Computer Science Fundamentals

Information & Source Coding – Basics

Technische Hochschule Rosenheim
Winter 2021/22
Prof. Dr. Jochen Schmidt

Overview

- Intermission: Probability theory crash course
- Statistical information content of a message
- Coding basics & examples for simple codes

Probability Theory Basics

Random Experiment

Technische Hochschule Rosenheim

- Process or trial that is subject to chance, or where we cannot predict the outcome for other reasons
- Quantitative statements are possible using mathematical methods of statistics
  - Repetition of the experiment under constant conditions
- The result of an experiment is referred to as outcome (*Elementarereignis*)
  - outcomes are mutually exclusive
- The set of all possible outcomes is the sample space (*Ergebnisraum*)
- An event (*Ereignis*) is the result of a random experiment, i.e., a subset of the sample space

Random Experiment – Examples

- Experiment „tossing of a coin "
  - sample space: { heads, tails }
- Experiment „rolling of a six-sided die"
  - sample space: { 1, 2, 3, 4, 5, 6 }
- Experiment „Measuring the service life of a light bulb"
  - Infinite (real valued) sample space

Relative Frequency

Technische Hochschule Rosenheim

Relative frequency h

- Quotient of the number of events that have a particular characteristic and the total number of events examined for that characteristic

$$h = \frac{\text{Number of events that have a desired characteristic}}{\text{Total number of events}}$$

- $0 \leq h \leq 1$ always holds by definition

Relative Frequency

How to determine relative frequency? By many repetitions of the random experiment.

Examples

- Tossing a coin

  The more often you throw a coin, the less h_{heads} and h_{tails} will differ from 1/2.

- Rolling dice

  The more often you throw a die, the less h_1, h_2, h_3, h_4, h_5 and h_6 will differ from 1/6.

Probability

Relationship between probability and relative frequency: Law of large numbers

$$p(A) = \lim_{n \to \infty} h(A)$$

Probability that event A will occur

Relative frequency of event A

Limit value of the sequence

A: Event under consideration
n: Number of trials

Probability

Exact mathematical definition of probability by the Kolmogorov Axiom System

- Axiom 1: The probability p(A) for the occurrence of a particular event A is a real function that can take all values between zero and one:

$$0 \leq p(A) \leq 1$$

- Axiom 2: The probability of the occurrence of an event A that is certain to occur is 1:

$$p(A) = 1$$

- Axiom 3: For mutually exclusive events A and B the addition theorem holds:

$$p(A \text{ or } B) = p(A) + p(B)$$
$$p(A \cup B) = p(A) + p(B)$$

Probability

Some conclusions from the axioms:

- Probability of an event A that is certain not to occur: $p(A) = 0$

- Probability that event A will not occur: $p(\bar{A}) = 1 - p(A)$ $\bar{A}$ = „not A“

- Probability that two events A and B will occur together:

$$p(A \text{ and } B) = p(A) \cdot p(B)$$
$$p(A \cap B) = p(A) \cdot p(B)$$

Condition:
Events A and B are not mutually exclusive and are independent of each other

Examples

- Roll two **distinguishable dice** (red and green) simultaneously
  - Probability of green die showing 1 and red die showing 2?
    p(A and B) = 1/6 · 1/6 = 1/36

- Roll a **single die** twice
  - Probability to roll a 1 in the first throw and then a 2 in the second one?
    p(A and B) = 1/6 · 1/6 = 1/36

- Roll two **indistinguishable dice** simultaneously
  - Probability of 1 and 2 being rolled?
    p(E) = (1/6 + 1/6) · 1/6 = 1/18

Remarks

- The previous slides include only what is absolutely necessary for this course
- Many important terms were not explained (e.g., conditional probabilities)
- This will follow in a separate course

Statistical Information Content

Messages & Alphabets

- Messages are formed as strings from single symbols
- The set of all available symbols is called an alphabet
- An alphabet consists of
  - a **countable set of symbols** (*abzählbare Menge*), typically a finite set, and
  - an **order relation** (*Ordnungsrelation*)
    (a rule that defines the order of the symbols)

