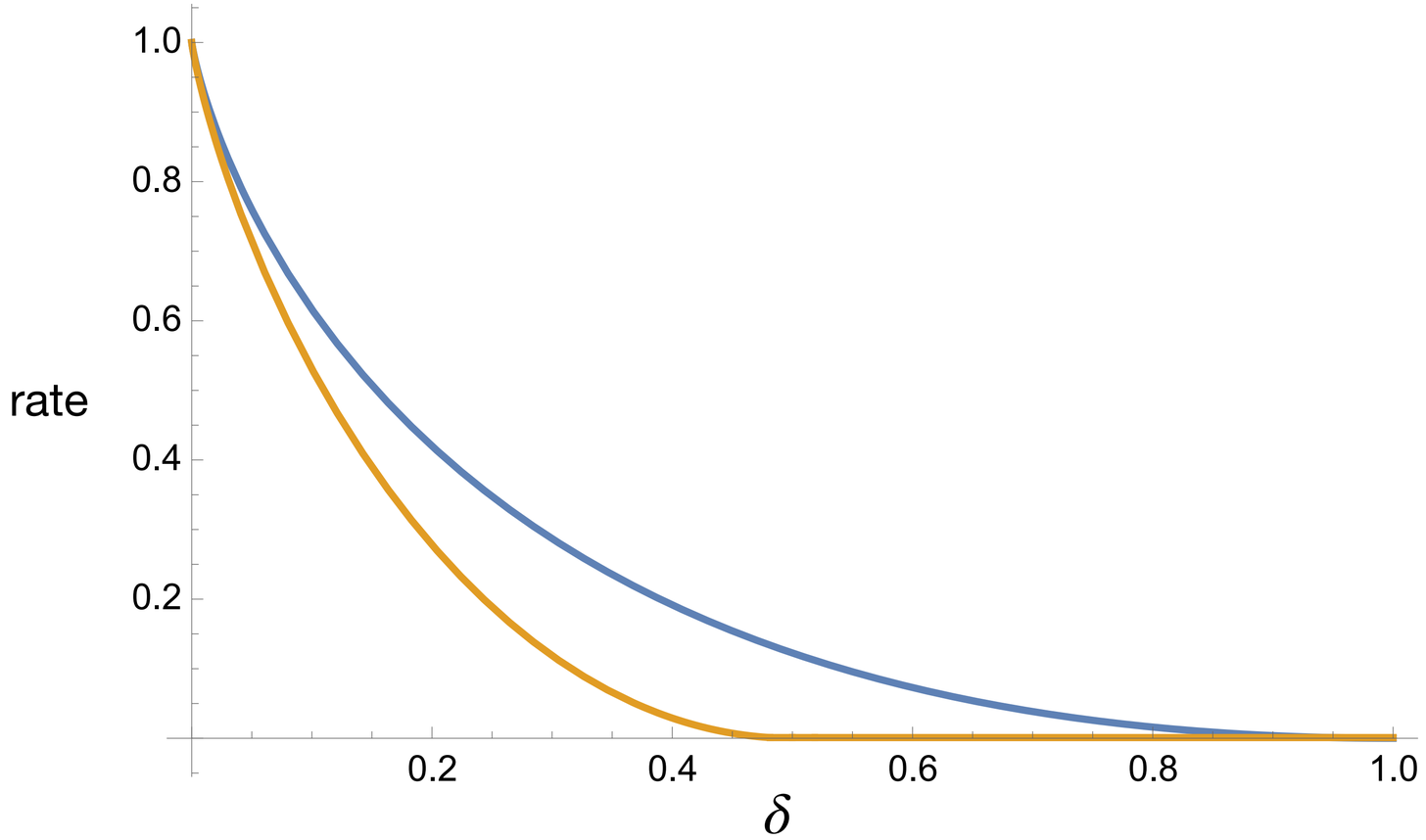
intricate explicit construction
[Guruswami-Haeupler-Shahrasbi '20]

$\Longrightarrow$ $C_{\text{ins}}(\delta) > 0$ for all $\delta < 1$.

In both cases, the lower bound is tiny!

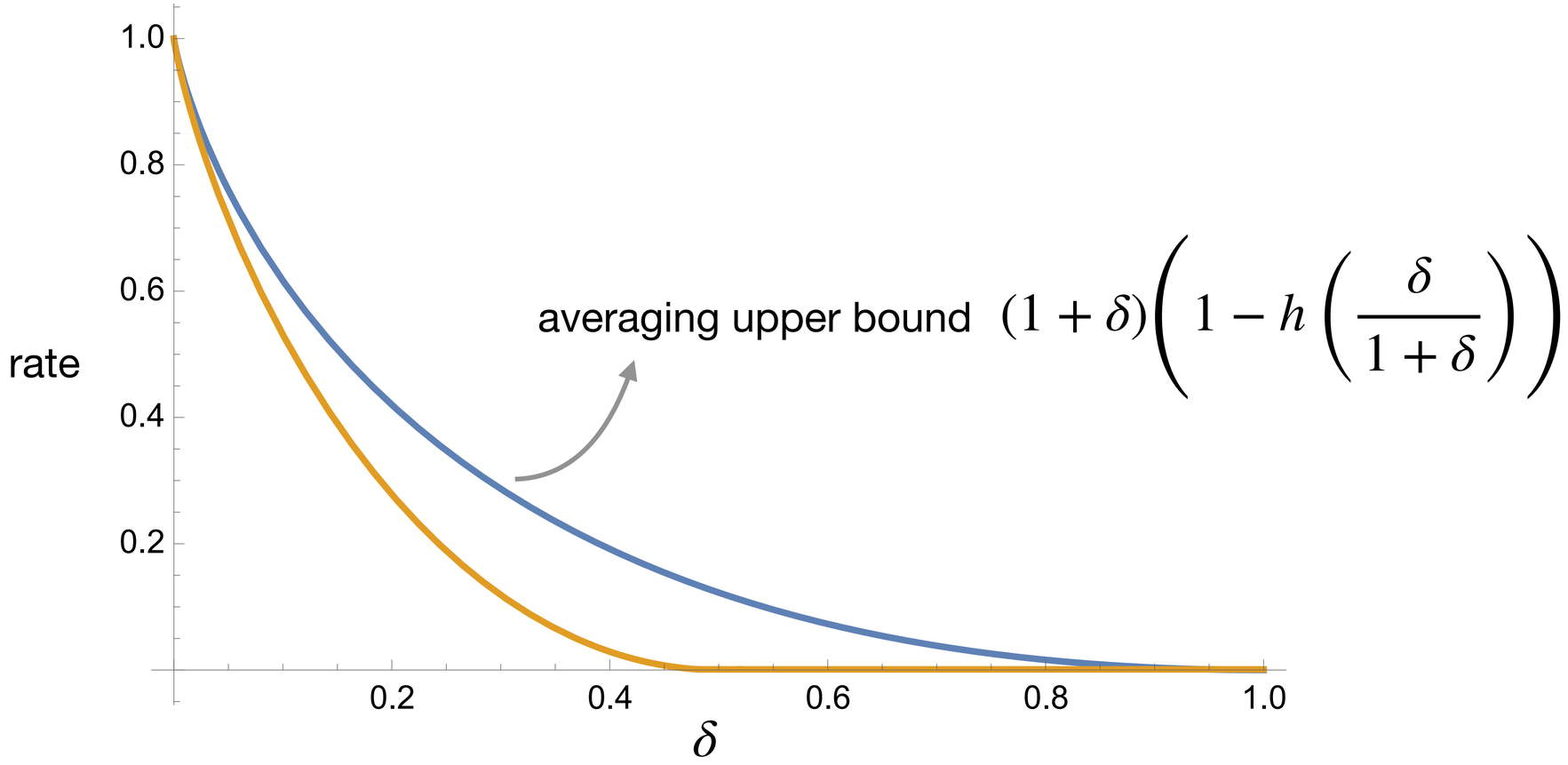
Capacity of list-decoding from insertions — the story so far

