CS3236 Introduction to Information theory

Lecture 4: Lossless compression

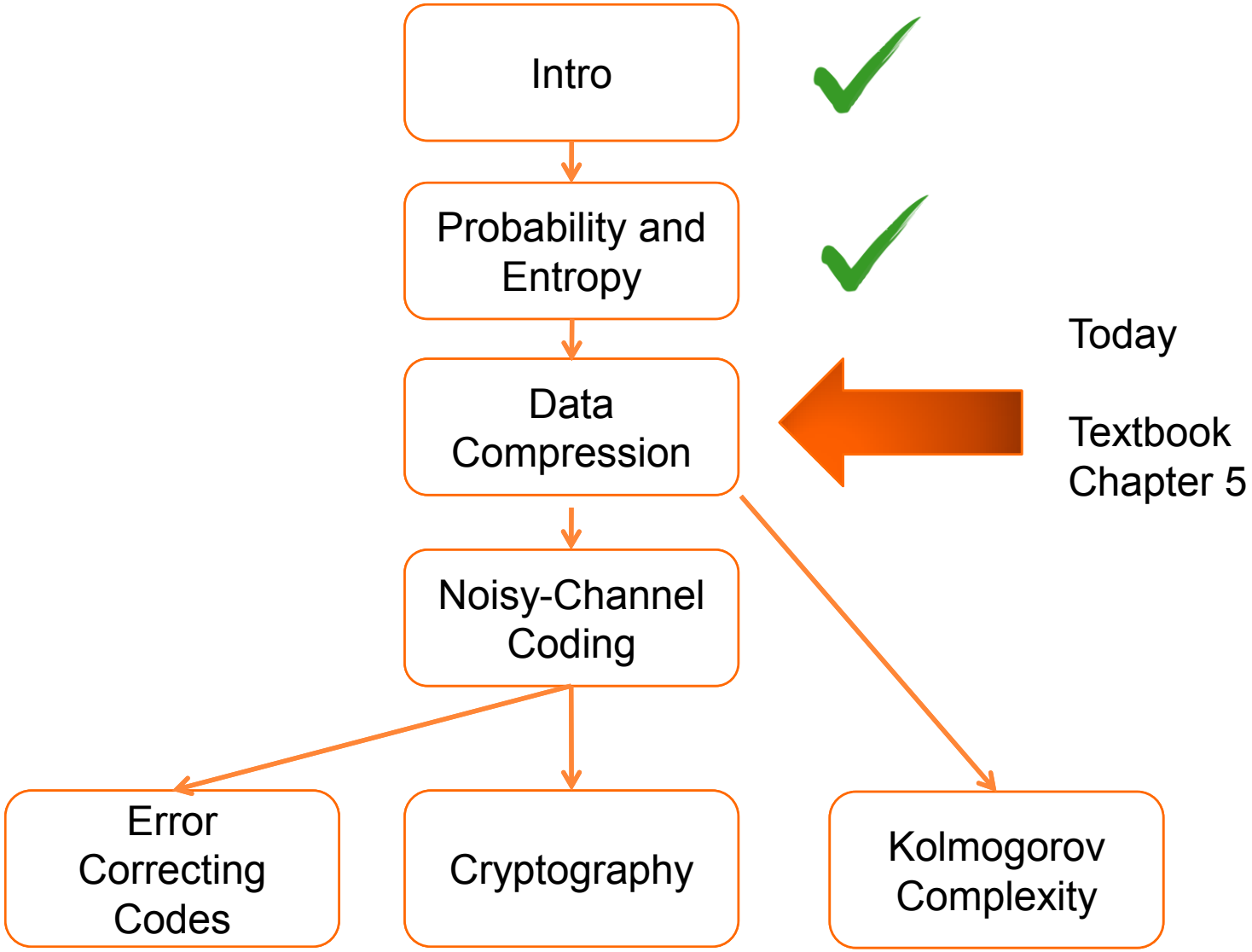
Course given by Pierre Senellart
Material by Stephanie Wehner, with additions by P. Senellart

Let's Recap…

Class survival guide

- Lecture on Monday 2pm-3:30pm
- Tutorial on Wed 4-5pm and Thu 2-3pm, COM1-02-18
  - Choose one session, no need to go to both
  - Discussion, QA, help with homework exercises, projects
- Recess week September 20-28
- No physical lecture/tutorial on Sep 8-12, e-Learning material
- No lecture on October 6 (Hari Raya Haji)
- No tutorial on week 10, October 22-23 (Deepavali)
- Available to meet for answering individual questions. Use http://www.doodle.com/psenellart to schedule a meeting. Office Icube #03-09.
- Grading
  - 50% Exam (Friday November 28)
  - 50% Continuous Assessment
    - 10% Mid-Term exam (October 13)
    - 20% Homework (assigned each week, due next Monday)
    - 20% Small project (handed out September 1, due November 14)
- If you have any doubt about the schedule, check IVLE

Where do we go from here?