- Examples:
  - {a, b, c, ..., z}
    Set of all lowercase letters in lexicographic order
  - {0, 1, 2, ..., 9}
    finite set of integers 0 to 9 with the order relation „<“
  - {2, 4, 6, … }
    infinite set of even natural numbers with the order relation „<“
  - {0, 1}
    Binary digits 0 and 1 with $0 < 1$

Messages & Alphabets

Let A be an alphabet, e.g., A = {a, b}

- A^i, i = 0, 1, 2, … is the set of all strings of length i that can be formed using A
  - A^0 is the message of length 0, i.e., “nothing”, the so-called empty string, denoted as ε or λ
  - e.g., $A^1 = A = \{a, b\}$, $A^2 = \{aa, ab, ba, bb\}$, $A^3 = \{aaa, aab, aba, abb, baa, \ldots\}$

- $A^+ = A^1 \cup A^2 \cup A^3 \cup \ldots$ is the set of all messages of any length, excluding length 0 (called Kleene plus, *positive Hülle*)
  - e.g., $A^+ = \{a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \ldots\}$

- $A^* = A^0 \cup A^1 \cup A^2 \cup \ldots$ is the set of all messages of any length, including length 0 (called Kleene closure, *Kleenesche Hülle*)
  - e.g., $A^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \ldots\}$

- (A, ∘) forms a semigroup (*Halbgruppe*) – more on alphabets in Theoretical Computer Science
  - ∘ = string concatenation being the operator
  - the empty string the neutral element

Information Content of a Message

Consideration of the term information from a specific point of view

- Mathematical definition of information
- Hence
  - not the semantic meaning of information
  - not oriented to the purpose pursued with a message
  - Meaning: Two messages (one with special content – one with "nonsense") can contain exactly the same amount of information.
- Shannon‘s information theory
  - Mainly developed by Claude Shannon until 1950
- Objective
  - Mathematical description of the statistical information content $I(x)$
  - of a symbol or word x,
  - having a probability of occurrence $p(x)$

Information

Requirements for a mathematical description

1. The less frequently a certain symbol x occurs, i.e., the smaller $p(x)$, the greater the information content of this symbol should be

$$I(x) \sim \frac{1}{p(x)}$$

2. Total information of a string, e.g., $x_1 x_2 x_3$ should result from the sum of the individual information

$$I(x_1 x_2 x_3) = I(x_1) + I(x_2) + I(x_3)$$

3. For the information content of a symbol x that occurs with certainty, i.e., in case $p(x) = 1$, we want zero information:

$$I(x) = 0$$

$\longrightarrow$ Logarithm function meets these requirements

$$I(x) = \log_b \frac{1}{p(x)}$$

b = scale for measuring information

Information

Definition: Use binary (two states, 0 and 1) $\longrightarrow b = 2$

$$I(x) = \operatorname{ld}\frac{1}{p(x)} = -\operatorname{ld} p(x) \quad [Bit]$$

binary logarithm

- Number of elementary decisions that are necessary to uniquely identify a message symbol by symbol
- Unit of measurement: Bit
- Information content of a symbol =
  Number of binary digits that must be used for a unique binary representation of the symbol

Information – Example

Calculation of information content for non-binary messages

- Given: Letter b occurs in a German-language text with a probability of 0.016
- Wanted: Information content of this letter

- Solution:

$$I(b) = \mathrm{ld}\frac{1}{0.016} = \frac{\log(\frac{1}{0.016})}{\log(2)} \approx \frac{1.79588}{0.30103} \approx 5.97\ [Bit]$$

Reminder – Base Change with Logarithms

- Base change

$$\log_b(x) = \frac{\log_{10}(x)}{\log_{10}(b)}, \qquad \text{i.e. } \log_2(x) = \mathrm{ld}(x) = \frac{\log_{10}(x)}{\log_{10}(2)}$$