Summarizing:



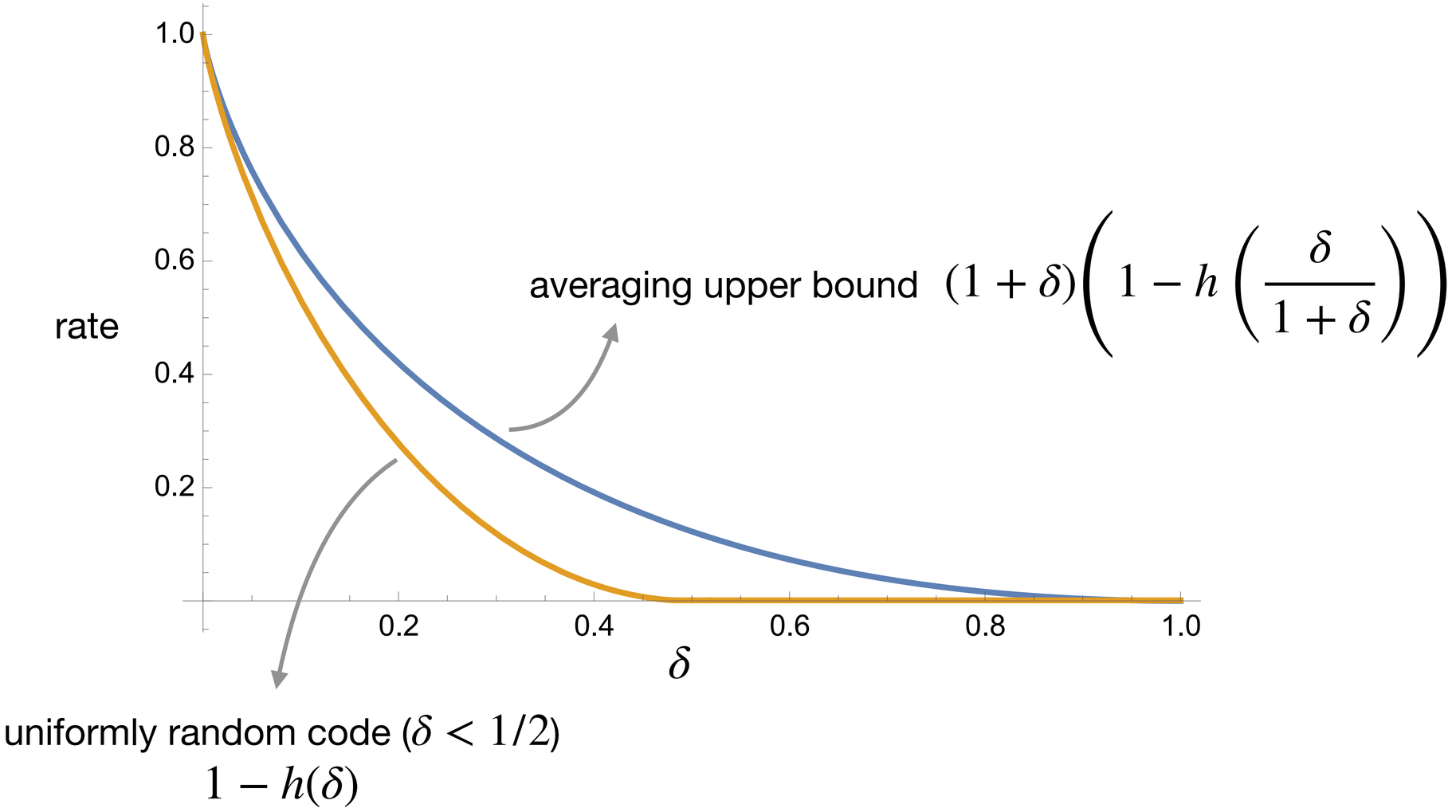
Capacity of list-decoding from insertions — the story so far

Summarizing:



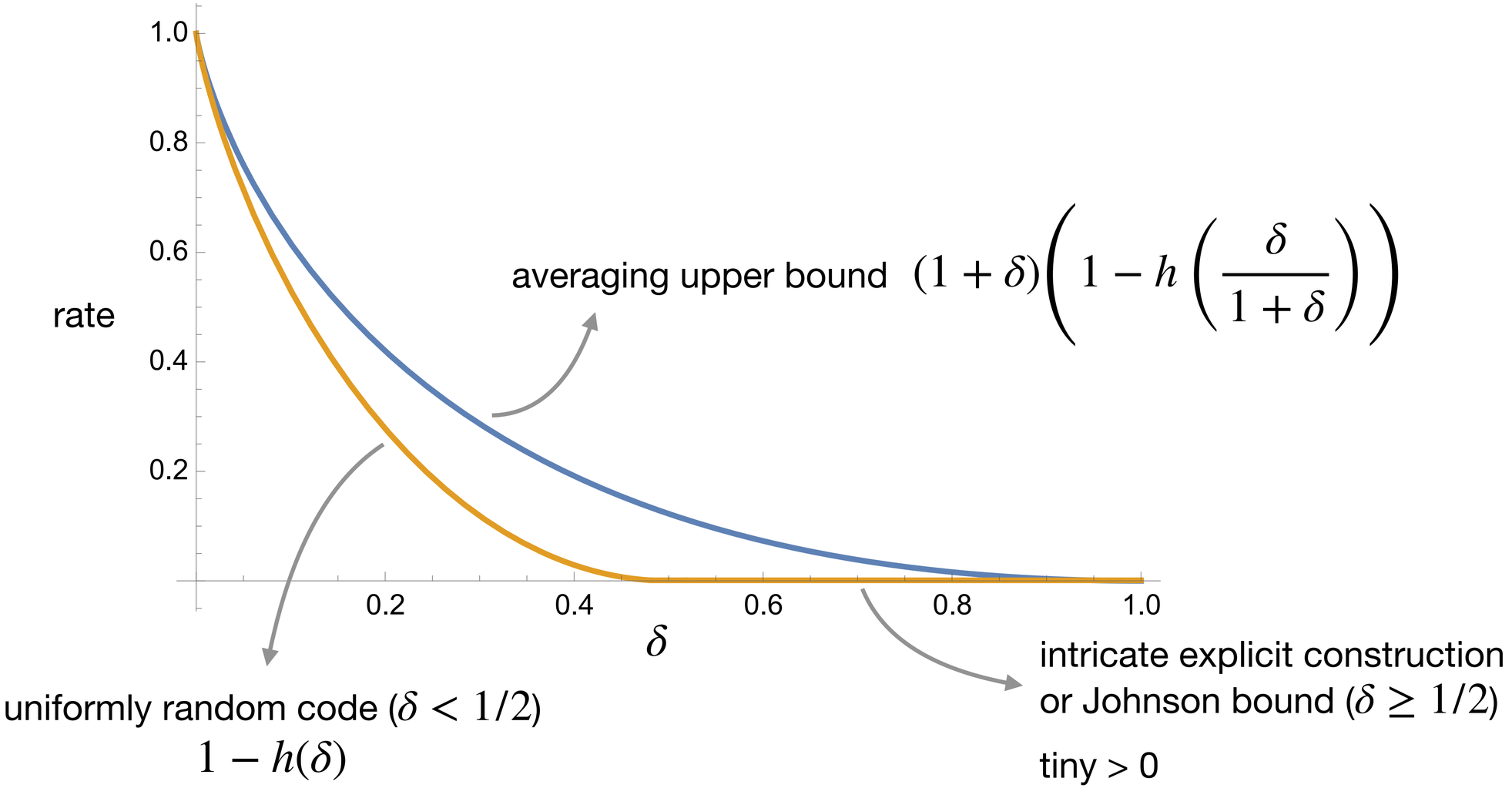
Capacity of list-decoding from insertions — the story so far