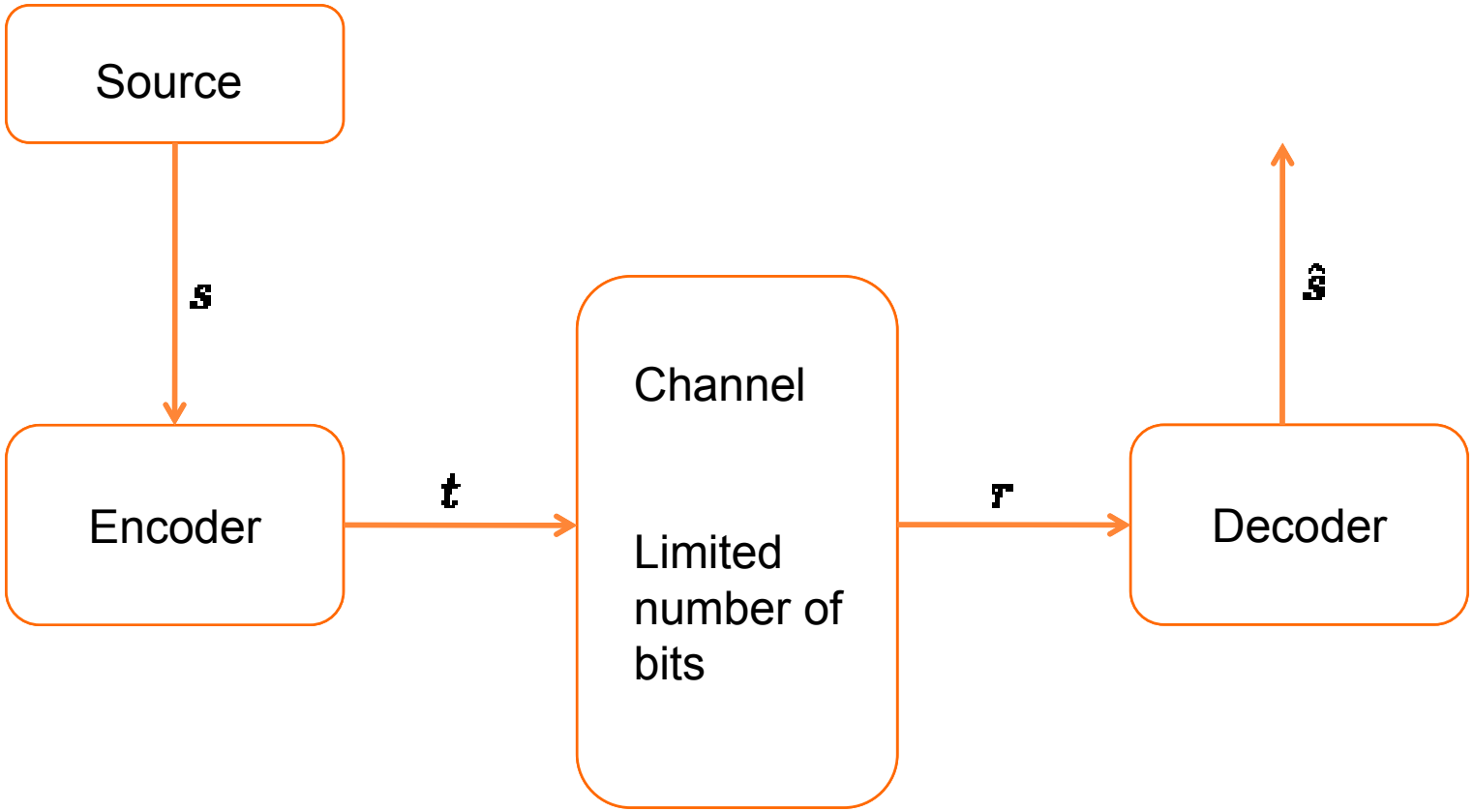
What we did last time

- Intro to data compression
- Lossy compression framework
- Statement of Shannon's source coding theorem for lossy compression
- The entropy of the source tells us exactly how well we can compress it

What we'll do today

- Lossless data compression
  - Basic concepts
  - Symbol codes
  - Kraft's inequality
  - Lossless compression and entropy
  - Shannon's source coding theorem for symbol codes
- An optimal symbol code: Huffman coding
- Limits of symbol codes

Data compression



Goal: Construct an encoder and decoder such that we can recover the information $\hat{s} = s$

Lossy and Lossless compression

- Lossy
  - maps some source symbols to the same encoding
  - failure probability δ
- Lossless compression
  - maps all possible source symbols to distinct encodings
  - works without failure probability

Lossless compression

- If we don't allow a failure probability, how can it be that we can compress at all?
- Idea
  - Use short encodings for symbols that are very likely
  - Use long encodings for symbols that are unlikely
- If we compress many times we will on compress on average as long encodings are not very frequent

Questions in Lossless compression

- Which codewords do we make shorter, and which ones longer?
- How can we make sure that our code is easy to decode in the end?
- What is the best compression that we can achieve?

