

Information Theory and Coding



Radim BELOHLAVEK

Dept. Computer Science
Palacky University, Olomouc
Czech Republic

Content

- **Information Theory**

See my slides Introduction to information theory and its applications.

- **Coding**

Many texts exist, I recommend:

- Adámek J.: Foundations of Coding. J. Wiley, 1991.
- Adámek J.: Kódování. SNTL, Praha, 1989.
- Ash R. B.: Information Theory. Dover, New York, 1990.

Introduction

- Aims of coding.
- What we cover.
- What we do not cover.

BASIC CONCEPTS OF CODING

Alphabet, words

- Alphabet: A non-empty finite set A . $a \in A$ are called characters or symbols.
- Word (or string) over alphabet A : A finite sequence of symbols from A . The empty word (no symbols) is denoted by ε .
- Word length: The number of symbols in a word. Length of word w is denoted by $|w|$. Example: $|010| = 3$, $|1| = 1$, $|\varepsilon| = 0$.
- Concatenation of words: If $u = a_1 \cdots a_m$, $v = b_1 \cdots b_n$, then the word $uv = a_1 \cdots a_m b_1 \cdots b_n$ is called the concatenation of u and v .
- Prefix, suffix: If $w = uv$ for some words u, v, w , u is called a prefix of w , v is called a suffix of w .
- Formal language over A : A subset of the set of all words over A .

Problem of coding, basic concepts

Problem of coding

- At the input are messages. Messages are sequences of symbols of the source alphabet $\{x_1, \dots, x_M\}$.
- Messages are to be sent over a communication channel.
- Before entering the communication channel, messages are encoded. At the output of the channel, the encoded message is decoded.
- Encoding is accomplished using symbols from a code alphabet $\{a_1, \dots, a_D\}$. Every symbol x_i is replaced by a word over the code alphabet. These words are called code words. Encoding must be devised in such a way that a reasonable decoding at the channel output is possible.
- Decoding consists in reconstructing from the symbols of the code alphabet received at the output the symbols x_i that were encoded and sent through the channel.

- Two assumptions about communication channel: Noiseless vs. noisy.
- Noiseless channel: Every symbol a_i sent through the channel is received at the output. There are no errors.
- Noisy channel: It may happen that if a_i is sent through the channel, a different symbol is received at the output. Errors may occur.

Coding, code word, code

Definition

A **coding** over a source alphabet A and code alphabet B is any injective mapping C of set A into the set B^+ of all non-empty words over B . Words $C(x)$ for $x \in A$ are called **code words**. The set $\{C(x) \mid x \in A\}$ of all code words is called a **code**.

Clearly, a coding $C : A \rightarrow B^+$ may be extended to to a coding of source words $C^* : A^+ \rightarrow B^+$ by $C^*(x_1 \cdots x_n) = C(x_1) \cdots C(x_n)$ (concatenation) of $C(x_i)$ s. Because $C^*(x) = C(x)$ for any $x \in A$, we denote C^* simply by C .

Often, we do not distinguish between coding and code (if the source symbols do not matter). We say “code C ”, etc.

Uniquely decipherable, block, and prefix codes

Definition

A code C (over A and B) is called

- **uniquely decipherable** (UD) if C^* is injective, i.e. every word over B can be broken into codewords in at most one way.
- **prefix code** (also prefix-free code, instantaneous code) if no code word is a prefix of a different code word.
- **block code** if all code words have the same length.
- **binary code** if the code alphabet has two symbols (usually denoted 0 and 1).

Clearly, every prefix code is uniquely decipherable; every block code is a prefix code. Every block code is UD.

Good feature of prefix codes: Decoding may be done by reading a sequence of code symbols symbol by symbol, from left to right.

Example Let $A = \{x_1, x_2, x_3, x_4\}$, $B = \{0, 1\}$, consider mappings C_i .

	x_1	x_2	x_3	x_4
C_1	0	1	01	0
C_2	0	010	01	10
C_3	0	100	101	11
C_4	0	01	011	0111
C_5	00	01	10	11

(meaning: $C_3(x_3) = 101$, etc.)

C_1 is not a coding because $C_1(x_1) = 0 = C_1(x_4)$ (C_1 is not injective, two different symbols have the same code words).

C_2 is a coding which is not UD. Namely, $C^*(x_1x_4x_3) = 01001 = C^*(x_2x_3)$, i.e. C^* is not injective. Put differently, sequence 01001 may be broken into code words two ways: 0|10|01 and 010|01.

C_3 is a prefix code (thus also UD), but is not a block code because $|0| = 1$ while $|100| = 3$ (two code words with different lengths).

C_4 is UD (why?) but not a prefix code (code word 0 is a prefix of all the other code words) (and thus not a block code).

C_5 is a block code of length 2.

Test of unique decipherability

Construct a sequence $S_0, S_1, \dots, S_i, \dots$ of sets code words for a given C :

- S_0 = set of code words of C
- construction of S_i (from S_0 and S_{i-1}):

$$S_i = \{v \mid [u \in S_0 \text{ and } uv \in S_{i-1}] \text{ or } [u \in S_{i-1} \text{ and } uv \in S_0]\}.$$

Theorem

A code is UD iff none of $S_1, S_2, S_3, \dots$ contains a code word.

Idea: Word t over code alphabet B is called a tail if $u_1 \cdots u_m t = v_1 \cdots v_m$ for some code words u_i, v_j , with $u_1 \neq v_1$. One can see that a code is not UD if and only if there exists a tail t which is a code word.

The above procedure looks for tails.

Several efficient algorithms were proposed. E.g.

Even S.: Tests for unique decipherability. IEEE Trans. Information Theory 9(1963), 109–112.

A procedure based on the theorem: Clearly, we need not include t in S_i if it was already included in some S_j for $j < i$. Creating S_i this way, i.e. taking $S'_i = S_i - \bigcup_{j < i} S_j$ and obtaining S_{i+1} from S_0 and S'_i , the procedure stops (there is only a finite number of words that can be added to S_i s).

Example Consider again

	x_1	x_2	x_3	x_4
C_2	0	010	01	10
C_4	0	01	011	0111

Procedure for C_2 :

- $S_0 = \{0, 010, 01, 10\}$
- $S_1 = \{10, 1, 0\}$, stop because S_1 contains a code word 0

Thus C_2 is not UD.

Procedure for C_4 :

- $S_0 = \{0, 01, 011, 0111\}$
- $S_1 = \{1, 11, 111\}$, $S'_1 = S_1$
- $S_2 = \emptyset$, stop because S_2 is empty

Neither S_1 nor S_2 contains a code word. Thus C_4 is UD.

NOISELESS CODING AND OPTIMAL CODES

Problem setting

- We assume noiseless channel.
- Symbols $x_1, \dots, x_M$ of source alphabet occur with probabilities $p_1, \dots, p_M$ (i.e., $0 \leq p_i \leq 1$ and $\sum_{i=1}^M p_i = 1$).
- How to construct a UD code for which the encoded messages are short?
- That is, if n_i is the length of code word assigned to x_i ($n_i = |C(x_i)|$), we want a code with a small average length of code word. The average length is defined by

$$\bar{n} = \sum_{i=1}^M p_i n_i.$$

- To see that $\bar{n}$ is a reasonable quantity to minimize, look at p_i as describing the frequency of occurrence of x_i . In a message $y_1 \cdots y_{100}$ ($y_i \in \{x_1, \dots, x_M\}$), $p_i = o_i/100$ where o_i is the number of occurrences of x_i . The length of an encoded message is $|C(y_1 \cdots y_{100})| = \sum_{i=1}^M o_i \cdot n_i$. Thus the average length of a code word in the encoded message is $|C(y_1 \cdots y_{100})|/100 = \sum_{i=1}^M \frac{o_i}{100} \cdot n_i = \sum_{i=1}^M p_i n_i = \bar{n}$.
- We thus need to take into account the probabilities p_i and assign code words of different lengths to x_i s. Basic rule is: If p_i is large, $C(x_i)$ is short.

Example

x_i	a	b	c	d
p_i	0.25	0.25	0.25	0.25
C_1	10	100	1000	10000
C_2	00	01	10	11

C_1 : $\bar{n} = 3.5$,

C_2 : $\bar{n} = 2$.