Summarizing:



Capacity of list-decoding from insertions — the story so far

Summarizing:

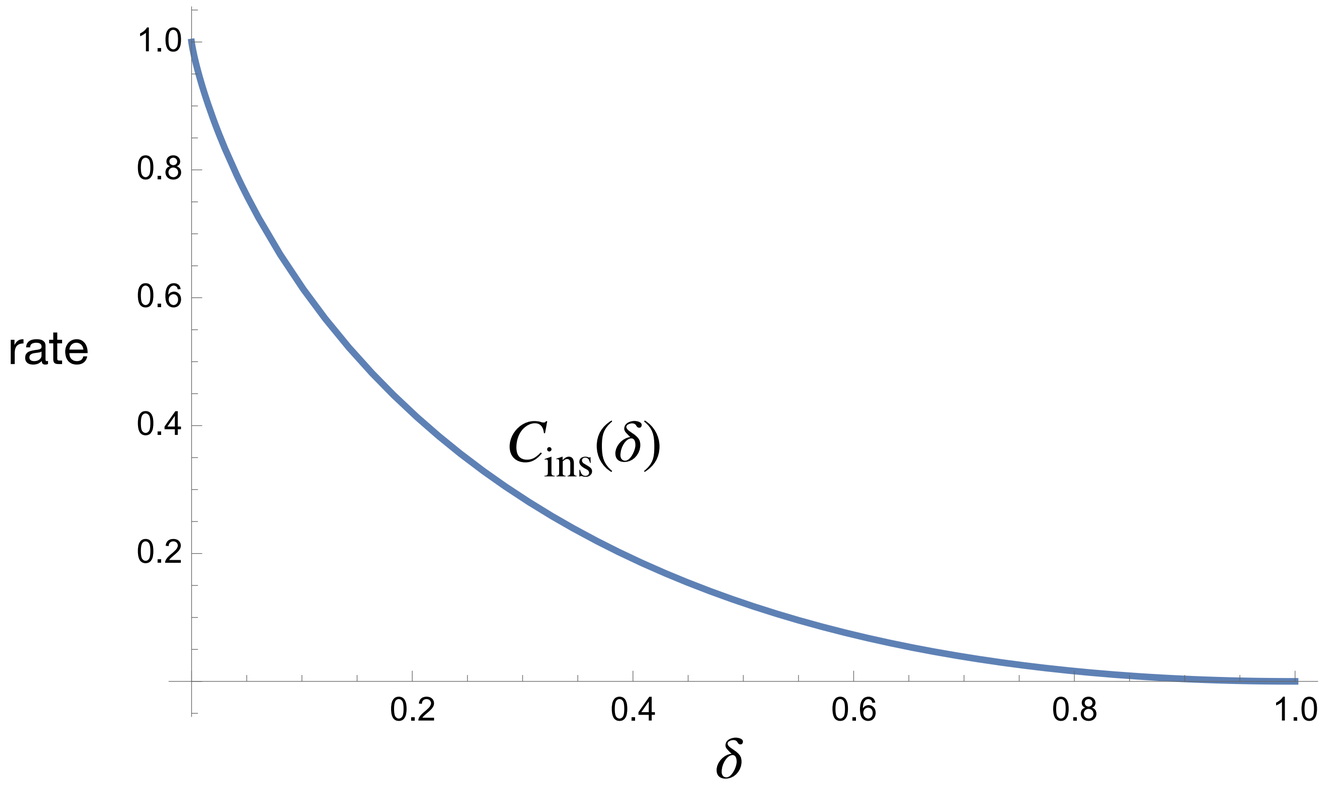


Our main result

$$C_{\text{ins}}(\delta) = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for all } \delta \in [0,1].$$

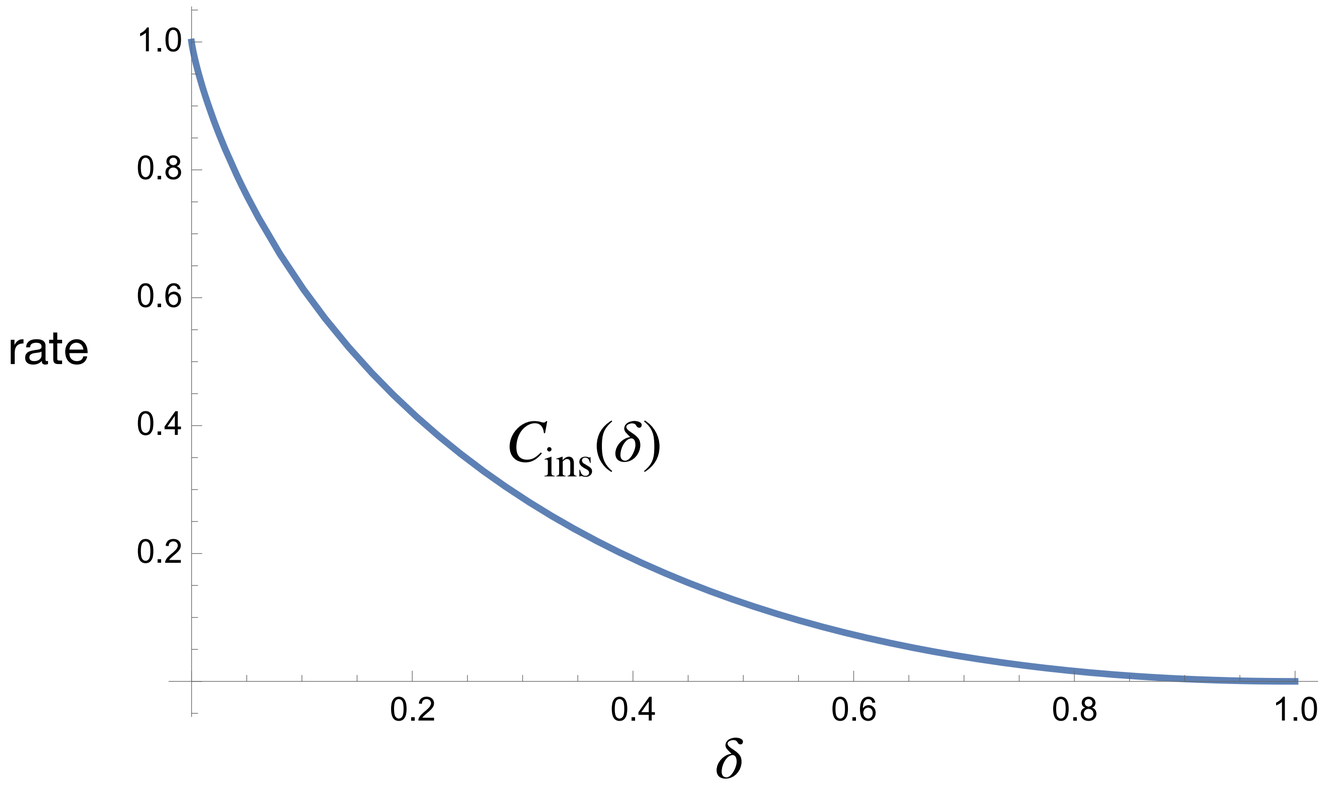
Our main result

$$C_{\text{ins}}(\delta) = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for all } \delta \in [0,1].$$