Some terminology: symbol code

- A symbol code for an ensemble is a mapping

$$\mathcal{A}_X \rightarrow \{0,1\}^+$$

- Codeword $c(x)$
- Length $l(x)$

All the bit strings of length at least 1

- Example

$$\mathcal{A}_X = \{a, b, c, d\}$$
$$P_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$$

x	c(x)	l(x)
a	1000	4
b	0100	4
c	0010	4
d	0001	4

Extended code

- The extended code C^+ is what we get when we use our code to encode an entire string

$$\mathcal{A}_X = \{a, b, c, d\}$$
$$P_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$$

x	c(x)	l(x)
a	1000	4
b	0100	4
c	0010	4
d	0001	4

- Example

$$c^+(accd) = 10000010001000001$$

Uniquely decodable

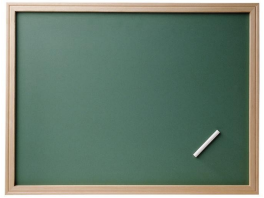
- A code is uniquely decodable if also for the extended code no two strings have the same encoding:

$$\forall x, y \in \mathcal{A}_X^+ \; x \neq y \implies c^+(x) \neq c^+(y)$$

Requirements of useful symbol codes

- The extended code must be uniquely decodable
  - Any string must have a unique decoding
- Must be easy to decode
- Achieve the highest possible compression

Easy to decode?



- Ideally we want to be able to decode while receiving the signal without storing the signal and looking ahead

- How about this code?

x	c(x)
a	0
b	011

Prefix Code

- A code is a prefix code if no codeword is a prefix of another
- Not a prefix code

x	c(x)
a	0
b	011

- Prefix code example

x	c(x)
a	1
b	011

- Does this help?

Prefix codes as trees

- Let's consider some examples

$0 \rightarrow 0$
$1 \rightarrow 10$
$2 \rightarrow 111$
$3 \rightarrow 110$

x	c(x)
a	1
b	011

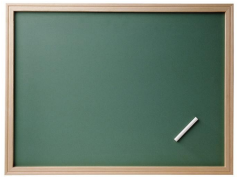
- Are they uniquely decodable?

- Complete prefix code: no unused branches

What limits does unique decodability impose on the code length?

- Let's for the moment only think about code lengths (and forget about probabilities)
- … that is, we'll worry only about unique decodability vs. code length
- If we want our code to be uniquely decodable we cannot allow some codewords to combine to others!
- Short codewords are a scarce resource!

Cost of codewords



- Budget of 1
- Cost will be $1/2^{l(x)}$

0	00	000	0000	The total symbol code budget
			0001	
		001	0010	
			0011	
	01	010	0100	
			0101	
		011	0110	
			0111	
1	10	100	1000	
			1001	
		101	1010	
			1011	
	11	110	1100	
			1101	
		111	1110	
			1111	

Kraft inequality

- There exists a uniquely decodable code if and only if the codeword lengths satisfy $$\sum_{x \in Ax} \frac{1}{2^{l(x)}} \leq 1$$

Furthermore, if such a code exists, a prefix code exists.

- Kraft: If the inequality is satisfied, then a prefix code exists with the given lengths (and prefix codes are uniquely decodable).
- McMillan: Conversely, any uniquely decodable code must satisfy this inequality.

KRAFT INEQUALITY

If equality holds the code is called a “complete code”.

- There exists a uniquely decodable code if and only if the codeword lengths satisfy

$$\sum_{x \in Ax} \frac{1}{2^{l(x)}} \leq 1$$

Furthermore, if such a code exists, a prefix code exists.

- Kraft: If the inequality is satisfied, then a prefix code exists with the given lengths (and prefix codes are uniquely decodable).
- McMillan: Conversely, any uniquely decodable code must satisfy this inequality.

KRAFT INEQUALITY

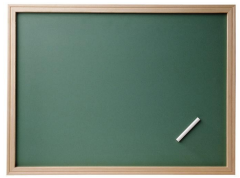
If equality holds the code is called a “complete code”.

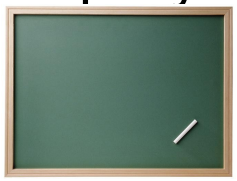
- There exists a uniquely decodable code if and only if the codeword lengths satisfy

$$\sum_{x \in Ax} \frac{1}{2^{l(x)}} \leq 1$$

Furthermore, if such a code exists, a prefix code exists.

- Kraft: If the inequality is satisfied, then a prefix code exists with the given lengths (and prefix codes are uniquely decodable).
- McMillan: Conversely, any uniquely decodable code must satisfy this inequality.