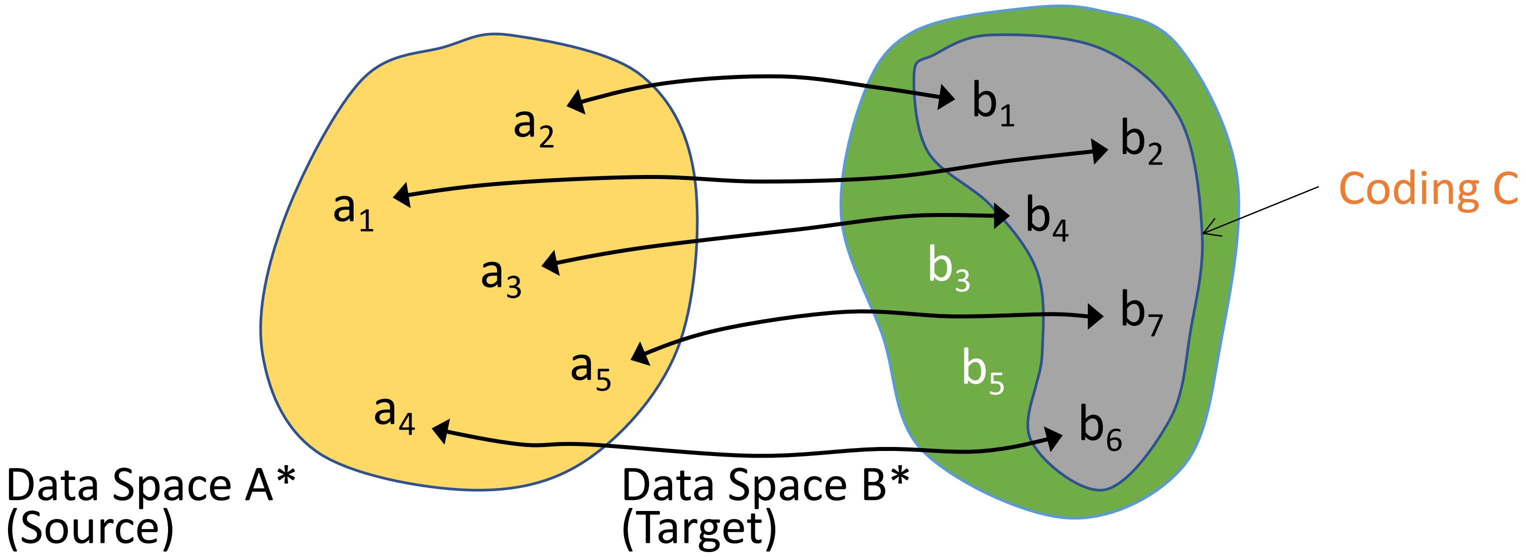
  - using $\log_{10}(2) = \log(2) = 0.30103$
  - … or use any other base than 10

- Notations:
  - $\log_{10}(x)$ is $\log(x)$
  - $\log_2(x)$ is $\mathrm{ld}(x)$
  - $\log_e(x)$ is $\ln(x)$ with $e \approx 2.71828...$

Coding Basics

Coding – Definitions

- Objective: Problem-specific representation of a data during storage and transmission



- Coding C
  - an injective mapping from A* to B*
  - Note: $C \subseteq B^*$ holds, i.e., C is a subset of B*

- Binary code C
  - Target set is a data space B* over the alphabet {0, 1}

Block Codes & Variable Length Codes

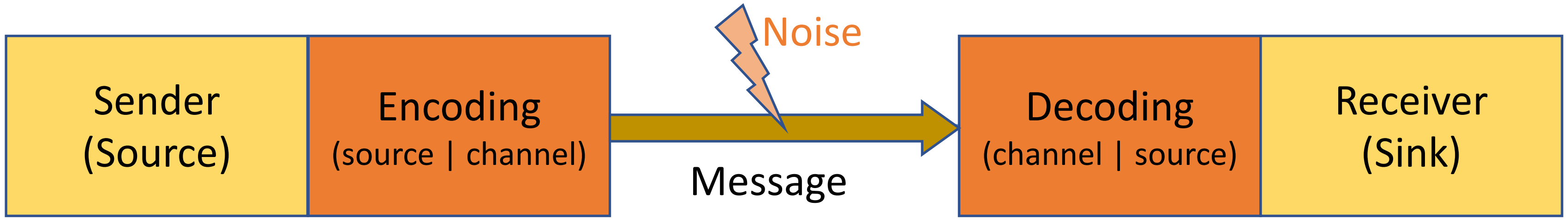
- Fixed word length encoding (block codes)
  - all encoded symbols have a constant code word length