Our main result

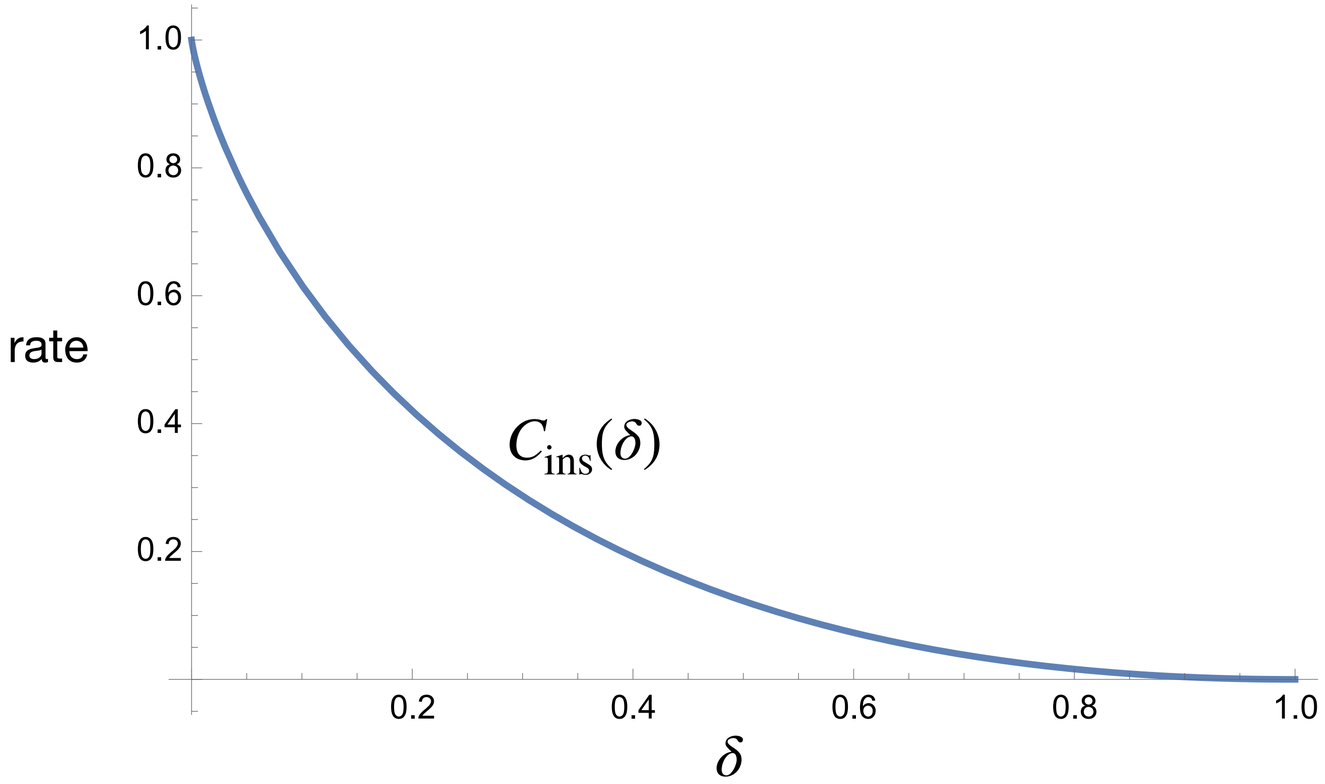
$$C_{\text{ins}}(\delta) = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for all } \delta \in [0,1].$$



We actually prove something stronger:

Our main result

$$C_{\text{ins}}(\delta) = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for all } \delta \in [0,1].$$

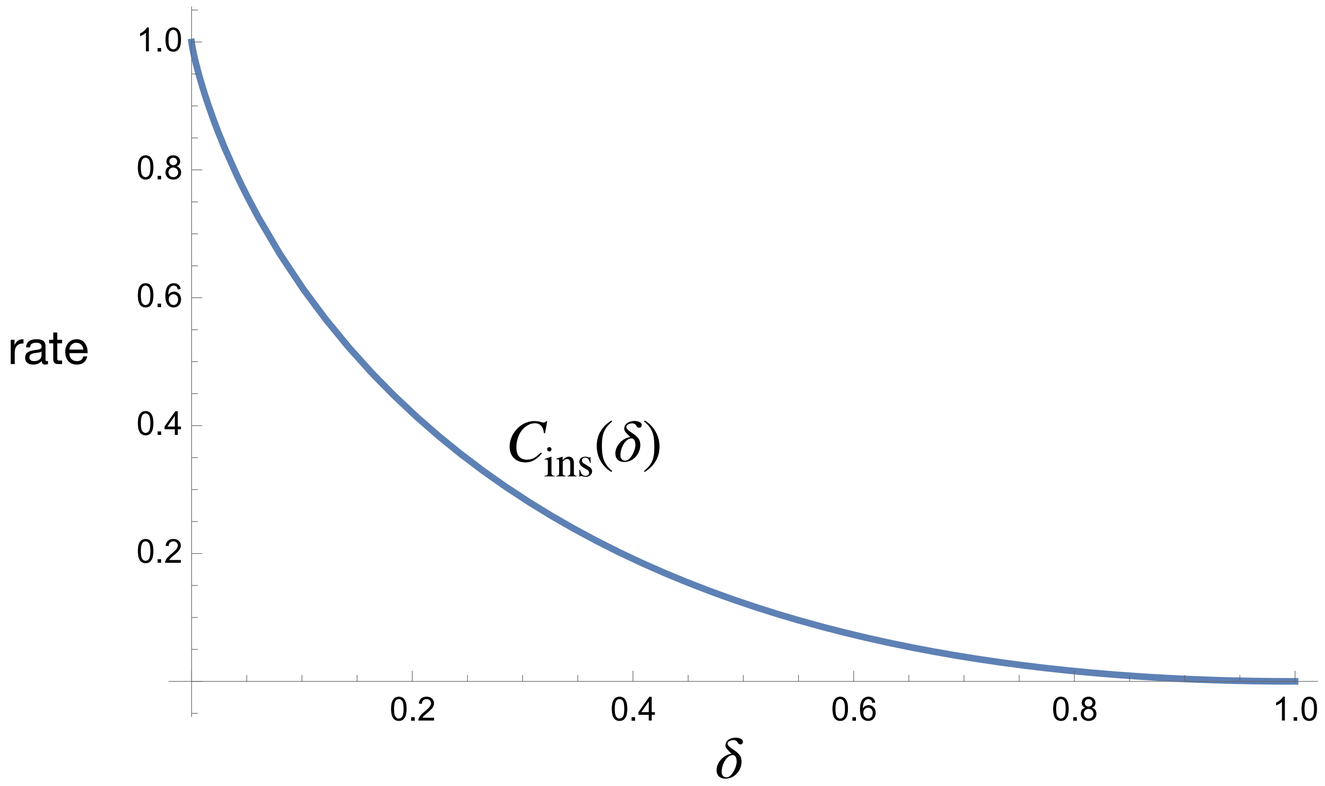


We actually prove something stronger:

- At rates ε-above capacity the list size is always at least $2^{\varepsilon n}$ **(prev. known)**;

Our main result

$$C_{\text{ins}}(\delta) = (1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right) \quad \text{for all } \delta \in [0,1].$$



We actually prove something stronger:

- At rates ε-above capacity the list size is always at least $2^{\varepsilon n}$ **(prev. known)**;
- At rates ε-below capacity we can achieve constant list size $\approx 2/\varepsilon$.

Lessons from information theory

Uniformly random codes achieve capacity on the binary symmetric channel, so it makes sense that they’re good for list-decoding from bit-flips too.

Lessons from information theory

Uniformly random codes achieve capacity on the binary symmetric channel, so it makes sense that they’re good for list-decoding from bit-flips too.

But uniformly random codes don’t achieve capacity against random insdels.

Lessons from information theory

Uniformly random codes achieve capacity on the binary symmetric channel, so it makes sense that they're good for list-decoding from bit-flips too.

But uniformly random codes don't achieve capacity against random insdels.