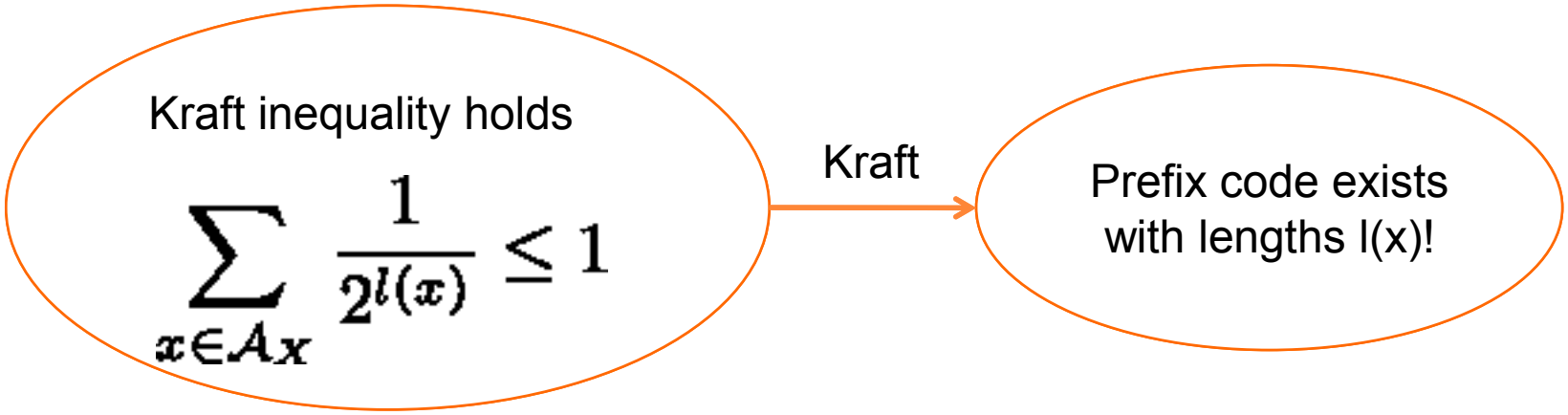


Unique decodability vs prefix free

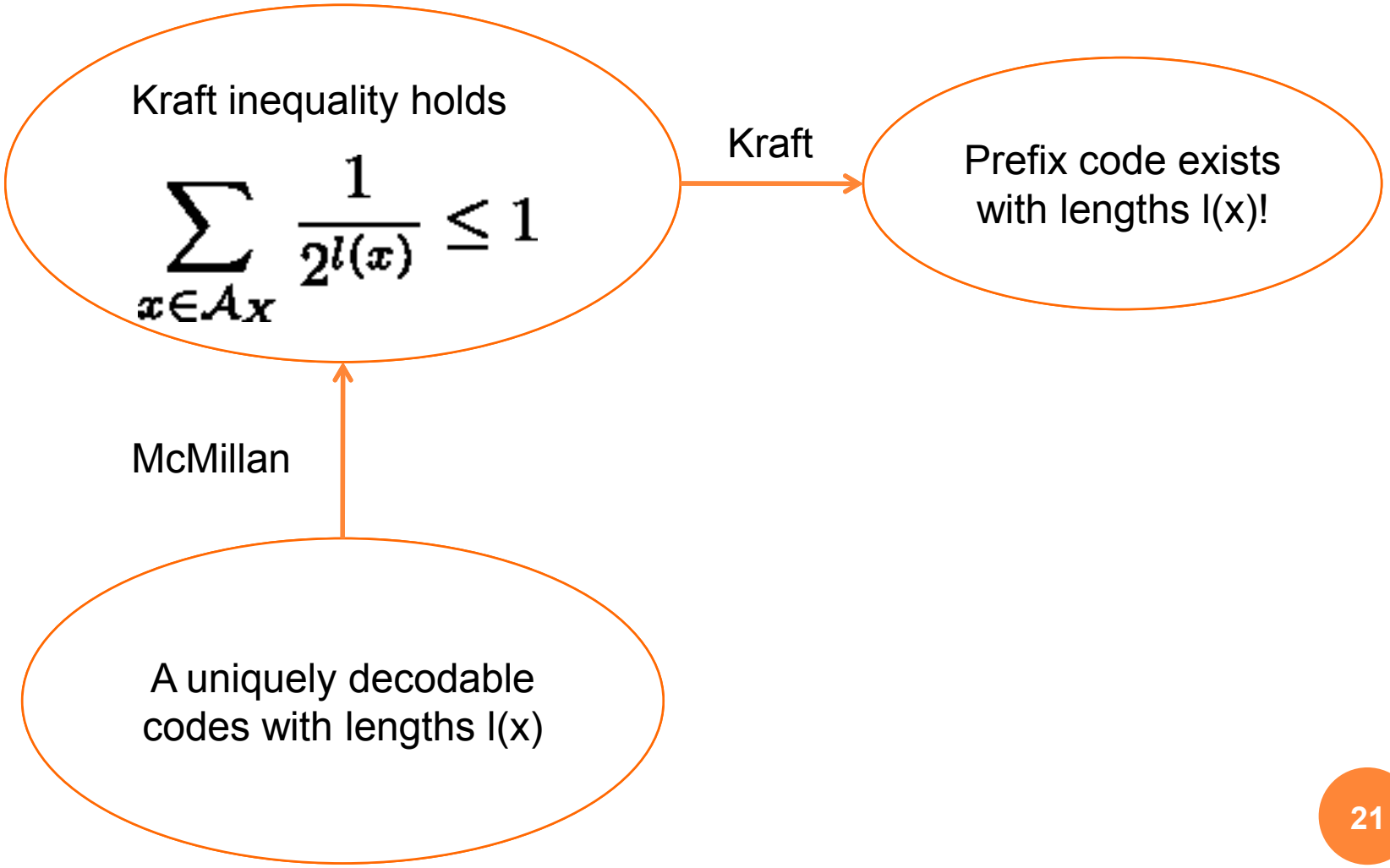
Kraft inequality holds

$$\sum_{x \in Ax} \frac{1}{2^{l(x)}} \leq 1$$

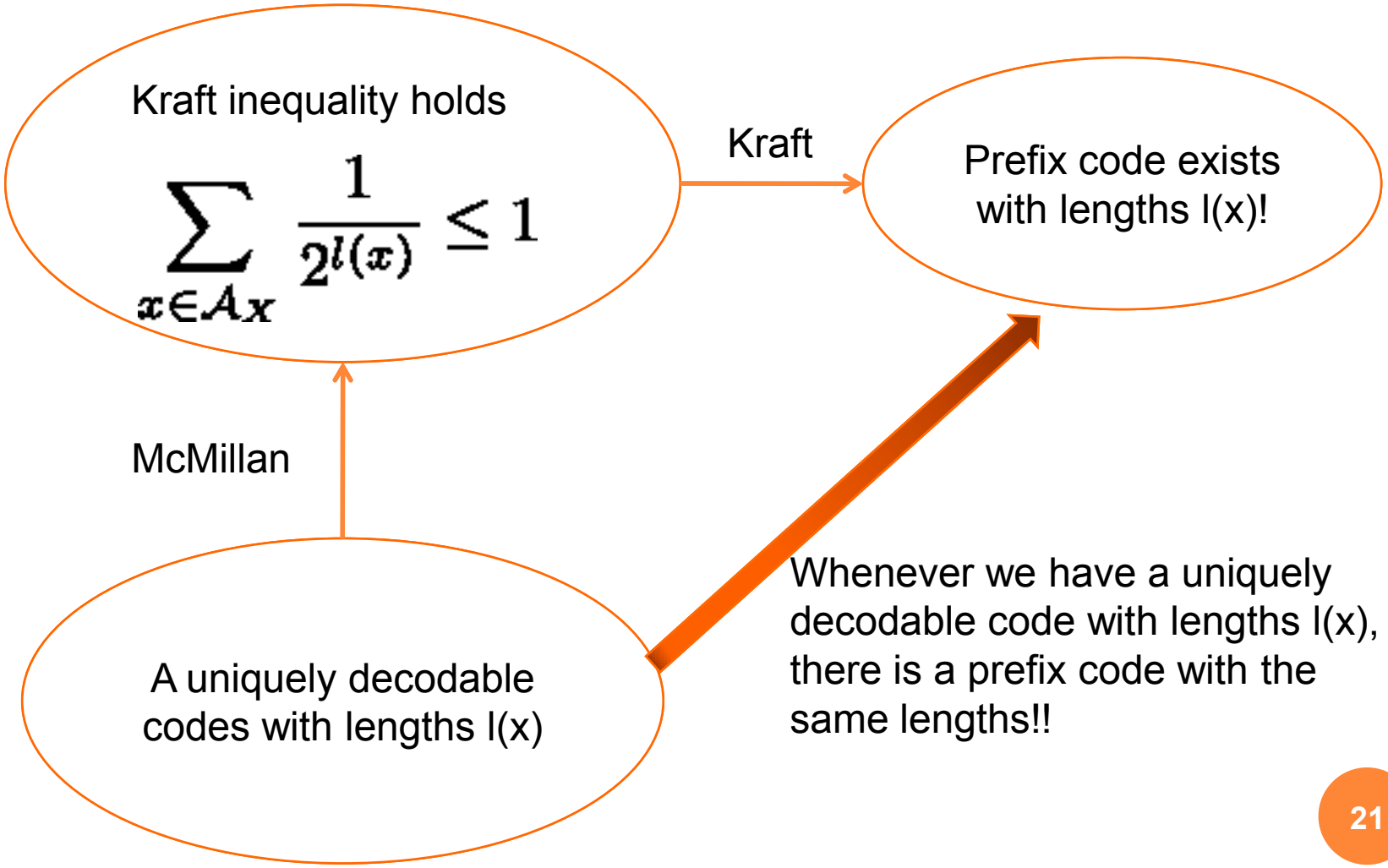
Unique decodability vs prefix free



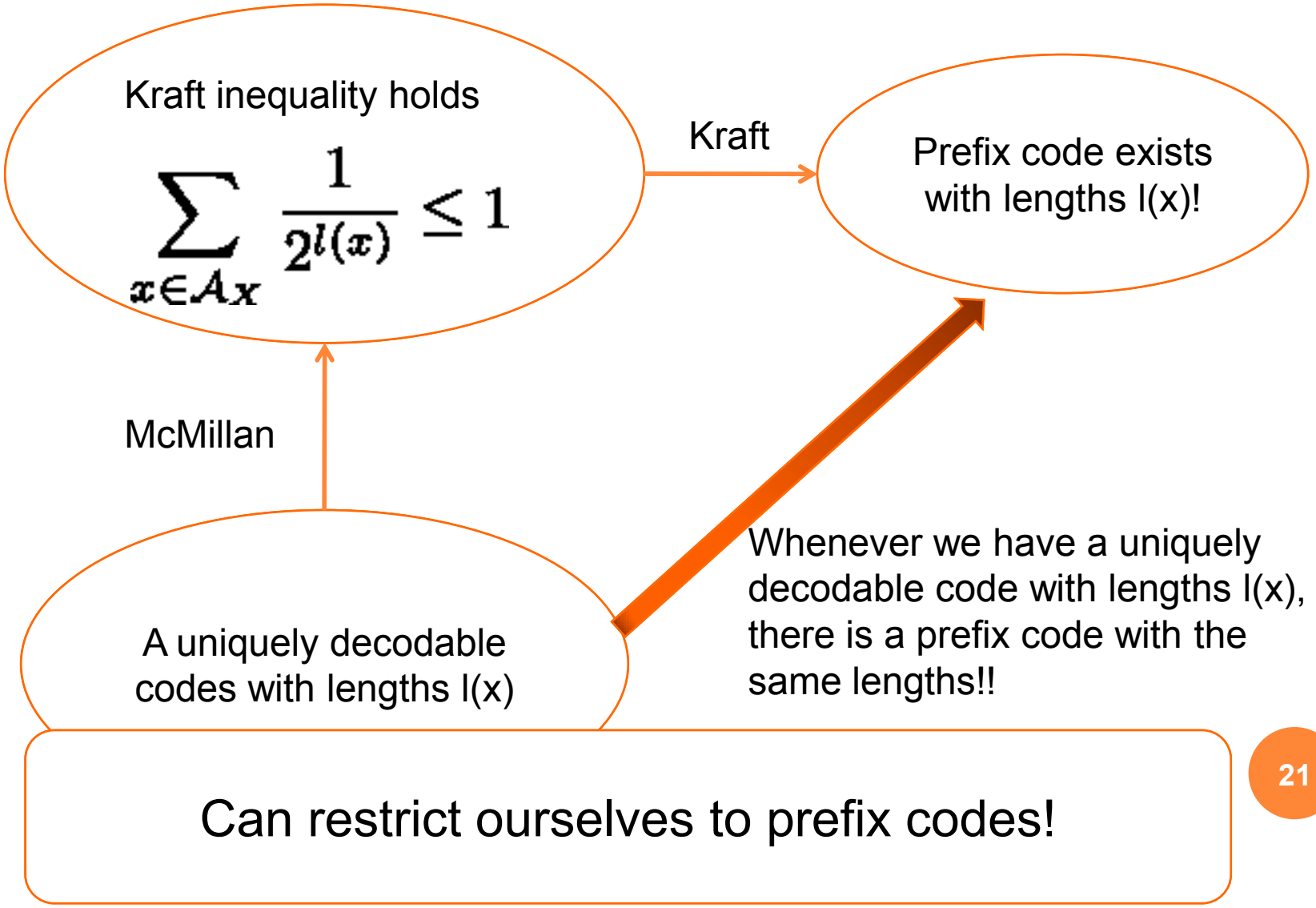
Unique decodability vs prefix free



Unique decodability vs prefix free



Unique decodability vs prefix free



Requirements of useful symbol codes

- The extended code must be uniquely decodable
  - Any string must have a unique decoding

- Must be easy to decode

- Achieve the highest possible compression
  - Remember: it's sufficient to consider prefix codes

Expected code length