- Variable word length encoding
  - Frequently occurring symbols receive short code words
  - Infrequently occurring symbols receive long code words

- First technical code with variable length: Morse Code
  - no binary encoding, as it contains three symbols: dot, dash, and space (a pause)

Coding – Definitions

- Schematic representation of the encoding and transmission of messages



- Expectations of a "good" coding
  - representation of the data with as few symbols as possible (source coding, *Quellencodierung*)
  - as insensitive as possible to noise (channel coding, *Kanalcodierung*)
  - code should be easy to process in a computer system

Entropy of Data Source

- Message consists of
  - symbols or symbols connected to strings,
  - each carrying different information content, as they occur with different frequency

- Definition of the term
  - average information content of all symbols of a data source
  - = Entropy *H (Entropie)* of a data source, which has symbols $x_1, x_2, \ldots, x_n$ of some alphabet A available.

- Sum of the information content of the characters weighted with the probabilities of occurrence

$$H = \sum_{i=1}^{n} p(x_i) \cdot \mathrm{ld}\frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \cdot \mathrm{ld}\, p(x) = \sum_{i=1}^{n} p(x_i) \cdot I(x_i)$$

  - The highest average information content is obtained when all characters occur with the same probability
  - Unit: bits/symbol

Source Redundancy R_S (*Quellen-Redundanz*)

- Difference between maximum possible entropy H_0 and actual entropy H of a source

$$R_S = H_0 - H \quad \text{(bits/symbol)}$$

- we get the maximum possible entropy H_0, if all symbols of the alphabet A are equally likely. With |A| = n:

$$H_0 = \sum_{i=1}^{n} p(x_i) \cdot \operatorname{ld} \frac{1}{p(x_i)} = \sum_{i=1}^{n} \frac{1}{n} \cdot \operatorname{ld} n = \frac{1}{n} \sum_{i=1}^{n} \operatorname{ld} n = \frac{1}{n} n \operatorname{ld} n = \operatorname{ld} n$$

- H_0 is independent of the code used

Entropy – Surprisal

- Other interpretation of the concept of entropy to compare data sources:
  - The smaller the entropy, the greater the certainty with which we can predict the occurrence of a symbol.
  - The higher the entropy of a data source, the greater its uncertainty (surprisal).

- Example:

  Two data sources
  - $A_1 = \{a, b, c, d\}$ with the probabilities of occurrence $p(a) = {}^{11}/_{16},\ \ p(b) = p(c) = {}^{1}/_{8},\ \ p(d) = {}^{1}/_{16}$
  - $A_2 = \{+, -, *\}$ with the probabilities of occurrence $p(+) = {}^{1}/_{6},\ \ p(-) = {}^{1}/_{2},\ \ p(*) = {}^{1}/_{3}$

  Entropies:
  - $H_1 = \frac{11}{16} \cdot \operatorname{ld} \frac{16}{11} + \frac{1}{8} \cdot \operatorname{ld} 8 + \frac{1}{8} \cdot \operatorname{ld} 8 + \frac{1}{16} \cdot \operatorname{ld} 16 \approx 1.372 \left[\frac{\text{bits}}{\text{symbol}}\right]$
  - $H_2 = \frac{1}{6} \cdot \operatorname{ld} 6 + \frac{1}{2} \cdot \operatorname{ld} 2 + \frac{1}{3} \cdot \operatorname{ld} 3 \approx 1.460 \left[\frac{\text{bits}}{\text{symbol}}\right]$
  - Uncertainty for A_2 is greater than for A_1, as H_2 is greater than H_1