Existence of codes

Problem: Given source symbols $x_1, \dots, x_M$, code symbols $a_1, \dots, a_D$, positive integers $n_1, \dots, n_M$, does there exist a code with certain properties (e.g., prefix, UD) such that n_i is the length of a code word corresponding to x_i ?

Example Is there a better code (shorter $\bar{n}$) than C_2 from the previous slide? One may check that there is not.

The codeword lengths are 2, 2, 2, 2. Is there a code with code word lengths 1, 2, 2, 3 (then $\bar{n}$ would be the same)? No.

Existence of prefix codes

Kraft, Leon G. (1949), A device for quantizing, grouping, and coding amplitude modulated pulses, Cambridge, MA: MS Thesis, Electrical Engineering Department, MIT.

Theorem (Kraft)

A prefix code with code words of lengths $n_1, \dots, n_M$ and a code alphabet of size D exists if and only if

$$\sum_{i=1}^M D^{-n_i} \leq 1.$$

The inequality is known as the Kraft's inequality.

Existence of UD codes

McMillan, B. (1956), Two inequalities implied by unique decipherability, IEEE Trans. Information Theory 2 (4): 115–116, doi:10.1109/TIT.1956.1056818.

Theorem (McMillan)

A UD code with code words of lengths $n_1, \dots, n_M$ and a code alphabet of size D exists if and only if

$$\sum_{i=1}^M D^{-n_i} \leq 1.$$

Thus, for every UD code there exists a prefix code with the same lengths of code words. Hence, in a sense, prefix codes are as efficient as UD codes.

Proof of Kraft and MacMillan theorems

One may easily see that Kraft and MacMillan theorems follows from the following claims.

Claim 1 If the Kraft inequality is satisfied, then there exists a prefix code with codewords of lengths $n_1, \dots, n_M$.

Claim 2 Every UD code satisfies Kraft inequality.

Proof of Claim 1

Let n_i be the required length of code word of x_i , assume $n_1 \leq \dots \leq n_M$.

Construct a prefix code C inductively as follows.

1. $C(x_1)$ is an arbitrary word of length n_1 .
2. With $C(x_1), \dots, C(x_{i-1})$ already defined, let $C(x_i)$ be an arbitrary word of length n_i such that none of $C(x_1), \dots, C(x_{i-1})$ is a prefix of $C(x_i)$.

We need to check that $C(x_i)$ from 2. exists. The number of words with prefix $C(x_j)$ for $j < i$ (forbidden words) is $D^{n_i - n_j}$. Therefore, while the total number of words of length n_i is D^{n_i} , the number of forbidden words is $\sum_{j=1}^{i-1} D^{n_i - n_j}$.

Therefore, we need to show that

$$D^{n_i} - \sum_{j=1}^{i-1} D^{n_i - n_j} \geq 1.$$

Kraft's inequality implies $\sum_{j=1}^i D^{-n_j} \leq 1$, i.e. $1 - \sum_{j=1}^{i-1} D^{-n_j} \geq D^{-n_i}$.

Multiplying this inequality by D^{n_i} we get the required inequality.

Proof of Claim 2

Let $n_i = |C(x_i)|$, $i = 1, \dots, M$.

We need to prove that $\sum_{i=1}^M D^{-n_i} \leq 1$. Denote $\sum_{i=1}^M D^{-n_i}$ by c . Since each $c > 1$ satisfies $\lim_{r \rightarrow \infty} \frac{c^r}{r} = \infty$, to show that $c \leq 1$, it is enough to show that the numbers $\frac{c^r}{r}$ are bounded from above for $r = 1, 2, 3, \dots$.

Computing the powers of c , we get

$$c^r = \sum_{i_1, \dots, i_r=1}^M D^{-(n_{i_1} + \dots + n_{i_r})}.$$

Reorder the members of this sum according to values of $j = n_{i_1} + \dots + n_{i_r}$. Denoting $S = \min\{n_1, \dots, n_M\}$ and $L = \max\{n_1, \dots, n_M\}$, The smallest and the largest value of j is rS and rL , respectively. So we have

$$c^r = \sum_{j=rS}^{rL} \sum_{1 \leq i_1, \dots, i_r \leq M, n_{i_1} + \dots + n_{i_r} = j} D^{-j}.$$

Now, the number of r -tuples $\langle i_1, \dots, i_r \rangle$ for which $n_{i_1} + \dots + n_{i_r} = j$ is at most D^j , the number of sequences of length j of the code alphabet. Namely, $n_{i_1} + \dots + n_{i_r}$ is the length of $C(x_{i_1} \dots x_{i_r})$. The number of such r -tuples is thus equal to the number of source messages $x_{i_1} \dots x_{i_r}$ whose code $C(x_{i_1} \dots x_{i_r})$ has length j . Because C is UD, there number of such messages is at most D^j .

We therefore get

$$c^r = \sum_{j=rS}^{rL} \sum_{1 \leq i_1, \dots, i_r \leq M, n_{i_1} + \dots + n_{i_r} = j} D^{-j} \leq \sum_{j=rS}^{rL} D^j D^{-j} = \sum_{j=rS}^{rL} 1 = rL - rS + 1 \leq rL$$

from which we get $\frac{c^r}{r} \leq L$, proving that $\frac{c^r}{r}$ are bounded.

Shannon's noiseless coding theorem

Recall: Symbols $x_1, \dots, x_M$ occur with probabilities $p_1, \dots, p_M$.

Code C with code words of lengths $n_1, \dots, n_M$. $\bar{n} = \sum_{i=1}^M p_i n_i$ is the average length of a code word.

Denote by $E(X)$ the corresponding entropy, i.e. $E(X) = -\sum_{i=1}^M p_i \lg p_i$. ($\lg$ is logarithm to base 2.)

Theorem (Shannon)

Every UD code with code alphabet of size D satisfies

$$\bar{n} \geq E(X) / \lg D$$

with equality if and only if $p_i = D^{-n_i}$ for $i = 1, \dots, M$.

Note that $E(X) / \lg D$ is the entropy of X computed using logarithms to base D . Namely, $\frac{E(X)}{\lg D} = \frac{E(X)}{\log_2 D} = -\sum_{i=1}^M p_i \frac{\log_2 p_i}{\log_2 D} = -\sum_{i=1}^M p_i \log_D p_i$.

Codes with smallest $\bar{n}$ are called **optimal**. Codes for which $\bar{n} = E(X) / \lg D$ are called **absolutely optimal**.

Example Source symbols a, b, c with probabilities $p_a = 0.5$, $p_b = 0.25$, $p_c = 0.25$. Code $C(a) = 0$, $C(b) = 10$, $C(c) = 11$ is absolutely optimal. Namely, $\bar{n} = 1.5$. $E(X) = -0.5 \lg 0.5 - 0.25 \lg 0.25 - 0.25 \lg 0.25 = 1.5$.

Example Source symbols a, b, c, d with probabilities $p_a = 0.4$, $p_b = 0.2$, $p_c = 0.2$, $p_d = 0.2$. Code $C(a) = 0$, $C(b) = 10$, $C(c) = 110$, $C(d) = 111$ is not absolutely optimal because $\bar{n} = 2 > E(X)$. However, this code is optimal (check).

Example Source symbols a, b, c, d with probabilities $p_a = 0.4$, $p_b = 0.2$, $p_c = 0.2$, $p_d = 0.2$. Code $C(a) = 00$, $C(b) = 01$, $C(c) = 10$, $C(d) = 11$ is not absolutely optimal because $\bar{n} = 2 > E(X)$. However, this code is optimal (check).

Proof of Shannon's noiseless coding theorem

We need: **Lemma** For positive $p_1, q_1, \dots, p_k, q_k$ with $\sum_{i=1}^k p_i = \sum_{i=1}^k q_i = 1$ we have $-\sum_{i=1}^k p_i \log p_i \leq -\sum_{i=1}^k p_i \log q_i$ with equality if and only if $p_i = q_i$ for $i = 1, \dots, k$ (see part on information theory).