- We cannot hope to reduce the total number of codewords, nor the sum of lengths of all codewords – can you say why?

- But we can hope to reduce the expected code length

$$L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$$

How well can we compress?

- Measured in terms of the average codeword length

$$L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$$

- What's the best compression we can hope for?
  - Lower bound by the Shannon entropy

$$H(X) \leq L(C, X)$$

Proof of the lower bound $H(X) \leq L(C, X)$

- Let's investigate $L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$

Proof of the lower bound $H(X) \leq L(C, X)$

- Let's investigate $L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$

Actual distribution

$P(x)$

Proof of the lower bound $H(X) \leq L(C, X)$

- Let's investigate $L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$

Actual distribution

$P(x)$

Implicit distribution

$$Q(x) = \frac{2^{-\ell(x)}}{z}$$

$$z = \sum_x 2^{-\ell(x)}$$

Proof of the lower bound $H(X) \leq L(C, X)$

- Let's investigate $L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$

Actual distribution

$$P(x)$$

Implicit distribution

$$Q(x) = \frac{2^{-\ell(x)}}{z}$$

$$z = \sum_x 2^{-\ell(x)}$$

Difference measured by the relative entropy "distance"
(Kullback-Leibler divergence)

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Proof of the lower bound $H(X) \leq L(C, X)$