Average Length of a Code

or: expected length of code L

- important property of a code
- defined as

$$L = \sum_{i=1}^{n} p_i \cdot l_i$$

where

  - n: number of code words
  - p_i probability of occurrence for code word i
  - l_i length of i-th code word

- this is a weighted average over all code word lengths

Shannon's Source Coding Theorem

- For any encoding of a data source

$$H \leq L$$

- Entropy H is a lower bound for the average word length L of an optimal code
  - for lower L information will be lost
- If all probabilities of occurrence are the same: $H = L$
- Proof by Claude Shannon 1948

- Note:
  - this applies to lossless coding
  - in the case of lossy encoding, the mapping of source to destination message space is no longer reversible and unique anyway.

Code Redundancy R (*Code-Redundanz*)

- Difference between average word length L and entropy H

$$R = L - H \qquad \text{(bits/symbol)}$$

- Indicates the proportion of a message that does not carry any information in the statistical sense
  - “wasted” part of a code
- Desirable: Codes with low redundancy
  - Reduced memory requirements and faster message transfer
  - $\longrightarrow$ source coding
- However, redundancy can increase robustness to noise
  - error detection/correction for data transmission/storage
  - $\longrightarrow$ channel coding

Example

- $A_1 = \{a, b, c, d\}$ with $p(a) = {}^{11}/_{16}, \;\; p(b) = p(c) = {}^{1}/_{8}, \; p(d) = {}^{1}/_{16}$
  - The entropy was $H = 1.37$ bits/symbol
- We would get the maximum entropy for a uniform distribution, i.e., all probabilities 0.25
  - $H_0 = \mathrm{ld}\, n = \mathrm{ld}\, 4 = 2$ bits/symbol
- Consider the following codes:

Symbol	Code
a	00
b	01
c	10
d	11

average code word length (surprise…it's 2)
$L = 2 \cdot {}^{11}/_{16} + 2 \cdot {}^{1}/_{8} + 2 \cdot {}^{1}/_{8} + 2 \cdot {}^{1}/_{16}$
$= 2$ bits/symbol

Redundancy
$R = L - H = 2 - 1.37$
$= 0.63$ bits/symbol

Symbol	Code
a	0
b	10
c	110
d	111

average code word length
$L = 1 \cdot {}^{11}/_{16} + 2 \cdot {}^{1}/_{8} + 3 \cdot {}^{1}/_{8} + 3 \cdot {}^{1}/_{16}$
$= 1.5$ bits/symbol

Redundancy
$R = L - H = 1.5 - 1.37$
$= 0.13$ bits/symbol

ASCII

- ASCII = American Standard Code for Information Interchange

- is a defined mapping (standard) for binary encoding of characters
  - first published in 1963
  - most common character encoding in the WWW until 2007; UTF-8 now
- includes
  - lowercase/uppercase letters of the Latin alphabet
  - Arabic numerals
  - and many special characters
- original ASCII: 7 Bit block code
  - 8^{th} bit could be used for parity ($\longrightarrow$ channel coding)
  - Different, standardized, ASCII code extensions available
    - these use the 8^{th} bit to encode an additional set of 128 characters

ASCII (8 Bits) – Subset

Technische Hochschule Rosenheim

Decimal	Octal	Hexa	Binary	Character
32	40	20	00100000	(space)
33	41	21	00100001	!
34	42	22	00100010	"
35	43	23	00100011	#
36	44	24	00100100	$
37	45	25	00100101	%
38	46	26	00100110	&
39	47	27	00100111	'
40	50	28	00101000	(
41	51	29	00101001	)
42	52	2A	00101010	*
43	53	2B	00101011	+
44	54	2C	00101100	,
45	55	2D	00101101	-
46	56	2E	00101110	.
47	57	2F	00101111	/
48	60	30	00110000	0
49	61	31	00110001	1
50	62	32	00110010	2
51	63	33	00110011	3
52	64	34	00110100	4
53	65	35	00110101	5
54	66	36	00110110	6
55	67	37	00110111	7
56	70	38	00111000	8
57	71	39	00111001	9
58	72	3A	00111010	:
59	73	3B	00111011	;
60	74	3C	00111100	<
61	75	3D	00111101	=
62	76	3E	00111110	>
63	77	3F	00111111	?