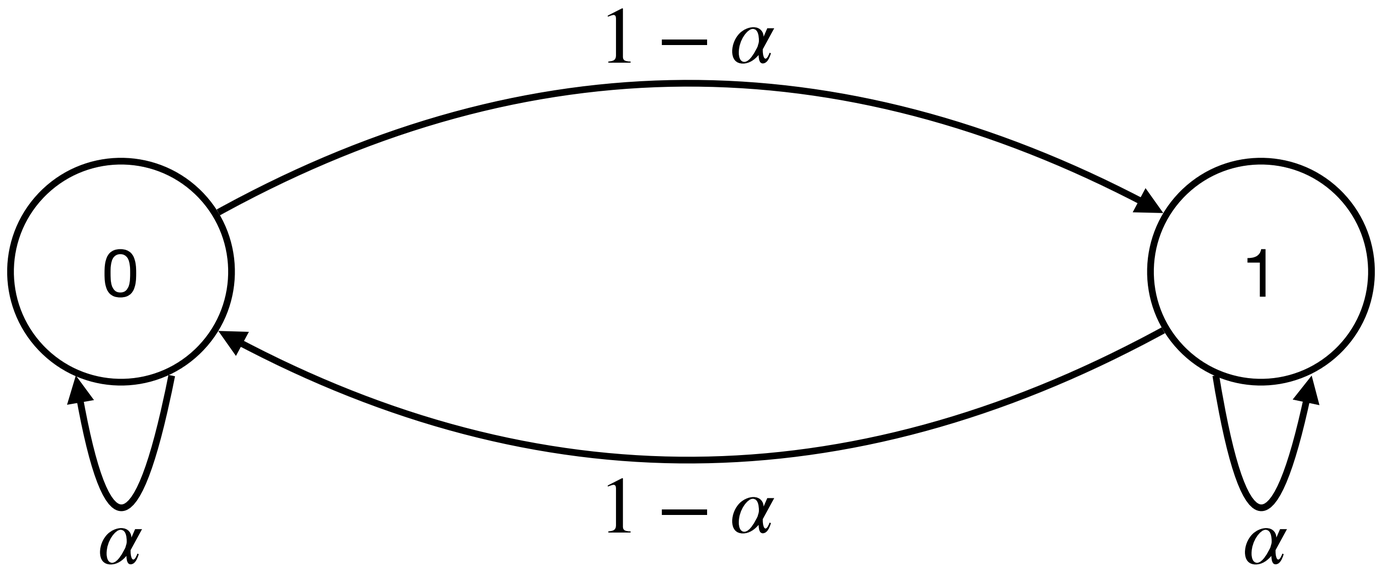
Codes sampled with memory give much better capacity lower bounds!
[Diggavi-Grossglauser '06, Drinea-Mitzenmacher '07,…]

Lessons from information theory

Uniformly random codes achieve capacity on the binary symmetric channel, so it makes sense that they're good for list-decoding from bit-flips too.

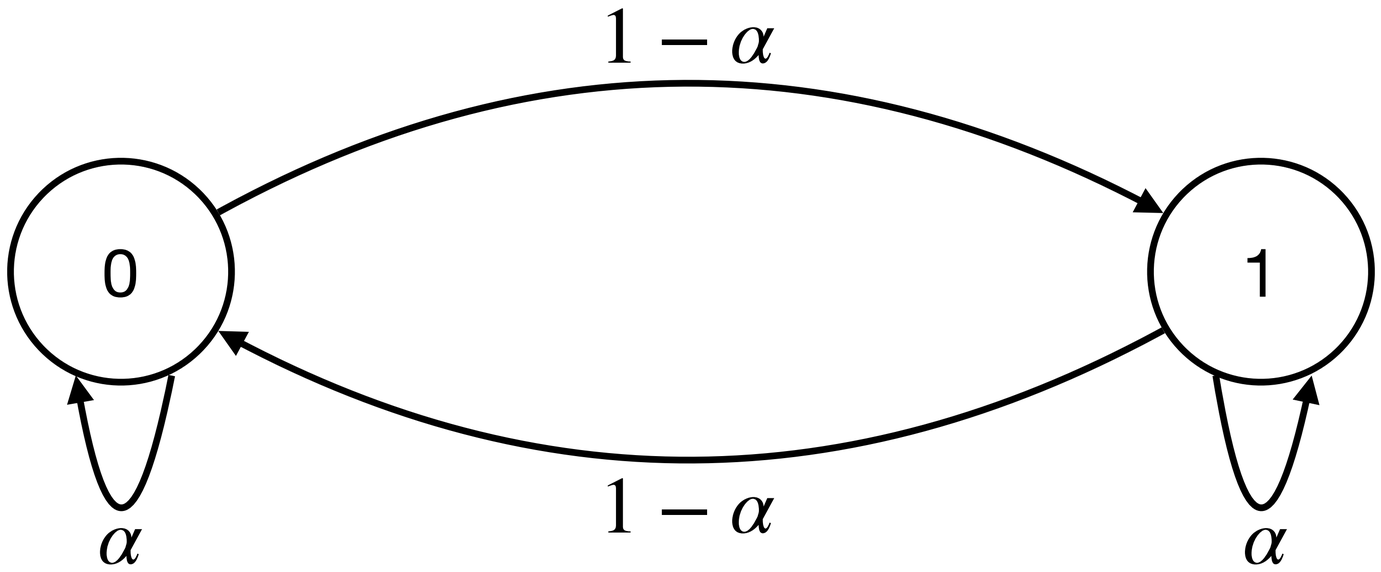
But uniformly random codes don't achieve capacity against random insdels.

Codes sampled with memory give much better capacity lower bounds!
[Diggavi-Grossglauser '06, Drinea-Mitzenmacher '07,…]



Markov codes and list-decoding from insertions

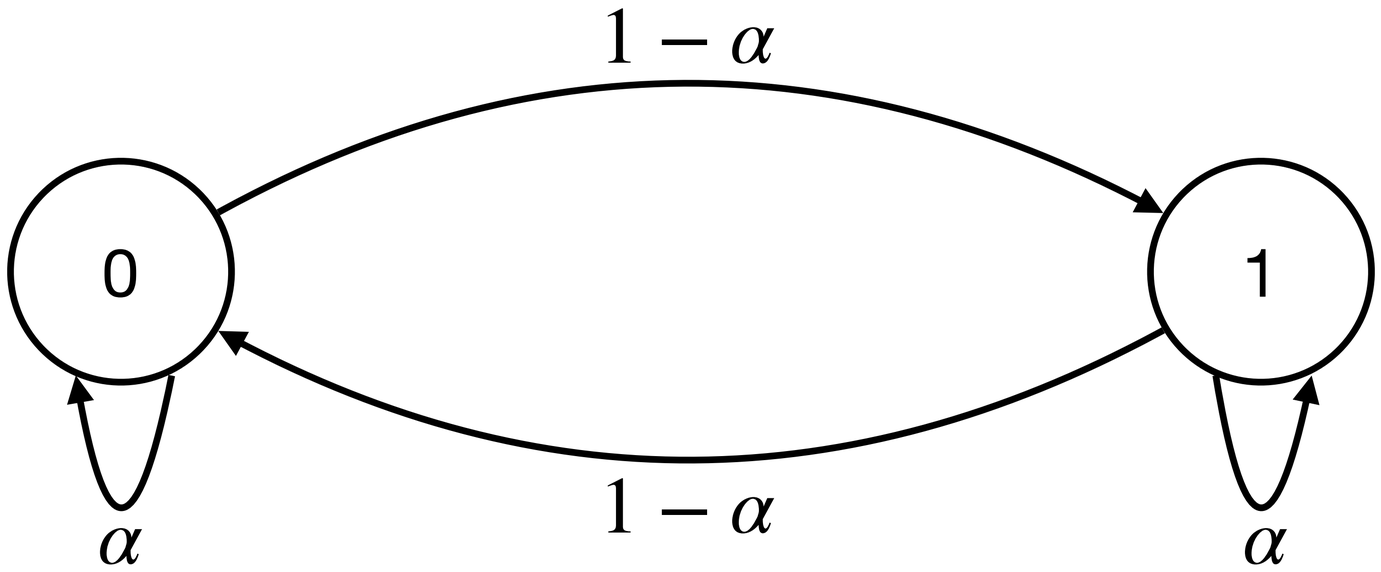
Idea: Instead of sampling a code uniformly at random, sample each codeword independently according to an order-1 Markov chain $X_1, X_2, \ldots, X_n$ with