$\bar{n} \geq E(X)/\lg D$ is equivalent to

$$\lg D \sum_{i=1}^M p_i n_i \geq -\sum_{i=1}^M p_i \lg p_i.$$

Due to $\lg D p_i n_i = p_i \lg D^{n_i} = -p_i \lg D^{-n_i}$, the latter inequality may be rewritten as

$$-\sum_{i=1}^M p_i \lg D^{-n_i} \geq -\sum_{i=1}^M p_i \lg p_i.$$

We want to apply Lemma. To this end, put $q_i = D^{-n_i} / \sum_{j=1}^M D^{-n_j}$. Then Lemma may be applied to p_i and q_i and yields

$$-\sum_{i=1}^M p_i \lg \frac{D^{-n_i}}{\sum_{j=1}^M D^{-n_j}} \geq -\sum_{i=1}^M p_i \lg p_i = E(X) \quad (1)$$

with equality iff $p_i = D^{-n_i} / \sum_{j=1}^M D^{-n_j}$.

We thus have

$$\begin{aligned} E(X) &\leq -\sum_{i=1}^M p_i \lg D^{-n_i} + \sum_{i=1}^M p_i \lg \sum_{j=1}^M D^{-n_j} = \\ &\sum_{i=1}^M p_i n_i \lg D + \lg \sum_{j=1}^M D^{-n_j} \cdot \sum_{i=1}^M p_i = \\ &\bar{n} \lg D + \lg \sum_{j=1}^M D^{-n_j} \end{aligned}$$

with equality iff $p_i = D^{-n_i} / \sum_{j=1}^M D^{-n_j}$. From MacMillan Theorem we know that $\sum_{j=1}^M D^{-n_j} \leq 1$, hence $\lg \sum_{j=1}^M D^{-n_j} \leq 0$, from which we obtain

$$E(X) \leq \bar{n} \lg D.$$

Furthermore, if $p_i = D^{-n_i}$ for all $i = 1, \dots, M$, then

$$E(X) = \sum_{i=1}^M p_i n_i \lg D = \bar{n} \lg D.$$

It remains to show that if $E(X) = \bar{n} \lg D$ then $p_i = D^{-n_i}$ for all i . Because

$E(X) \leq \bar{n} \lg D + \lg \sum_{j=1}^M D^{-n_j}$ (see above), $E(X) = \bar{n} \lg D$ implies

$\lg \sum_{j=1}^M D^{-n_j} \geq 0$. MacMillan Theorem implies $\lg \sum_{j=1}^M D^{-n_j} \leq 0$ (see

above), whence $\lg \sum_{j=1}^M D^{-n_j} = 0$, from which we obtain $\sum_{j=1}^M D^{-n_j} = 1$.

Now, (1) implies that $p_i = D^{-n_i}$ for all i .

Shannon's noiseless coding theorem cntd.

Given $p_1, \dots, p_M$, it is rare that an absolutely optimal code exists. Namely, if we take n_i such that $p_i = D^{-n_i}$, i.e. $n_i = (-\lg p_i) / \lg D$, then n_i need not be an integer. Obviously, because lengths of code words are integers, an absolutely optimal code does not exist in this case.

However, one can take integers n_i such that

$$\frac{-\lg p_i}{\lg D} \leq n_i < \frac{-\lg p_i}{\lg D} + 1$$

for $i = 1, \dots, M$, i.e. $n_i = \lceil \frac{-\lg p_i}{\lg D} \rceil$.

A question is whether a code with code words of such lengths n_i exists.

Theorem

For every $x_1, \dots, x_M$ occurring with positive probabilities $p_1, \dots, p_M$ there exists a prefix code with code words of lengths $n_1, \dots, n_M$ satisfying $n_i = \lceil \frac{-\lg p_i}{\lg D} \rceil$. For any such code,

$$\frac{E(X)}{\lg D} \leq \bar{n} < \frac{E(X)}{\lg D} + 1.$$

Proof Due to Kraft's Theorem, we need to verify $\sum_{i=1}^M D^{-n_i} \leq 1$.

$n_i = \lceil \frac{-\lg p_i}{\lg D} \rceil$ means

$$(*) \quad \frac{-\lg p_i}{\lg D} \leq n_i < \frac{-\lg p_i}{\lg D} + 1$$

from which we get $\lg p_i \geq -n_i \lg D$, i.e. $p_i \geq D^{-n_i}$. Summing over i we get $\sum_{i=1}^M D^{-n_i} \leq \sum_{i=1}^M p_i = 1$. Now, multiplying $(*)$ by p_i and summing over i gives $E(X)/\lg D \leq \bar{n} < E(X)/\lg D + 1$. □

That is: There exists a code with less than one symbol longer average length of a code word compared to the limit given by Shannon's theorem. (assumption $p_i > 0$ is essential, consider $p_1 = 0, p_2 = 1$, then $\bar{n} < 1 = E(X)/\lg D + 1$ is not possible)

Approaching the lower limit by block codes

Using block codes, one can approach the limit arbitrarily close.

Consider a random vector $Y = \langle X_1, \dots, X_s \rangle$ with $X_i = X$ (independent), and encode the outputs of Y using a code alphabet of size D .

Then there is a code (apply the above theorem) for which

$$\frac{E(Y)}{\lg D} \leq \bar{n}_s < \frac{E(Y)}{\lg D} + 1$$

where $\bar{n}_s$ is the average length of encoding of an output of Y . Since X_i are independent, $E(Y) = E(X_1) + \dots + E(X_s) = sE(X)$, i.e.

$$\frac{sE(X)}{\lg D} \leq \bar{n}_s < \frac{sE(X)}{\lg D} + 1,$$

i.e.

$$\frac{E(X)}{\lg D} \leq \frac{\bar{n}_s}{s} < \frac{E(X)}{\lg D} + \frac{1}{s}.$$

Note that $\frac{\bar{n}_s}{s}$ is the average number of code characters per value of X . As s grows to infinity, $\frac{\bar{n}_s}{s}$ approaches the lower limit $\frac{E(X)}{\log D}$ arbitrarily close.

Construction of optimal codes

Definition

A code C for X (i.e. $x_1, \dots, x_M$ and $p_1, \dots, p_M$) is **optimal in a class $\mathcal{C}$ of codes** if $C \in \mathcal{C}$ and C has the smallest average code word length $\bar{n}$ of all the codes from $\mathcal{C}$.

Theorem

If C is optimal for X in a class of prefix codes then C is optimal for X in the class of all UD codes.

Proof Suppose C' is a UD code with code word lengths $n'_1, \dots, n'_M$ with a smaller average code word length $\bar{n}'$ than the length $\bar{n}$ of C , i.e. $\bar{n}' < \bar{n}$. MacMillan's theorem implies that $n'_1, \dots, n'_M$ satisfy Kraft's inequality. Kraft's theorem implies that there exists a prefix code with code word lengths $n'_1, \dots, n'_M$. The average length of this prefix code is $\bar{n}'$, and thus less than $\bar{n}$, which contradicts the fact that C is optimal in the class of prefix codes.

In the rest, assume for simplicity $D = 2$ (binary codes)

Theorem (properties of optimal binary codes)

Let C be a prefix code for X . Assume that $p_1 \geq \dots \geq p_M$ and that if $p_i = p_{i+1} = \dots = p_{i+r}$ then $n_i \leq n_{i+1} \leq \dots \leq n_{i+r}$. If C is optimal in the class of prefix codes then

- (a) $p_j > p_k$ implies $n_j \leq n_k$,
- (b) $n_{M-1} = n_M$ (i.e., the two symbols with least probability have code words of equal length);
- (c) among the code words of length n_M (the longest ones), there exist two words which agree in all symbols except the last one.

Proof (a) Suppose $p_j > p_k$ and $n_j > n_k$. Then switching the code words for x_j and x_k we get a code C' with a smaller average length $\bar{n}'$ than the average length $\bar{n}$ of C . Namely,

$$\bar{n}' - \bar{n} = p_j n_k + p_k n_j - (p_j n_j + p_k n_k) = (p_j - p_k)(n_k - n_j) < 0.$$

(b) Clearly, $n_{M-1} \leq n_M$ because either $p_{M-1} > p_M$ and then $n_{M-1} \leq n_M$ by (a), or $p_{M-1} = p_M$ and then $n_{M-1} \leq n_M$ by our assumption. Suppose $n_{M-1} < n_M$. Then, because C is a prefix code, we may delete the last symbol from the code word $C(x_M)$ and still obtain a prefix code for X . This code has a smaller average length than C , a contradiction to optimality of C .