- Let's investigate $L(C, X) = \sum_{x \in \mathcal{A}_X} P_X(x) l(x)$

Actual distribution

$$P(x)$$

Implicit distribution

$$Q(x) = \frac{2^{-\ell(x)}}{z}$$

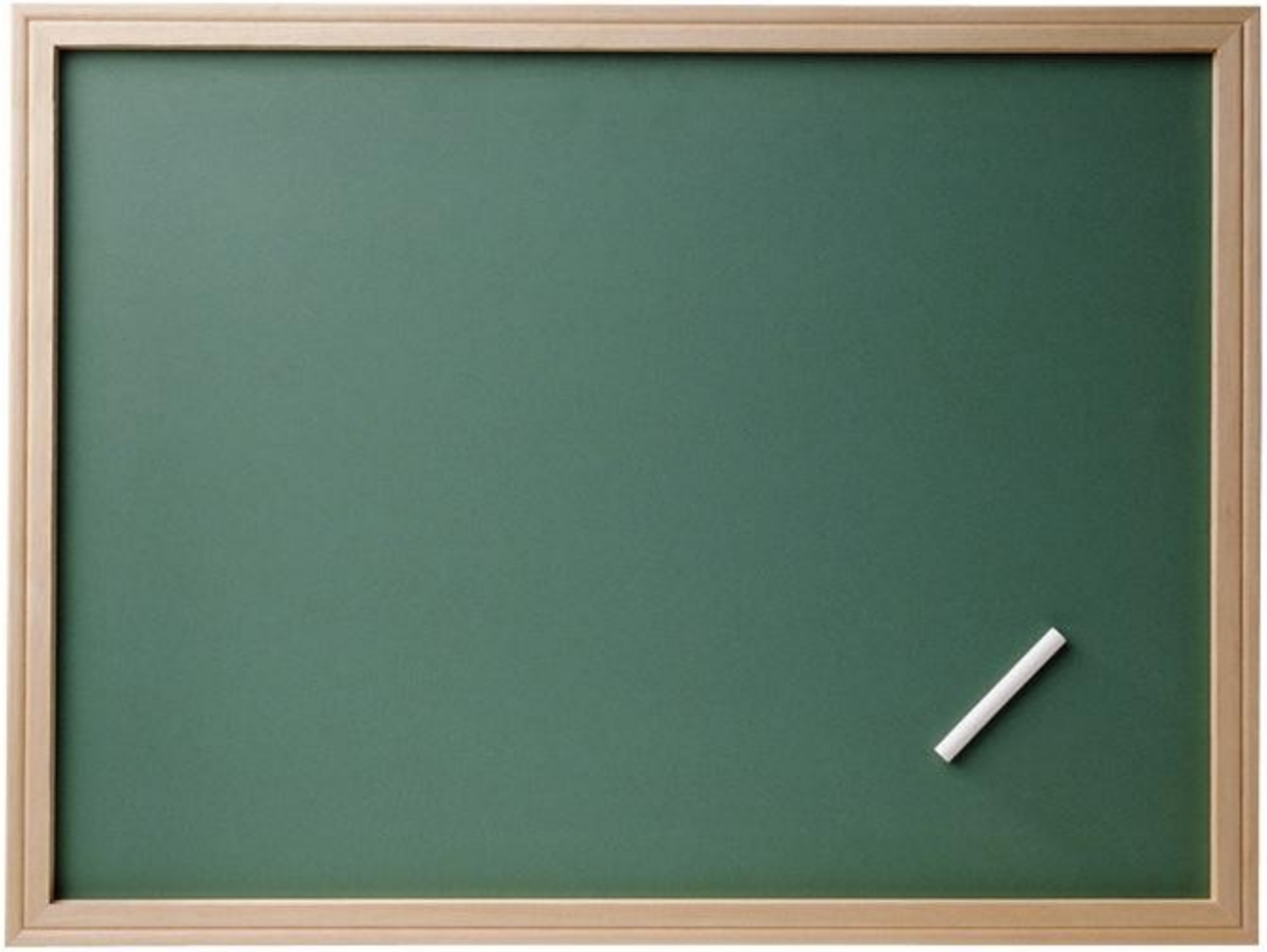
$$z = \sum_x 2^{-\ell(x)}$$

Difference measured by the relative entropy "distance"
(Kullback-Leibler divergence)

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \geq 0$$

Gibbs inequality

Proof of the lower bound $H(X) \leq L(C, X)$



Summary of remarks on the lower bound

- When is the lower bound achieved?

$$H(X) = L(C, X)$$

  - Optimal source codelength equal to information content

$$l(x) = \log \frac{1}{P_X(x)}$$

  - Complete code $z = 1$

- Conversely, any given codelengths would be optimal for the distribution

$$Q_X(x) = \frac{2^{-l(x)}}{z} \qquad z = \sum_x 2^{-l(x)}$$

What if we use a complete code but different codelengths?

What if we use a complete code but different codelengths?

Average codeword length becomes larger

$$L(C,X) = H(X) + \sum_x P_X(x) \log \frac{P_X(x)}{\tilde{Q}_X(x)} \qquad \tilde{Q}_X(x) = 2^{-l(x)}$$

What if we use a complete code but different codelengths?

Average codeword length becomes larger

$$L(C,X) = H(X) + \sum_x P_X(x) \log \frac{P_X(x)}{\tilde{Q}_X(x)} \qquad \tilde{Q}_X(x) = 2^{-l(x)}$$

Term is positive and equal to the so-called relative entropy

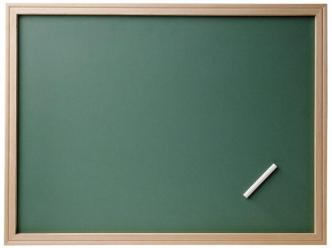
$$D(P||\tilde{Q})$$

How much can we compress?

- There exists a compression scheme such that

$$L(C, X) \leq H(X) + 1$$

- Proof using the Kraft inequality



What we'll do today

- Lossless data compression
  - Basic concepts
  - Symbol codes
  - Kraft's inequality
  - Lossless compression and entropy
  - Shannon's source coding theorem for symbol codes
- An optimal symbol code: Huffman coding
- Limits of symbol codes

Goal

- We want an explicit coding scheme such that:

$$H(X) \leq L(C, X) < H(X) + 1$$

- Ideally, *no other symbol code has lower expected codelength*

- We call such a scheme *optimal*

- Any ideas?

Some clues

- We know that we only need to think about prefix codes
- … and these will automatically be uniquely decodable
- Any prefix code is a tree.
- So how can we make such a tree?