Markov codes and list-decoding from insertions

Idea: Instead of sampling a code uniformly at random, sample each codeword independently according to an order-1 Markov chain $X_1, X_2, \ldots, X_n$ with

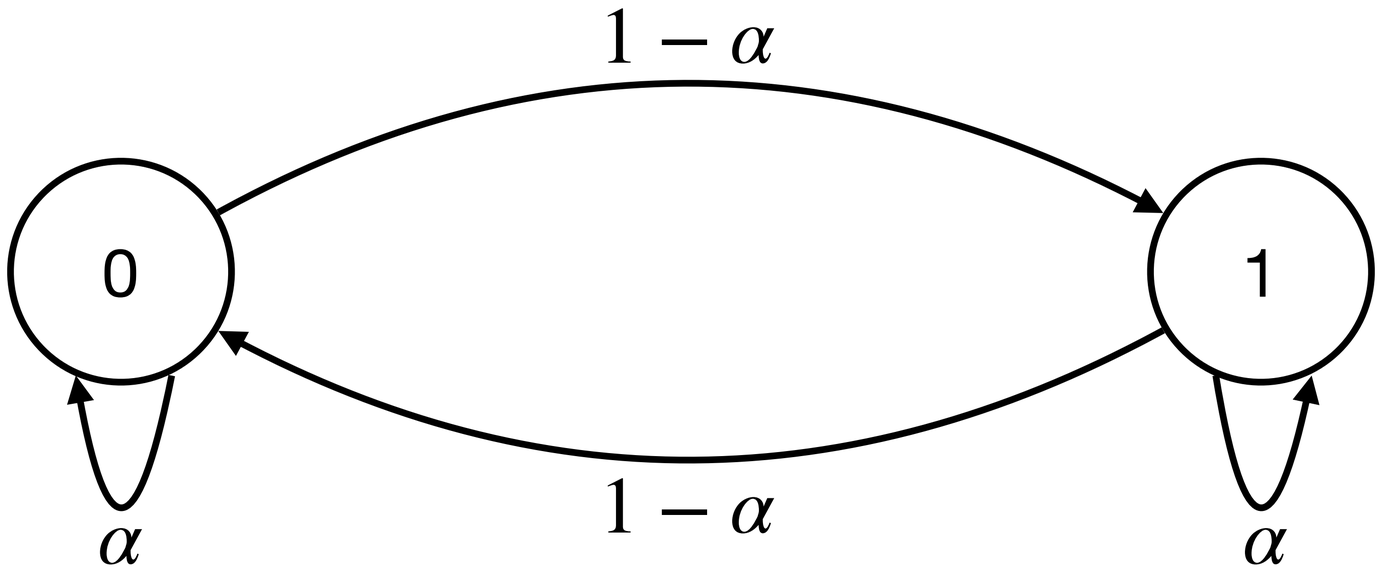
- X_1 a uniformly random bit;



Markov codes and list-decoding from insertions

Idea: Instead of sampling a code uniformly at random, sample each codeword independently according to an order-1 Markov chain $X_1, X_2, \ldots, X_n$ with

- X_1 a uniformly random bit;
- $\Pr[X_i = X_{i-1}] = \alpha$ for all $i > 1$.

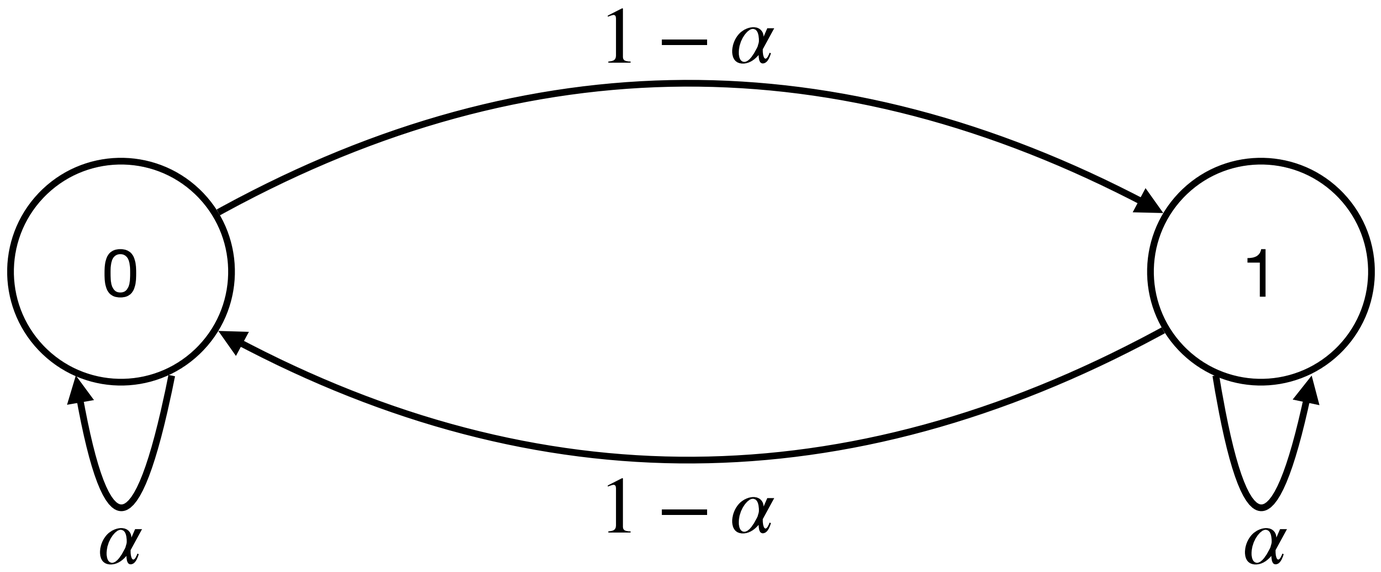


Markov codes and list-decoding from insertions

Idea: Instead of sampling a code uniformly at random, sample each codeword independently according to an order-1 Markov chain $X_1, X_2, \ldots, X_n$ with

- X_1 a uniformly random bit;
- $\Pr[X_i = X_{i-1}] = \alpha$ for all $i > 1$.

Choosing $\alpha = 1/2$ recovers uniformly random coding.



Markov codes and list-decoding from insertions

We need to upper bound

$$\Pr[X \text{ is a subsequence of } y]$$

as a function of $\alpha = \Pr[X_i = X_{i-1}]$.

Markov codes and list-decoding from insertions

We need to upper bound

$$\Pr[X \text{ is a subsequence of } y]$$

as a function of $\alpha = \Pr[X_i = X_{i-1}]$.

$N(y)$ = number of bits of X matched greedily to y.

Markov codes and list-decoding from insertions

We need to upper bound

$$\Pr[X \text{ is a subsequence of } y]$$

as a function of $\alpha = \Pr[X_i = X_{i-1}]$.

$N(y)$ = number of bits of X matched greedily to y.

$$\Pr[X \text{ is a subsequence of } y] = \Pr[N(y) = n]$$

Markov codes and list-decoding from insertions

We need to upper bound

$$\Pr[X \text{ is a subsequence of } y]$$

as a function of $\alpha = \Pr[X_i = X_{i-1}]$.

$N(y)$ = number of bits of X matched greedily to y.

$$\Pr[X \text{ is a subsequence of } y] = \Pr[N(y) = n]$$

Idea: Control moment-generating function of $N(y)$, then apply Chernoff-type bound.

Markov codes and list-decoding from insertions

We need to upper bound

$$\Pr[X \text{ is a subsequence of } y]$$

as a function of $\alpha = \Pr[X_i = X_{i-1}]$.