(c) If (c) is not the case, then because C is a prefix code, we may delete the last symbol from all code words of the largest length n_M and still obtain a prefix code. Now this code is better than C , a contradiction.

Huffman construction of optimal codes

D. A. Huffman: A method for the construction of minimum-redundancy codes, Proceedings of the I.R.E., Sept. 1952, pp. 1098–1102.

Assume again $p_1 \geq p_2 \geq \dots \geq p_M$ for $x_1, \dots, x_M$.

Huffman method is recursive.

Constructing a code C_1 for $x_1, \dots, x_M$ with $p_1, \dots, p_M$ reduces to constructing a code C_2 for $x_1, \dots, x_{M-1,M}$ with $p_1, \dots, p_{M-1,M}$, where

- $x_{M-1,M}$ is a new symbol (results by merging x_{M-1} and x_M),
- $p_{M-1,M} = p_{M-1} + p_M$.

Prior to entering the procedure for $x_1, \dots, x_{M-1,M}$ with $p_1, \dots, p_{M-1,M}$, the new probabilities (and correspondingly, new symbols) must be reordered to form a non-increasing sequence, as required by assumption.

From a code C_2 for the new situation ($M - 1$ symbols), Huffman procedure constructs a code C_1 for the original situation (M symbols) as follows:

- code words in C_1 for $x_1, \dots, x_{M-2}$ are the same as those in C_2
- code word for $x_{M-1} =$ appending 0 to code word w for $x_{M-1,M}$ in C_2 ;
- code word for $x_M =$ appending 1 to code word w for $x_{M-1,M}$ in C_2 ;

The terminating situation for the recursive procedure is $M = 2$, i.e. a situation Y of the form y_1, y_2 with $q_1 \geq q_2$.

A code C for this situation is obtained as follows: $C(y_1) = 0$, $C(y_2) = 1$. Clearly, C is an optimal code for Y in the class of all prefix codes (average length is 1, because $q_1|0| + q_2|1| = q_1 + q_2 = 1$).

Optimality of Huffman procedure follows from the following lemma.

Lemma

Given the conditions above, if C_2 is optimal in the class of prefix codes then C_1 is optimal in the class of prefix codes.

Proof Suppose C_1 is not optimal. Let C'_1 be optimal in the class of prefix codes. C'_1 has code words with lengths $n'_1 \geq \dots \geq n'_M$. Above Theorem (b) implies $n'_{M-1} = n'_M$. Due to Theorem (c), we may assume without loss of generality that $C'_1(x_{M-1})$ and $C'_1(x_M)$ differ only in the last symbol (namely, we may switch two words with equal length without changing the average word code length).

Let us merge x_{M-1} and x_M into $x_{M-1,M}$ and construct a new code C'_2 which has the same code words as C'_1 for $x_1, \dots, x_{M-2}$, and $C'_2(x_{M-1,M})$ is obtained from $C'_1(x_M)$ (or, equivalently, from $C'_1(x_{M-1})$) by deleting the last symbol.

It is now easy to check (see below) that C'_2 is a prefix code (for the same problem as C_2) and has a smaller average code word length than C_2 . This is a contradiction to optimality of C_2 .

Proof of Lemma cntd.:

The average code word length $\overline{n'_2}$ of C'_2 is

$$\begin{aligned}\overline{n'_2} &= \sum_{i=1}^{M-2} p_i n'_i + (p_{M-1} + p_M)(n'_{M-1} - 1) = \\ &= \sum_{i=1}^{M-2} p_i n'_i + p_{M-1} n'_{M-1} + p_M n'_M - (p_{M-1} + p_M) = \\ &= \sum_{i=1}^M p_i n'_i - (p_{M-1} + p_M),\end{aligned}$$

and since the average length $\sum_{i=1}^M p_i n'_i$ of C'_1 is by assumption $<$ than the average length $\sum_{i=1}^M p_i n_i$ of C_1 , and since $n_{M-1} = n_M$ (due to the construction of C_1), we continue:

$$\begin{aligned}&= \sum_{i=1}^M p_i n'_i - (p_{M-1} + p_M) < \sum_{i=1}^M p_i n_i - (p_{M-1} + p_M) = \\ &= \sum_{i=1}^{M-2} p_i n_i + (p_{M-1} + p_M) n_{M-1} - (p_{M-1} + p_M) = \\ &= \sum_{i=1}^{M-2} p_i n_i + (p_{M-1} + p_M)(n_{M-1} - 1) = \overline{n_2}\end{aligned}$$

($\overline{n_2}$ is the average code word length of C_2).

The proof is complete.

Optimality of Huffman code

We thus get:

Theorem

For any X , a code obtained by the above procedure is optimal in the class of all prefix codes.

Note that the construction of Huffman codes, as described above, is not unique (one may switch code words for symbols of equal probability).

Example of Huffman code

Source symbols $a, b, 1, 2, 3$ with probabilities 0.3, 0.4, 0.1, 0.1, 0.1.

symbol x_i	b	a	1	2	3
probability p_i	0.4	0.3	0.1	0.1	0.1
code word $C_1(x_i)$	1	00	011	0100	0101
symbol x_i	b	a	2,3	1	
probability p_i	0.4	0.3	0.2	0.1	
code word $C_2(x_i)$	1	00	010	011	
symbol x_i	b	a	2,3,1		
probability p_i	0.4	0.3	0.3		
code word $C_3(x_i)$	1	00	01		
symbol x_i	a,2,3,1	b			
probability p_i	0.6	0.4			
code word $C_4(x_i)$	0	1			

ERROR CORRECTION AND ERROR DETECTION: BASIC CONCEPTS

Introduction

In presence of noise (noisy channel), special codes must be used which enable us to detect or even correct errors.

We assume the following type of error: A symbol that we receive may be different from the one that has been sent.

Basic way to deal with this type of error: Make codes redundant (then e.g., some symbols are used to check whether symbols were received correctly).

Simple example:

A 0 or 1 (denote it a) is appended to a transmitted word w so that wa contains an even number of 1s. If $w = 001$ then $a = 1$ and $wa = 0011$; if $w = 101$ then $a = 0$ and $wa = 1010$. Such code (codewords have length 4) detects single errors because if a single error occurs (one symbol in wa is flipped during transmission), we recognize an error (the number of 1s in the received word is odd). Such code, however, does not correct single errors (we do not know which symbol was corrupted).

A simple code that enables us to correct single errors is: Instead of $a \in \{0, 1\}$, send aaa (triple the symbol). Such code (codewords have length 3, both the source and code alphabets are $\{0, 1\}$) corrects single errors (why?).

Overall goal: Construct codes that correct errors but are not too redundant.

We consider block codes (code words are of equal lengths). Given a code alphabet B , the set of code words C is therefore a subset of B^n where n is a given length.

Rate

Error detection and error correction codes are often constructed the following way. Symbols from source alphabet A are encoded by words u of length k over code alphabet B . We add to these words u additional symbols to obtain words of length n . That is, we add $n - k$ symbols from B to every word u of length k . The original k symbols are called information symbols, the additional $n - k$ symbols are called check symbols and their role is to make the code detect or correct errors. Intuitively, the ratio $R(C) = \frac{k}{n}$ measures the redundancy of the code. If $k = n$, then $R(C) = 1$ (no redundancy added). Small $R(C)$ indicates that redundancy increased significantly by adding the $n - k$ symbols. In general, we have the following definition.

Definition

Let C be a block code of length n . If $|C| = |B|^k$ for some k we say that the **rate** (or information rate) $R(C)$ of code C is

$$R(C) = \frac{k}{n}.$$

Example

(1) **Repetition code** C_n over alphabets $A = B$ has codewords $\{a \cdots a \text{ } n \text{ times} \mid a \in B\}$. For example, C_3 for $A = B = \{0, 1\}$ is $C_3 = \{000, 111\}$. A symbol a in A is encoded by n -times repetition of a . Clearly, $R(C_n) = \frac{1}{n}$.

(2) **Even-parity code** of length n has codewords $a_1 \dots a_n \in \{0, 1\}^n$ such that the number of 1s in the code words is even. For example, for $n = 3$, $C = \{000, 011, 101, 110\}$. The idea is that the first $n - 1$ symbols are chosen arbitrarily, the last symbol is added so that the word has even parity (even number of 1s). Clearly, $R(C) = \frac{n-1}{n}$.