Decimal	Octal	Hexa	Binary	Character
64	100	40	01000000	@
65	101	41	01000001	A
66	102	42	01000010	B
67	103	43	01000011	C
68	104	44	01000100	D
69	105	45	01000101	E
70	106	46	01000110	F
71	107	47	01000111	G
72	110	48	01001000	H
73	111	49	01001001	I
74	112	4A	01001010	J
75	113	4B	01001011	K
76	114	4C	01001100	L
77	115	4D	01001101	M
78	116	4E	01001110	N
79	117	4F	01001111	O
80	120	50	01010000	P
81	121	51	01010001	Q
82	122	52	01010010	R
83	123	53	01010011	S
84	124	54	01010100	T
85	125	55	01010101	U
86	126	56	01010110	V
87	127	57	01010111	W
88	130	58	01011000	X
89	131	59	01011001	Y
90	132	5A	01011010	Z
91	133	5B	01011011	[
92	134	5C	01011100	\
93	135	5D	01011101	]
94	136	5E	01011110	^
95	137	5F	01011111	_

Decimal	Octal	Hexa	Binary	Character
96	140	60	01100000	`
97	141	61	01100001	a
98	142	62	01100010	b
99	143	63	01100011	c
100	144	64	01100100	d
101	145	65	01100101	e
102	146	66	01100110	f
103	147	67	01100111	g
104	150	68	01101000	h
105	151	69	01101001	i
106	152	6A	01101010	j
107	153	6B	01101011	k
108	154	6C	01101100	l
109	155	6D	01101101	m
110	156	6E	01101110	n
111	157	6F	01101111	o
112	160	70	01110000	p
113	161	71	01110001	q
114	162	72	01110010	r
115	163	73	01110011	s
116	164	74	01110100	t
117	165	75	01110101	u
118	166	76	01110110	v
119	167	77	01110111	w
120	170	78	01111000	x
121	171	79	01111001	y
122	172	7A	01111010	z
123	173	7B	01111011	{
124	174	7C	01111100	|
125	175	7D	01111101	}
126	176	7E	01111110	~
127	177	7F	01111111	(del.)

Average word length: 8 bits

Entropy? Redundancy?

When would this be the optimal code to use?

Unicode

Technische Hochschule Rosenheim

- 1991: Foundation of the Unicode Consortium
- originally, Unicode characters were represented using a two-byte system
  - up to 65,536 different characters can be represented
  - BMP (Basic Multilingual Plane) = 2-Byte-System
- Unicode version 3.0 (1999): contained already 49,194 characters
  - extension to 4-byte system required…
- current version (2020): contains 143,859 characters
- different encoding formats available

Unicode

Some Unicode encoding formats (UTF = Unicode Transformation Format)

- UTF-32: uses always 32 bit/4 bytes (not commonly used)
- UTF-16: uses 2 or 4 bytes
  - depending on the character to be encoded – common characters require 2 bytes
  - used, e.g., in Windows and Java
- UTF-8: uses 1 to 4 bytes
  - 1 byte (characters 1-127) = 7-bit ASCII
  - 8^{th} bit controls the extension to 2, 3, 4 byte encoding
  - efficient for Latin characters (1 byte)
  - most other characters require 3 bytes
  - used, e.g., WWW, Unix E-Mail

BCD – Binary Coded Decimals

- a method for encoding numbers
- a fixed number of bits is used for each decimal digit
  - usually four, sometimes eight, bits
- each decimal digit is converted independently to a binary block
  - just as we did for conversion hexadecimal to binary

Decimal	Standard Dual	BCD
294	100100110	0010.1001.0100 2 9 4
16289	11111110100001	0001.0110.0010.1000.1001 1 6 2 8 9