$N(y)$ = number of bits of X matched greedily to y.

$$\Pr[X \text{ is a subsequence of } y] = \Pr[N(y) = n]$$

Idea: Control moment-generating function of $N(y)$, then apply Chernoff-type bound.

For $\alpha = \dfrac{1+\delta}{2}$, $\Pr[X \text{ is a subsequence of } y] \leq 2^{-(1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right)n}$

Markov codes and list-decoding from insertions

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

	$y = 0 \circ z$	$y = 1 \circ z$
$X_1 = 0, X_2 = 0$	$N_0(y) = 1 + N_0(z)$	$N_0(y) = N_0(z)$
$X_1 = 0, X_2 = 1$	$N_0(y) = 1 + N_1(z)$	$N_0(y) = N_0(z)$
$X_1 = 1, X_2 = 0$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_0(z)$
$X_1 = 1, X_2 = 1$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_1(z)$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

	$y = 0 \circ z$	$y = 1 \circ z$
$X_1 = 0, X_2 = 0$	$N_0(y) = 1 + N_0(z)$	$N_0(y) = N_0(z)$
$X_1 = 0, X_2 = 1$	$N_0(y) = 1 + N_1(z)$	$N_0(y) = N_0(z)$
$X_1 = 1, X_2 = 0$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_0(z)$
$X_1 = 1, X_2 = 1$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_1(z)$

$$\tau = \ln \frac{1}{2\alpha - 1} \implies M(\tau, b \circ z) \leq \frac{\alpha}{2\alpha - 1} \cdot M(\tau, z)$$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

	$y = 0 \circ z$	$y = 1 \circ z$
$X_1 = 0, X_2 = 0$	$N_0(y) = 1 + N_0(z)$	$N_0(y) = N_0(z)$
$X_1 = 0, X_2 = 1$	$N_0(y) = 1 + N_1(z)$	$N_0(y) = N_0(z)$
$X_1 = 1, X_2 = 0$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_0(z)$
$X_1 = 1, X_2 = 1$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_1(z)$

$$\tau = \ln \frac{1}{2\alpha - 1} \implies M(\tau, b \circ z) \leq \frac{\alpha}{2\alpha - 1} \cdot M(\tau, z) \implies M(\tau, y) \leq \left(\frac{\alpha}{2\alpha - 1} \right)^{(1+\delta)n} \cdot M(\tau, \epsilon)$$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

	$y = 0 \circ z$	$y = 1 \circ z$
$X_1 = 0, X_2 = 0$	$N_0(y) = 1 + N_0(z)$	$N_0(y) = N_0(z)$
$X_1 = 0, X_2 = 1$	$N_0(y) = 1 + N_1(z)$	$N_0(y) = N_0(z)$
$X_1 = 1, X_2 = 0$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_0(z)$
$X_1 = 1, X_2 = 1$	$N_1(y) = N_1(z)$	$N_1(y) = 1 + N_1(z)$

$$\tau = \ln \frac{1}{2\alpha - 1} \implies M(\tau, b \circ z) \leq \frac{\alpha}{2\alpha - 1} \cdot M(\tau, z) \implies M(\tau, y) \leq \left(\frac{\alpha}{2\alpha - 1} \right)^{(1+\delta)n}$$

~~$M(\tau, \epsilon)$~~ 1

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

$$M\left(\ln \frac{1}{2\alpha - 1}, y\right) \leq \left(\frac{\alpha}{2\alpha - 1}\right)^{(1+\delta)n}$$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

$$M\left(\ln \frac{1}{2\alpha - 1}, y\right) \leq \left(\frac{\alpha}{2\alpha - 1}\right)^{(1+\delta)n}$$

$$\Pr[N(y) = n] \leq \left(\frac{\alpha^{1+\delta}}{(2\alpha - 1)^{\delta}}\right)^{n}$$

Markov codes and list-decoding from insertions

$N(y)$ = number of bits of X matched greedily to y.

$M(\tau, y) = \mathbb{E}[e^{\tau N(y)}]$

$\Pr[N(y) = n] \leq e^{-\tau n} M(\tau, y)$

$N_b(y)$ = number of bits of X matched greedily to y, conditioned on $X_1 = b$.

$$M\left(\ln \frac{1}{2\alpha - 1}, y\right) \leq \left(\frac{\alpha}{2\alpha - 1}\right)^{(1+\delta)n}$$

$$\Pr[N(y) = n] \leq \left(\frac{\alpha^{1+\delta}}{(2\alpha - 1)^{\delta}}\right)^{n} \implies \Pr[N(y) = n] \leq 2^{-(1+\delta)\left(1 - h\left(\frac{\delta}{1+\delta}\right)\right)n}$$

$$\alpha = \frac{1+\delta}{2}$$

Markov codes and list-decoding from deletions?

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$ [Haeupler-Shahrasbi-Sudan ’18]

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan ’18]

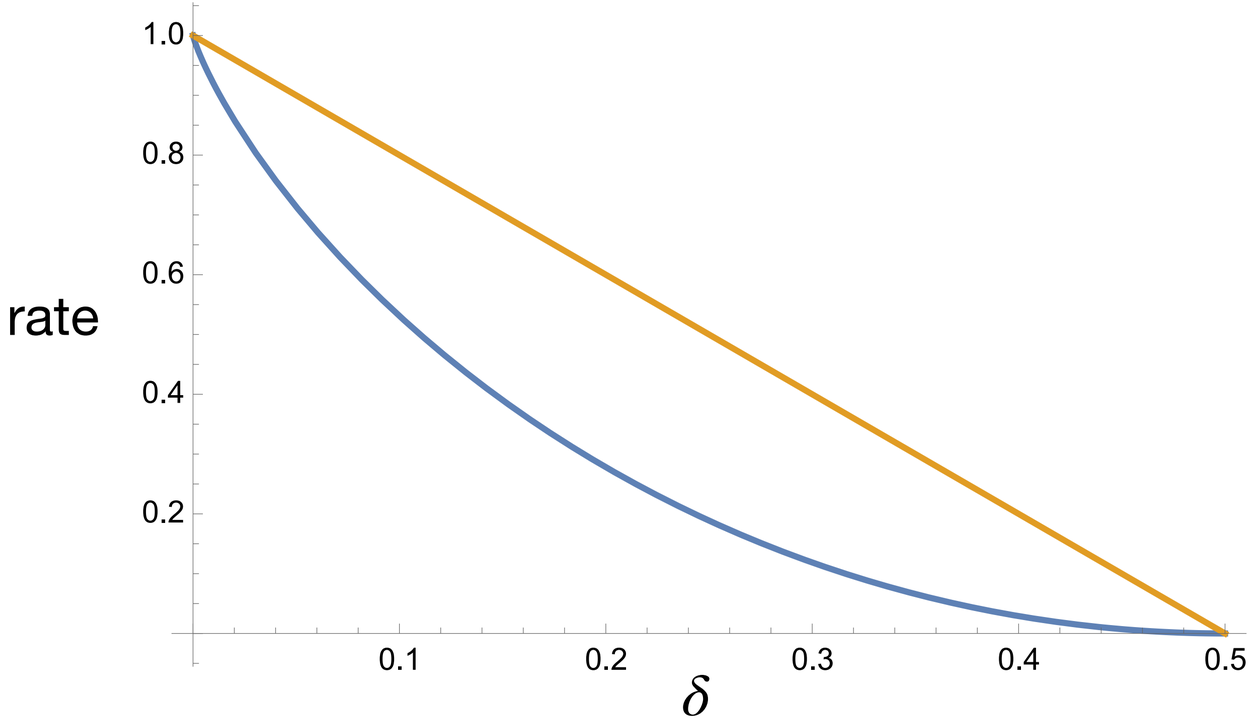
uniformly random code

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan '18]

uniformly random code



Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan ’18]

uniformly random code

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan ’18]

uniformly random code

Markov codes are optimal for list-decoding from insertions and do better than random codes over the binary deletion channel.