Error detection and correction

Lemma

Let for words $u = u_1 \dots u_n, v = v_1 \dots, v_n \in B^n$,

$$\delta(u, v) = |\{i \mid 1 \leq i \leq n, u_i \neq v_i\}|, \quad (*)$$

i.e. $\delta(u, v)$ is the number of positions in which u and v differ. Then δ is a metric on B^n , i.e.

$$\delta(u, v) = 0 \text{ if and only if } u = v,$$

$$\delta(u, v) = \delta(v, u),$$

$$\delta(u, v) \leq \delta(u, w) + \delta(w, v).$$

Proof The first two conditions are obvious. Third condition (triangular inequality):

Definition

Metric δ defined by (*) is called the **Hamming distance**. A **minimum distance** of a block code C is the smallest distance $d(C)$ of two distinct words from C , i.e. the number $d(C) = \min\{\delta(u, v) \mid u, v \in C, u \neq v\}$.

Example: $\delta(011, 101) = 2$, $\delta(011, 111) = 1$, $\delta(0110, 1001) = 4$,
 $\delta(01, 01) = 0$.

Minimum distance of $C = \{0011, 1010, 1111\}$ is 2; minimum distance of $C = \{0011, 1010, 0101, 1111\}$ is 2.

Exercise Realize that C has minimum distance $> d$ if and only if for every $u \in V$, $\mathcal{B}_d(u) \cap C = \{u\}$. Here, $\mathcal{B}_d(u) = \{v \in B^n \mid \delta(u, v) \leq d\}$ is called the ball with center u and radius d .

Idea: d errors occur if when a code word u is sent through the (noisy) channel, a word v received such that $\delta(u, v) = d$ (d symbols are corrupted). Code C

- detects d errors if whenever $t \leq d$ errors occur, we can tell from the word that was received that errors occurred;
- corrects d errors if whenever $t \leq d$ errors occur, we can tell from the word that was received the code word that was sent.

Formally:

Definition

A code C **detects d errors** if for every code word $u \in C$, no word that results from u by corrupting (changing) at most d symbols is a code word.

The following theorem is obvious.

Theorem

C detects d errors if and only if $d(C) > d$.

So the largest d such that C detects d errors is $d = d(C) - 1$.

Example (1) For the even-parity code C , $d(C) = 2$. Thus C detects 1 error. Notice that this code does not detect 2 errors.

(2) For the repetition code C_n , $d(C) = n$. Therefore, C_n detects $n - 1$ errors.

Definition

A code C **corrects** d **errors** if for every code word $u \in C$ and every word v that results from u by corrupting (changing) at most d symbols, u has the smallest distance from v among all code words. That is, if $u \in C$ and $\delta(u, v) \leq d$ then $\delta(u, v) < \delta(w, v)$ for every $w \in C$, $u \neq w$.

If we receive v with at most d errors, the property from the definition ($\delta(u, v) < \delta(w, v)$ for every $w \in C$) enables us to tell the code word u that was sent, i.e. to decode: go through all code words and select the one closest to v . This may be very inefficient. The challenge consists in designing codes for which decoding is efficient.

Theorem

C corrects d errors if and only if $d(C) > 2d$.

So the largest d such that C corrects d errors is $d = \lfloor (d(C) - 1)/2 \rfloor$.

Proof

Let $d(C) > 2d$. Let $u \in C$, $\delta(u, v) \leq d$. If there is a code word $w \neq u$ with $\delta(w, v) \leq \delta(u, v)$ then using triangle inequality, we get $2d = d + d \geq \delta(w, v) + \delta(u, v) \geq \delta(w, u)$, a contradiction to $d(C) > 2d$ because both w and u are code words. Thus C corrects d errors.

Let C correct d errors. Let $d(C) \leq 2d$. Then there exist code words u, w with $t = \delta(u, w) \leq 2d$. Distinguish two cases.

First, if t is even, let v be a word that has distance $t/2$ from both u and w . v may be constructed the following way: Let $I = \{i \mid u_i \neq w_i\}$ (i.e. I has t elements), let $J \subset I$ such that J has $t/2$ elements. Put $v_i = u_i$ for $i \in J$ and $v_i = w_i$ for $i \notin J$. Then $\delta(u, v) = t/2 = \delta(w, v)$. Therefore, $\delta(u, v) = t/2 \leq d$ but $\delta(u, v) = \delta(w, v)$, a contradiction to the assumption that C corrects d errors.

Second, if t is odd, then $t + 1$ is even and still $t + 1 \leq 2d$. As above, let v have distance $(t + 1)/2$ from both u and w . In this case, $\delta(u, v) = (t + 1)/2 \leq d$ but $\delta(u, v) = \delta(w, v)$, again a contradiction to the assumption that C corrects d errors. The proof is complete.

Example (1) For the even-parity code C , $d(C) = 2$. Thus C does not correct any number of errors ($\lfloor 2 - 1/2 \rfloor = 0$).

(2) For the repetition code C_n , $d(C) = n$. Therefore, C_n corrects $\lfloor n - 1/2 \rfloor$ errors, i.e. $(n - 1)/2$ errors if n is odd and $(n - 2)/2$ errors if n is even.

(3) Two dimensional even-parity code. Information symbols are arranged in a $p - 1 \times q - 1$ array to which we add an extra column and an extra row. The extra column contains parity bits of the original rows (rows contain $q - 1$ symbols), the extra row contains parity bits of the original columns (columns contain $p - 1$ symbols). The bottom-right parity symbol is set so that the parity of the whole array is even. Code words have length pq . Example of a codeword for $(p, q) = (4, 5)$:

0	1	1	0	0
0	0	1	0	1
1	1	1	0	1
1	0	1	0	0

Such codes correct 1 error. If error on position (i, j) occurs, we can detect i by parity symbols on rows (i th symbol of the extra column) and j by parity symbols on columns (j th symbol of the extra row). What is $d(C)$?

(4) Two-out-of-five code (particular case of m -out-of- n code) C . Binary block code of length 5. Code words have exactly two 1s and three 0s. There are $\binom{5}{2} = 10$ such words: 00011, 00101, 00110, 01001, 01010, 01100, 10001, 10010, 10100, 11000. Popular for representing digits 0, 1, ..., 9 using five binary digits.

Check that $d(C) = 2$, hence C detects single errors but does not correct any errors. Note that rate is not defined for this code (10 is not a power of 2).

Notice that if one assigns weights 0, 1, 2, 3, 6 to the five positions, the codewords represent numbers 1, ..., 9. For example, code word 00011 represents digit 9 because $9 = 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 1 \cdot 6$. Every number is represented by exactly one code word, except for 3 which is represented by 10010 and 01100. The latter code word is used to represent digit 0. This way, one may decode.

A block code C of length n is called a **systematic** code if for some $k < n$, for every word $u \in B^k$ there exists exactly one word $v \in B^{n-k}$ for which uv is a code word. (Sometimes systematic codes are defined as a bit more general.)

Both the repetition code and even-parity code are systematic.

Clearly, $R(C) = k/n$.

One may easily verify that for a systematic code, $d(C) \leq n - k + 1$.

This shows: On one hand, we try to have large k (to have small redundancy). On the other hand, the larger k , the smaller $d(C)$, and therefore, the smaller the number of errors the code may correct.

LINEAR CODES

Introduction

Linear codes are efficient codes. The basic idea behind linear codes is to equip the code alphabet B with algebraic operations so that B becomes a field and B^n becomes a linear space (or vector space).

In such case, a block code C of length n (understood as a set of its code words) is called a linear code if it is a subspace of the linear space B^n .

For simplicity, we start with binary linear codes.

BINARY LINEAR CODES

Operations $+$ and $\cdot$ on $\{0, 1\}$

Consider operations $+$ (“binary addition”, addition modulo 2) and $\cdot$ (“binary multiplication”) given by

$$\begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \quad \text{and} \quad \begin{array}{c|cc} \cdot & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}.$$

So, 0 is neutral w.r.t. $+$, we have inverse (or opposite) elements w.r.t. $+$ ($0 + 0 = 0$, hence $-0 = 0$; $1 + 1 = 0$, hence $-1 = 1$).