Attempt #1: Top down (greedy)

- Example

$$\mathcal{A}_X = \{a, b, c, d, e, f, g\}$$

$$P_X = \{0.01, 0.24, 0.05, 0.20, 0.47, 0.01, 0.02\}$$

$H(X) = 1.93$

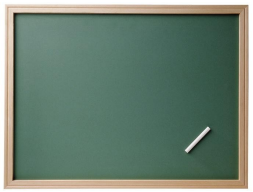
Attempt #1: Top down (greedy)

- Example

$$A_X = \{a, b, c, d, e, f, g\}$$

$$P_X = \{0.01, 0.24, 0.05, 0.20, 0.47, 0.01, 0.02\}$$

$H(X) = 1.93$



$$L(C, X) = 2.53$$

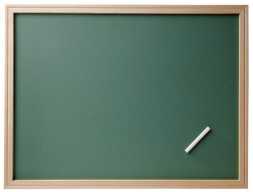
Attempt #1: Top down (greedy)

- Example

$$\mathcal{A}_X = \{a, b, c, d, e, f, g\}$$

$$P_X = \{0.01, 0.24, 0.05, 0.20, 0.47, 0.01, 0.02\}$$

$H(X) = 1.93$



Attempt #2: Bottom up (huffman)

Attempt #2: Bottom up (Huffman)

- Take the two least probable symbols
  - These will get the longest codewords
  - Of equal length, differing in the last digit only

Attempt #2: Bottom up (huffman)

- Take the two least probable symbols
  - These will get the longest codewords
  - Of equal length, differing in the last digit only
- Combine them into a single symbol

Attempt #2: Bottom up (huffman)

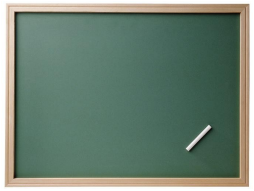
- Take the two least probable symbols
  - These will get the longest codewords
  - Of equal length, differing in the last digit only
- Combine them into a single symbol

Repeat

Attempt #2: Bottom up (Huffman)

- Take the two least probable symbols
  - These will get the longest codewords
  - Of equal length, differing in the last digit only
- Combine them into a single symbol

Repeat

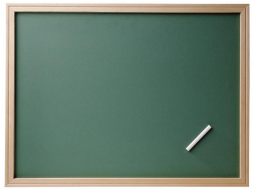


$$L(C, X) = 1.97$$

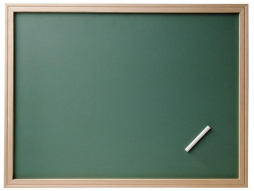
Attempt #2: Bottom up (Huffman)

- Take the two least probable symbols
  - These will get the longest codewords
  - Of equal length, differing in the last digit only
- Combine them into a single symbol

Repeat

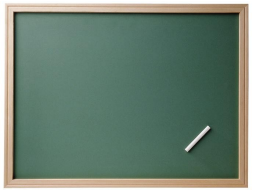


HUFFMAN CODING EXAMPLE



$\mathcal{A}_X$	a	b	c	d	e
P_X	0.25	0.25	0.2	0.15	0.15

HUFFMAN CODING EXAMPLE



$\mathcal{A}_X$	a	b	c	d	e
P_X	0.25	0.25	0.2	0.15	0.15

x	$P_X(x)$	$h(p_X(x))$	$l(x)$	$c(x)$
a	0.25	2	2	00
b	0.25	2	2	10
c	0.2	2.3	2	11
d	0.15	2.7	3	010
e	0.15	2.7	3	011

Performance of Huffman coding

- Can be proven to be optimal! No other better symbol code
- Does this mean it is the solution to all our compression problems?

Disadvantages

- Does not make use of context!
  - E.g., when compressing English language

Disadvantages

- Does not make use of context!
  - E.g., when compressing English language

That is, does not adapt to changing ensemble

Disadvantages

- Does not make use of context!
  - E.g., when compressing English language

  That is, does not adapt to changing ensemble

- Single bit can be significant when the entropy is small

$$H(X) \leq L(C, X) < H(X) + 1$$

  Can be dealt with to some extent by compressing blocks at once

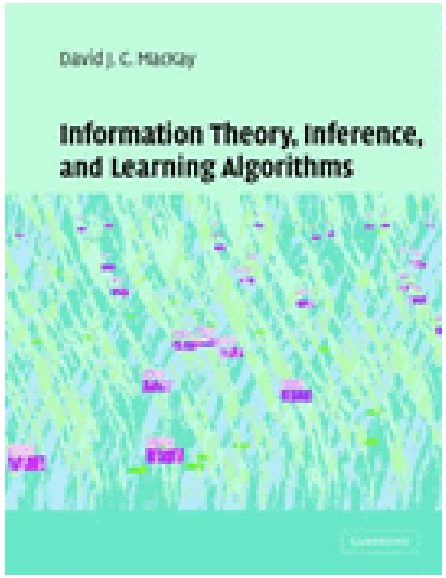
Let's summarize

- Basics of lossless compression and symbol codes
- Prefix codes
- Kraft's inequality
- Optimal expected code length determined by the Shannon entropy
- Optimal Huffman code
- Limits of symbol codes: no context, at least one bit per encoded symbol

- Next time: Stream codes

Reading for this lecture

- Chapter 5 in the book



Information Theory, Inference and Learning Algorithms
by David J. C. MacKay
Cambridge University Press, 2003

- Homework due by Monday 2pm next week, upload to IVLE Workbin