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan '18]

uniformly random code

Markov codes are optimal for list-decoding from insertions and do better than random codes over the binary deletion channel.

How do Markov codes perform in list-decoding from deletions?

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$ [Haeupler-Shahrasbi-Sudan '18]

uniformly random code

Markov codes are optimal for list-decoding from insertions and do better than random codes over the binary deletion channel.

How do Markov codes perform in list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}]$, there is $y \in \{0,1\}^{(1-\delta)n}$ such that

$$\Pr[y \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

Markov codes and list-decoding from deletions?

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$

[Haeupler-Shahrasbi-Sudan '18]

uniformly random code

Markov codes are optimal for list-decoding from insertions and do better than random codes over the binary deletion channel.

How do Markov codes perform in list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}]$, there is $y \in \{0,1\}^{(1-\delta)n}$ such that

$$\Pr[y \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

$\Longrightarrow$ **Markov codes never do better than random!**

Markov codes and list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}] > 1/2$,

$$\Pr[0^{(1-\delta)n} \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

Markov codes and list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}] > 1/2$,

$$\Pr[0^{(1-\delta)n} \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

But perhaps we can do better than random by expurgating the random code!

Markov codes and list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}] > 1/2$,

$$\Pr[0^{(1-\delta)n} \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

But perhaps we can do better than random by expurgating the random code!

We tried but didn't get anywhere…

Markov codes and list-decoding from deletions?

For any $\alpha = \Pr[X_i = X_{i-1}] > 1/2$,

$$\Pr[0^{(1-\delta)n} \text{ is a subsequence of } X] \geq 2^{-(1-h(\delta)+o(1))n}.$$

But perhaps we can do better than random by expurgating the random code!

We tried but didn't get anywhere…

Perhaps you have good ideas. :)

Insertions vs. Deletions

Insertions vs. Deletions

- Equivalent for unique decoding, but not for list-decoding;

Insertions vs. Deletions

- Equivalent for unique decoding, but not for list-decoding;
- In list-decoding from **insertions**, order-1 Markov codes are optimal;

Insertions vs. Deletions

- Equivalent for unique decoding, but not for list-decoding;
- In list-decoding from **insertions**, order-1 Markov codes are optimal;
- In list-decoding from **deletions**, order-1 Markov codes never do better than random;

Insertions vs. Deletions

- Equivalent for unique decoding, but not for list-decoding;
- In list-decoding from **insertions**, order-1 Markov codes are optimal;
- In list-decoding from **deletions**, order-1 Markov codes never do better than random;
- For the binary deletion channel, order-1 Markov codes do much better than random, but don’t achieve capacity.

Insertions vs. Deletions

- Equivalent for unique decoding, but not for list-decoding;
- In list-decoding from **insertions**, order-1 Markov codes are optimal;
- In list-decoding from **deletions**, order-1 Markov codes never do better than random;
- For the binary deletion channel, order-1 Markov codes do much better than random, but don’t achieve capacity.

What’s going on??

A better upper bound on the deletion list-decoding capacity

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$ [Haeupler-Shahrasbi-Sudan ’18]

A better upper bound on the deletion list-decoding capacity

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$ [Haeupler-Shahrasbi-Sudan '18]

Sharp upper bound for small deletion rate δ:

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

A better upper bound on the deletion list-decoding capacity

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - 2\delta, \qquad \delta \in [0,1/2]$$ [Haeupler-Shahrasbi-Sudan '18]

Sharp upper bound for small deletion rate δ:

$$1 - h(\delta) \leq C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

- Uniformly random coding is essentially optimal for small δ;
- Matches asymptotic behavior of deletion channel capacity!

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \geq 2^{-(1-\delta)n} \sum_{c \in C} |B_{\delta}^{\mathrm{del}}(c)|$.

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \geq 2^{-(1-\delta)n} \sum_{c \in C} |B_{\delta}^{\mathrm{del}}(c)|$.

Idea: If C has high rate, then most codewords of C must have many runs. And strings with many runs have many subsequences. So the list size must be large.

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \geq 2^{-(1-\delta)n} \sum_{c \in C} |B_{\delta}^{\mathrm{del}}(c)|$.

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \geq 2^{-(1-\delta)n} \sum_{c \in C} |B_{\delta}^{\mathrm{del}}(c)|$.

If C has rate $R > h(\gamma)$, then most codewords of C have at least γn runs.

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \leq 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \geq 2^{-(1-\delta)n} \sum_{c \in C} |B_\delta^{\mathrm{del}}(c)|$.

If C has rate $R > h(\gamma)$, then most codewords of C have at least γn runs.

A string with γn runs has at least $\binom{\gamma n - \delta n}{\delta n} \approx 2^{(\gamma-\delta)h\left(\frac{\delta}{\gamma-\delta}\right)n}$ length-$(1-\delta)n$ subsequences.

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \le 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \ge 2^{-(1-\delta)n} \sum_{c \in C} |B_\delta^{\mathrm{del}}(c)|$.

If C has rate $R > h(\gamma)$, then most codewords of C have at least γn runs.

A string with γn runs has at least $\binom{\gamma n - \delta n}{\delta n} \approx 2^{(\gamma-\delta)h\left(\frac{\delta}{\gamma-\delta}\right)n}$ length-$(1-\delta)n$ subsequences.

$$R \ge \max\left(h(\gamma), 1 - \delta - (\gamma - \delta)h\left(\frac{\delta}{\gamma - \delta}\right)\right) + \varepsilon \implies L \ge 2^{\varepsilon n}.$$

A better upper bound on the deletion list-decoding capacity

$$C_{\mathrm{del}}(\delta) \le 1 - h(\delta) + o(\delta)$$

By double counting, a code C has list size $L \ge 2^{-(1-\delta)n} \sum_{c \in C} |B_\delta^{\mathrm{del}}(c)|$.

If C has rate $R > h(\gamma)$, then most codewords of C have at least γn runs.

A string with γn runs has at least $\binom{\gamma n - \delta n}{\delta n} \approx 2^{(\gamma-\delta)h\left(\frac{\delta}{\gamma-\delta}\right)n}$ length-$(1-\delta)n$ subsequences.

$$R \ge \max\left(h(\gamma), 1 - \delta - (\gamma - \delta)h\left(\frac{\delta}{\gamma - \delta}\right)\right) + \varepsilon \implies L \ge 2^{\varepsilon n}.$$

take $\gamma = \frac{1}{2} - \sqrt{\frac{h(\delta)\ln 2}{2}}$

Wrapping up

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;

- Capacity of list-decoding from insertions and deletions is much more mysterious!

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;
- Capacity of list-decoding from insertions and deletions is much more mysterious!
- Insertion list-decoding capacity is achieved by order-1 Markov codes;

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;
- Capacity of list-decoding from insertions and deletions is much more mysterious!
- Insertion list-decoding capacity is achieved by order-1 Markov codes;
- But order-1 Markov codes aren’t better than random in list-decoding from deletions…

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;
- Capacity of list-decoding from insertions and deletions is much more mysterious!
- Insertion list-decoding capacity is achieved by order-1 Markov codes;
- But order-1 Markov codes aren’t better than random in list-decoding from deletions…

THE open problem:

Wrapping up

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;
- Capacity of list-decoding from insertions and deletions is much more mysterious!
- Insertion list-decoding capacity is achieved by order-1 Markov codes;
- But order-1 Markov codes aren't better than random in list-decoding from deletions…

THE open problem:

Beat $1 - h(\delta)$ in list-decoding from deletions??

Wrapping up

Thanks!

- Capacity of list-decoding from bit-flips is easy to work out, achieved by a uniformly random code;
- Capacity of list-decoding from insertions and deletions is much more mysterious!
- Insertion list-decoding capacity is achieved by order-1 Markov codes;
- But order-1 Markov codes aren't better than random in list-decoding from deletions…

THE open problem:

Beat $1 - h(\delta)$ in list-decoding from deletions??