(Later we see that $\mathbf{Z}_2 = \{0, 1\}$ equipped with $+$ is a commutative group and that $\mathbf{Z}_2$ equipped with $+$ and $\cdot$ is a field.)

Words of length n over $\mathbf{Z}_2$ may be identified with vectors with n components (i.e. 00101 with $\langle 0, 0, 1, 0, 1 \rangle$). The algebraic setting makes it natural to consider e.g. multiplication of words over $\mathbf{Z}_2$ by elements from $\mathbf{Z}_2$: $a \cdot \langle u_1, \dots, u_n \rangle = \langle a \cdot u_1, \dots, a \cdot u_n \rangle$, i.e. $a \cdot u_1 \cdots u_n = a \cdot u_1 \cdots a \cdot u_n$. Example: $1 \cdot 1010 = 1010$, $0 \cdot 1010 = 0000$ (multiplication is somewhat degenerate with $\mathbf{Z}_2$).

Similarly, $\langle u_1, \dots, u_n \rangle + \langle v_1, \dots, v_n \rangle = \langle u_1 + v_1, \dots, u_n + v_n \rangle$.

Codes as (solutions to homogenous) linear equations

From linear algebra: The set of all solutions of a system of homogenous linear equations (i.e. equations are $a_{i1}x_1 + \dots + a_{in}x_n = 0$) forms a linear subspace. Now, important codes we have seen may be described as sets of solutions of homogenous linear equations over $\mathbf{Z}_2$.

Even-parity code of length n . Codewords are exactly the solutions of a single equation

$$x_1 + x_2 + \dots + x_n = 0.$$

Repetition code of length n . Codewords are exactly the solutions of a system of equations

$$\begin{aligned}x_1 + x_n &= 0 \\ \dots & \\ x_{n-1} + x_n &= 0.\end{aligned}$$

Exercise: $n = 6$, describe code given by equations $x_1 + x_2 = 0$, $x_3 + x_4 = 0$, $x_5 + x_6 = 0$.

Exercise: provide a system of equations describing the two-dimensional even-parity code (provide particularly for $(p, q) = (4, 5)$).

$$x_{11} + \cdots + x_{1q} = 0$$

$$x_{21} + \cdots + x_{2q} = 0$$

...

$$x_{p1} + \cdots + x_{pq} = 0$$

$$x_{11} + \cdots + x_{p1} = 0$$

$$x_{12} + \cdots + x_{p2} = 0$$

...

$$x_{1q} + \cdots + x_{pq} = 0$$

Note that the last equation may be removed (the rest still describes the code).

Binary linear block codes

Definition

A binary block code C is called **linear** $u + v \in C$ for every $u, v \in C$.

How does that fit into our picture?

Recall that a non-empty subset C of a linear space V (set of vectors) over K (set of scalars) is called a subspace if (1) $u + v \in C$ for every $u, v \in C$ (closedness under addition), and (2) $a \cdot u \in C$ for every $a \in K$ and $u \in C$ (closedness under scalar multiplication).

In our case, $V = \{0, 1\}^n$, $K = \{0, 1\}$. (1) implies that C contains the word $0 \cdots 0$ (because $0 \cdots 0 = u + u$ for any u). (2) is trivially satisfied for any C containing $0 \cdots 0$. Therefore, a block code C is a linear subspace of $\{0, 1\}^n$ if and only if C is linear according to our definition.

Recall from linear algebra that a set of all solutions of a system of homogenous linear equations is a linear subspace (of the whole linear space, usually $\mathbf{R}^n$). In our setting, one easily proves:

Lemma

For any n , the set C of all solutions of a system of homogenous linear equations over $\mathbf{Z}_2$ is a linear subspace of $\mathbf{Z}_2^n$, hence C is a linear block code.

As a corollary we obtain that the even-parity code, the two-dimensional even-parity code, and the repetition code are linear block codes (because we gave the systems of equations for these codes).

On the other hand, the two-out-of-five code C is not linear because, for example, $00110, 11000 \in C$ but $00110 + 11000 = 11110 \notin C$.

Definition

If word u is sent and v received, the word e which contains 1 exactly at positions in which u and v differ is called the **error word** for u and v .

Therefore, if e is the error word for u and v , we have

$$v = u + e,$$

and hence also $e = v - u (= v + u)$.

Definition

A **Hamming weight** of word u is the number of symbols in u different from 0. A **minimum weight** of a non-trivial block code C is the smallest Hamming weight of a word from C that is different from $0 \cdots 0$.

The Hamming weight of 01011 is 3. The minimum weight of an even-parity code of length n is 2. The minimum weight of a two-dimensional even-parity code with (p, q) is 4.

Theorem

The minimum weight of a non-trivial binary linear code C is equal to $d(C)$.

Proof Let c be the minimum weight of C .

Let u be a code word with Hamming weight c . Clearly,

$$c = \delta(u, 0 \cdots 0) \geq d(C).$$

Conversely, let $d(C) = \delta(u, v)$ for $u, v \in C$. Then $u + v \in C$. Clearly, $\delta(u, v)$ equals the Hamming weight of $u + v$ ($u + v$ has 1s exactly on positions where u and v differ). Therefore, $d(C) = \delta(u, v) = \text{Hamming weight of } u + v \geq c$. □

Definition

A **parity check matrix** of a binary linear block code C of length n is a binary matrix H such that for every $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ we have

$$H \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ if and only if } x \text{ is a codeword of } C.$$

For brevity, we write $\mathbf{0}$ for a vector consisting of 0s only. We denote the transpose of a vector x (a row) by x^T (a column). Therefore, H is a parity check matrix for C if

$$C = \{x \in \{0, 1\}^n \mid Hx^T = \mathbf{0}^T\}.$$

The following view will be justified later: If rows of H are independent (in linear algebra sense, i.e. none may be removed without changing the corresponding code), H may be seen as a $n - k \times n$ matrix ($n - k$ is the number of check symbols). Each row of H (i.e. each equation) determines one symbol of a code word.

Examples (1) A parity check matrix for the even-parity code of length n is a $1 \times n$ matrix H given by

$$H = (1 \cdots 1).$$

(2) A parity check matrix for the repetition code of length 5 is a 4×5 matrix H given by

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

(3) A parity check matrix for the two-dimensional even-parity code with $(p, q) = (3, 4)$ is the matrix H given by

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Again, note that the last row may be deleted (the matrix without the last row describes the same code). (Namely, row 7 is the sum of all previous rows, i.e. a linear combination of them.)

Hamming codes (perfect for single errors)

Hamming codes are important codes that correct single errors. Code words have length $n = 2^m - 1$ for some integer m ; m is the number of check symbols; hence, $2^m - m - 1$ is the number of information symbols. Such code is also called $(2^m - 1, 2^m - m - 1)$ code (e.g. $(7, 4)$ code). Minimum distance of a Hamming code is 3.

The parity check matrix is thus an $m \times 2^m - 1$ matrix. The columns are formed by all the $2^m - 1$ non-zero binary vectors of length m . It is convenient to order the columns according to the values of the numbers represented by the columns (if viewed as binary representations of numbers $1, \dots$).

Thus, for $m = 2$, the parity check matrix (of the $(3, 1)$ code) is

$$H = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

Why?

Theorem

- (1) *If a binary linear code C corrects single errors then each parity check matrix of C has non-zero and mutually distinct columns.*
- (2) *Every binary matrix with non-zero and mutually distinct columns is a parity check matrix of a binary linear code that corrects single errors.*

Proof If C corrects single errors, it has a minimum distance and thus a minimum Hamming weight > 2 (according to the criterion $d(C) > 2d$ with $d = 1$). Let H be a parity check matrix for C . Let $b^i \in \{0, 1\}^n$ have 1 exactly at position i ($b_j^i = 1$ for $j = i$, $b_j^i = 0$ for $j \neq i$); let $b^{i,j} \in \{0, 1\}^n$ have 1 exactly at positions i and j . If column i of H has only 0s, then $Hb^{i,T} = \mathbf{0}^T$, i.e. b_i is a code word with Hamming weight 1, a contradiction to the fact that the minimum weight of C is > 2 . Similarly, if columns i and j of H are equal, then $Hb^{i,j,T} = \mathbf{0}^T$ ($Hb^{i,j,T}$ is the sum of columns i and j), i.e. $b^{i,j}$ is a code word of weight 2, a contradiction again.