BCD

- Bit patterns 1010, 1011, ..., 1111 are not used in BCD code (only 10 decimal digits exist)
- Often used for other purposes, e.g.,
  - 1010 for sign +
  - 1011 for sign –
- Advantages
  - Exact representation of fractions possible (e.g., 0.1_{10}) – no accuracy loss caused by conversion
  - conversion from memory representation to display is simple
  - simple scaling by factors of 10
- Disadvantages
  - ineffective (memory wasting) compared to standard binary
  - computationally more expensive (e.g., adding is more complex)
- Used for special-purpose applications
  - business applications that require exact number conversions
  - LCDs/LEDs displaying numbers
  - Storing telephone numbers

Kullback-Leibler Divergence (*KL-Divergenz*)

- Also: relative entropy, information gain
- Measures how many bits are wasted on average
  - when using (an incorrect) probability distribution q for coding,
  - rather than the real distribution of the code words p?
- Or: how much a probability distribution q differs from the actual distribution p
- For discrete distributions using a finite alphabet with n symbols:

$$\mathrm{KL}(p,q) = \sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,\frac{p(x_i)}{q(x_i)}$$

= expected value of $\mathrm{ld}\,\frac{p(x)}{q(x)}$ using the density $p(x)$ for averaging

$$= \underbrace{\sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,p(x_i)}_{\text{–Entropy } H(p)} - \underbrace{\sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,q(x_i)}_{\text{Cross Entropy } H(p,q)}$$

Note: $\mathrm{KL}(p,q) \neq \mathrm{KL}(q,p)$

Cross Entropy (*Kreuzentropie*)

Technische Hochschule **Rosenheim**

- Measures how many bits are needed on average if
  - we use (an incorrect) probability distribution q for coding
  - rather than the real distribution of the code words p
- For discrete distributions using a finite alphabet with n symbols:

$$H(p,q) = H(p) + KL(p,q) = -\sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,q(x_i)$$

- Can also be used for comparing two distributions
- KL/Cross entropy are also widely applied in machine learning for training classifiers like random forests or neural networks

Example

Technische Hochschule **Rosenheim**

- $\text{A}_1 = \{a, b, c, d\}$ with $p(a) = {}^{11}/_{16},\ \ p(b) = p(c) = {}^{1}/_{8},\ p(d) = {}^{1}/_{16}$
  - This is the real distribution
  - The entropy was $H(p) = 1.37$ bits/symbol
- What if we use a uniform distribution q, i.e., $p(a) = p(b) = p(c) = p(d) = {}^{1}/_{4}$ instead of the actual one?
- Cross entropy:
  $H(p, q) = -\sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,q(x_i) = -\left({}^{11}/_{16}\,\mathrm{ld}\,\frac{1}{4} + {}^{1}/_{8}\,\mathrm{ld}\,\frac{1}{4} + {}^{1}/_{8}\,\mathrm{ld}\,\frac{1}{4} + {}^{1}/_{16}\,\mathrm{ld}\,\frac{1}{4}\right) = 2$ bits/symbol
- KL divergence:
  $\mathrm{KL}(p, q) = \sum_{i=1}^{n} p(x_i)\,\mathrm{ld}\,\frac{p(x_i)}{q(x_i)} = \frac{11}{16}\mathrm{ld}\,\frac{{}^{11}/_{16}}{{}^{1}/_{4}} + \frac{1}{8}\mathrm{ld}\,\frac{{}^{1}/_{8}}{{}^{1}/_{4}} + \frac{1}{8}\mathrm{ld}\,\frac{{}^{1}/_{8}}{{}^{1}/_{4}} + \frac{1}{16}\mathrm{ld}\,\frac{{}^{1}/_{16}}{{}^{1}/_{4}} = 0.63$ bits/symbol
- Note: These measures do not depend on a particular code (as opposed to redundancy)

Example – Neural Network Training

Training = Determine weights

- define loss function L of output neurons – you'll often find cross entropy here
  - p: real distribution, defined by training data
  - q: current distribution, defined by network output
  - we basically compare p to q
- minimize loss