...cntd.

Conversely, let H have non-zero columns that are mutually distinct. Then using the arguments above, one can easily see that no b^i or $b^{i,j}$ is a code word. Hence, the minimum weight of C is > 2 . This implies that C corrects single errors. $\square$

According to the previous theorem, matrix

$$H = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

is a parity check matrix of a binary linear code correcting single errors. Codewords have length 2 and $C = \{00\}$. Clearly, the first row may be removed. The resulting matrix would still be a parity check matrix of C ($0 \cdot x_1 + 0 \cdot x_2$ is always the case). More generally and rigorously, the first row may be removed because it is a linear combination of some of the other rows, i.e. is linearly dependent on the other rows (in this case, this is trivial, e.g. $(0, 0) = 0 \cdot (0, 1)$). In order to have a matrix in which none row is linearly dependent on the other rows, the number of columns must be $\geq$ number of rows. An example is

$$H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Codewords have length 3. The corresponding code is $C = \{000\}$, again a trivial code (because, as was mentioned above, the number of check symbols (rows) equals the number of all symbols (columns)).

Let us improve.

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Codewords have length 4. The corresponding code is $C = \{0000, 1101\}$. Now we have 1 information bit (e.g. the first one) and 3 control bits.

(By the way, what happens if we switch two or more columns in a parity check matrix H ? How is the code corresponding to H related to the code corresponding to the new matrix?)

Now adding further columns (non-zero, so that no two are equal) we construct parity check matrices with code words of codes with lengths $5, \dots$, each having 3 check symbols. Every such code corrects single errors but none corrects 2 errors.

(Why: Correcting 2 errors requires minimum weight > 4 . However, with 3 rows and $n \geq 4$ columns there always exist three or four columns whose sum is a zero column. Namely, take any four columns i, j, k, l . As $4 > m = 3$, they are linearly dependent, say i is a linear combination of j, k, l . In this combination $H_{\cdot i} = a_j \cdot H_{\cdot j} + a_k \cdot H_{\cdot k} + a_l \cdot H_{\cdot l}$, two or three coefficients must be equal to 1, say j, k or j, k, l . Then the sum of columns i, j, k or i, j, k, l is the zero column. Now, this means that for the vector u with 1s exactly at positions i, j, k or i, j, k, l , $Hu^T = \mathbf{0}^T$, i.e. u is a codeword. So, C has minimum weight at most 3 or 4.)

Therefore adding further columns (non-zero ones, so that no two columns in the resulting matrix are equal), we get an $3 \times n$ parity check matrix H of a code correcting single errors with 3 check symbols and $n - 3$ information symbols ($n - 3$ symbols of the code words may be chosen arbitrarily). The larger n , the better the code (more information symbols). The largest possible n is $n = 7$ because there are exactly 7 non-zero columns.

The same reasoning applies for general m , in which case the largest possible n , and thus the best code, is obtained for $n = 2^m - 1$. Matrices constructed this way are just parity check matrices of Hamming codes.

Definition

A **Hamming code** is a binary linear code which has, for some m , an $m \times 2^m - 1$ parity check matrix whose columns are all non-zero binary vectors.

A parity check matrix (one of them) of a Hamming code for $m = 3$ is the 3×7 matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The corresponding system of homogenous linear equations is:

$$\begin{array}{rclcl} & & x_4 + & x_5 + x_6 + & x_7 = 0 \\ & x_2 + & x_3 + & x_6 + & x_7 = 0 \\ x_1 + & x_3 + & x_5 + & & x_7 = 0 \end{array}$$

A simple computation shows that the previous system is equivalent to:

$$x_5 = x_2 + x_3 + x_4$$

$$x_6 = x_1 + x_3 + x_4$$

$$x_7 = x_1 + x_2 + x_4$$

This enables us to look at the codewords the following way: Choose $x_1, x_2, x_3, x_4 \in \{0, 1\}$ arbitrarily (information symbols) and compute x_5, x_6, x_7 (check symbols) according to the above equations. This is how **encoding** is done with Hamming codes (this is the same in principle for $m > 3$).

Decoding If u is a codeword then $Hu^T = \mathbf{0}^T$. If a single error occurs on position i then we receive word $v = u + b_i$ where u is a code word that was sent. In such case,

$$Hv^T = H(u + b_i)^T = Hu^T + Hb_i^T = \mathbf{0}^T + Hb_i^T = Hb_i^T = H_{\cdot i},$$

i.e. Hv^T equals the i th column of H . Because the i th column of H is a binary representation of number i , we need to correct (switch) the symbol of v on a position whose binary representation is Hv^T .

Example We want to send 1010. Encoding: We compute $x_5 = 1$, $x_6 = 0$, and $x_7 = 1$, and send $x_1 \cdots x_7 = 1010101$.

If no error occurs, we receive $v = 1010101$. Decoding: We compute Hu^T and since $Hu^T = \mathbf{0}^T$, we decide that no error occurred and output 1010101.

If an error occurs at position 6, we receive $v = 1010111$. Decoding: We compute Hu^T and since $Hu^T = (110)^T$, we decide that an error occurred at position whose binary representation is 110, i.e. at position 6. We correct the symbol at position 6 and output 1010101. $\square$

The rate of a Hamming code C for m is $R(C) = \frac{2^m - m - 1}{2^m - 1} = 1 - \frac{m}{2^m - 1}$ which quickly converges to 1 as m increases (small redundancy). But we pay for the the small redundancy in case of large m : Of the $2^m - m - 1$ symbols, we are able to correct only one.

Remark Hamming codes are perfect codes. We do not define this concept but: They have the smallest possible redundancy among all codes of length $2^m - 1$ correcting single errors. This may be seen by realizing that for code with m , every word of length $n = 2^m - 1$ is either a code word or has distance 1 from exactly one code word.

LINEAR CODES

Groups, fields, linear algebra

... recalling basic concepts.

A **group** is an algebra $\mathbf{G} = \langle G, + \rangle$ such that $+$ is a binary operation on G satisfying:

- for every $x, y, z \in G$: $x + (y + z) = (x + y) + z$ (associativity);
- there exists $0 \in G$ such that for every $x \in G$: $x + 0 = 0 + x = x$ (0 is a neutral element);
- for every $x \in G$ there exists $y \in G$ such that $x + y = y + x = 0$ (existence of inverse elements).

If, moreover, $x + y = y + x$ for every $x, y \in G$, we speak of a commutative group. As usual, we use just G instead of $\mathbf{G}$, $\langle G, + \rangle$ instead of $\langle G, +, 0 \rangle$, etc.

Examples Integers $\mathbf{Z}$ and reals $\mathbf{R}$ with natural addition form groups.

$\mathbf{Z}_p = \{0, \dots, p-1\}$ with addition modulo p forms a group. (We used $\mathbf{Z}_2$ in binary linear codes.)

A **field** is an algebra $\mathbf{F} = \langle F, +, \cdot, 0, 1 \rangle$ such that $+$ and $\cdot$ are a binary operations on F satisfying:

- $\langle F, +, 0 \rangle$ is a commutative group,
- $\langle F - \{0\}, \cdot, 1 \rangle$ is a commutative group,
- for every $x, y, z \in G$: $x \cdot (y + z) = x \cdot y + x \cdot z$ (distributivity).

Some consequences: $x \cdot 0 = x$; $xy = xz$ implies $y = z$.

Examples Reals $\mathbf{R}$ with natural addition and multiplication form a field.
Important: If p is prime, then $\mathbf{Z}_p$ with addition and multiplication modulo p , and 0 and 1 forms a field.

Let $\mathbf{F} = \langle F, +, \cdot, 0, 1 \rangle$ be a field. A **linear space** (or vector space) over $\mathbf{F}$ is a structure $\mathbf{V}$ consisting of a non-empty set V (set of vectors), a binary operation $+$: $V \times V \rightarrow V$ (vector addition), an operation $\cdot$: $F \times V \rightarrow V$ (scalar multiplication) such that

- $\langle V, + \rangle$ is a commutative group,
- $(ab)\mathbf{v} = a(b\mathbf{v})$ for every $a, b \in F$ and $\mathbf{v} \in V$,
- $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$ for every $a \in F$ and $\mathbf{u}, \mathbf{v} \in V$,
- $1\mathbf{u} = \mathbf{u}$ for every $\mathbf{u} \in V$.

Examples For any field F , the set $F^n = \{ \langle a_1, \dots, a_n \rangle \mid a_i \in F \}$, with addition $+$ and multiplication $\cdot$ defined by

$$\begin{aligned} \langle a_1, \dots, a_n \rangle + \langle b_1, \dots, b_n \rangle &= \langle a_1 + b_1, \dots, a_n + b_n \rangle \\ a \cdot \langle a_1, \dots, a_n \rangle &= \langle a \cdot a_1, \dots, a \cdot a_n \rangle \end{aligned}$$

forms a linear space. For $F = \mathbf{R}$, we get the well-known n -dimensional linear space known from basic linear algebra. For $F = \mathbf{Z}_2$, we get the linear space we used in binary linear codes.

Let $\mathbf{V}$ be a linear space over $\mathbf{F}$, U a non-empty subset of V which satisfies

- $\mathbf{u} + \mathbf{v} \in U$ for every $\mathbf{u}, \mathbf{v} \in U$,
- $a\mathbf{u} \in U$ for every $a \in F$ and $\mathbf{u} \in V$. Then U (equipped with the restrictions of $+$ and $\cdot$) is called a **linear subspace** of $\mathbf{V}$.

Examples For any field F , the set $\{\langle 0, a_2, \dots, a_n \rangle \mid a_i \in F\}$ is a linear subspace of F^n . For any $\mathbf{u} \in F^n$, $\{a\mathbf{u} \mid a \in F\}$ forms a linear subspace of F^n . $\{\mathbf{0}\}$ forms a linear subspace (trivial subspace) of any linear space.

A **linear combination** of vectors $\mathbf{u}_1, \dots, \mathbf{u}_n \in V$ is any vector $a_1\mathbf{u}_1 + \dots + a_n\mathbf{u}_n$ where $a_1, \dots, a_n \in F$.

Theorem Given vectors $\mathbf{u}_1, \dots, \mathbf{u}_n \in V$, the set of all linear combinations of these vectors forms a linear subspace of $\mathbf{V}$. This subspace, denoted by $\text{span}(\{\mathbf{u}_1, \dots, \mathbf{u}_n\})$, is the smallest subspace of $\mathbf{V}$ containing all $\mathbf{u}_i$ s.

A collection U of vectors from $\mathbf{V}$ is called **linearly independent** if no $\mathbf{u} \in U$ is a linear combination of other vectors from U (i.e. from $U - \{\mathbf{u}\}$). A **basis** of $\mathbf{V}$ is a linearly independent subset $U \subseteq V$ for which $\text{span}(U) = V$. If $\mathbf{V}$ has a finite basis, it is called finitely dimensional; if it has a basis with k vectors, it is called k -dimensional (in such case, any basis has k vectors).

If $U = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ is a basis of $\mathbf{V}$, then for every $\mathbf{u} \in V$ there exists unique scalars $a_1, \dots, a_n$ for which $\mathbf{u} = a_1\mathbf{e}_1 + \dots + a_n\mathbf{e}_n$. Hence, if the field F is finite and has r elements, every k -dimensional space over F has r^k elements.

Every collection $\mathbf{u}_1, \dots, \mathbf{u}_p$ of linearly independent vectors of $\mathbf{V}$ may be completed to a collection $\mathbf{u}_1, \dots, \mathbf{u}_p, \dots, \mathbf{u}_k$ that forms a basis of $\mathbf{V}$.

Matrix over field F : A with elements $A_{ij} \in F$.

Systems of homogenous equations ... behave as in the well-known case of $F = \mathbf{R}$.

Operations with matrices: sum, multiplication by a scalar, multiplication of matrices.

Linear codes

Definition Let F be a finite field. A **linear** (n, k) -**code** is any k -dimensional linear subspace of F^n (F^n may be thought of as the linear space of “all words over alphabet F of length n ”).

A linear (n, k) -code has k information symbols and $n - k$ check symbols.

Namely, C as a k -dimensional subspace of F^n has a basis $\mathbf{e}_1, \dots, \mathbf{e}_k$.

Every word $u = u_1 \cdots u_k$ determines a code word $v = \sum_{i=1}^k u_i \mathbf{e}_i$.

Conversely, to every code word $\mathbf{v} \in C$ there exists a unique word

$u = u_1 \cdots u_k$ such that $v = \sum_{i=1}^k u_i \mathbf{e}_i$.

Therefore, if $|F| = r$, then C has r^k code words.

Definition If C is a linear code with basis $\mathbf{e}_1, \dots, \mathbf{e}_k$, then the $k \times n$ matrix G

$$G = \begin{pmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_k \end{pmatrix} = \begin{pmatrix} e_{11} & \cdots & e_{1n} \\ \vdots & \vdots & \vdots \\ e_{k1} & \cdots & e_{kn} \end{pmatrix}$$

is called a **generator matrix** of C .

Examples (1) Hamming (7,4)-code has a generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Why: Every row is a codeword. The rows are independent (G has rank 4, see e.g. the first four columns). The dimension of the code is 4. Therefore, the rows form a basis.

(2) Even-parity code of length 4 has a generator matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Why: Every row is a codeword. The rows are independent (G has rank 3, see e.g. the last three columns). The dimension of the code is 3. Therefore, the rows form a basis.

Every matrix obtained from G by elementary transformations (permuting rows, multiplication of a row by a non-zero scalar, replacing a row by the sum of itself and a linear combination of other rows) is also a generator matrix of the same code. For example, the following matrix results from G by elementary transformations:

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

(3) Let $F = \mathbf{Z}_3$. Matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \text{ is a generator matrix of a } (6,3)\text{-code. Why:}$$

The rows are linearly independent and hence form a basis of a 3-dimensional subspace (i.e. the code) of F^6 . By elementary

transformation, we get $G' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$. Every code word is

a linear combination of the rows. So the code consists of words in which every symbol is repeated twice (112200, 220011, etc.).

Let G be an $k \times n$ a generator matrix. By properties of linear spaces, every word (vector) $u = u_1 \cdots u_k$ uniquely determines a code word v (code words are vectors of the k -dimensional subspace of F^n spanned by the rows of G , i.e. by the basis):

$$\mathbf{v} = \mathbf{u}G$$

The transformation $\mathbf{u} \mapsto \mathbf{u}G$ is the **encoding rule** for a linear code with G .

Different generator matrices lead to different encoding rules. The most convenient? When we select $u_1 \cdots u_k$ (information symbols) and the assigned code word is $u_1 \cdots u_k v_{k+1} \cdots v_n$ (completion by check symbols). In this case, code would be a systematic code, see above and definition below. This is the case when the generator matrix has the form

$$G = (I|B)$$

where I is the $k \times k$ identity matrix (1s on the main diagonal, 0s elsewhere) and B is a $k \times n - k$ matrix.

Definition A linear code is called **systematic** if it has a generator matrix of the form $G = (I|B)$.

Codes C_1 and C_2 are called **equivalent** if there exists a permutation π of $\{1, \dots, n\}$ such that for every $u_1, \dots, u_n \in F$:

$$u_1 \cdots u_n \in C_1 \text{ if and only if } u_{\pi(1)} \cdots u_{\pi(n)} \in C_2,$$

i.e. the code words of the codes differ only in the order of their symbols.

The Hamming (7,4)-code is systematic. The even-parity code of length 4 is systematic. (See the matrices above.) The (6,3)-code over $\mathbf{Z}_3$ above is not systematic. However, it is equivalent to a systematic code with generator matrix G'' which results from the matrix G' of the code by switching the second and fourth columns and the third and fifth columns. That is, the code given by G' is equivalent to the one given by G'' due to permutation π given by $(\pi(1), \pi(2), \dots, \pi(n)) = (1, 4, 5, 2, 3, 6)$.

Theorem Every linear code is equivalent to a systematic code.

Proof.

Definition An n -column matrix H over a field F is called a **parity-check matrix** of a linear code C if for every word $\mathbf{u} \in F^n$ we have

$$\mathbf{v} \in C \text{ if and only if } H\mathbf{v}^T = \mathbf{0}^T.$$

See examples of parity-check matrices for various codes above.

Theorem For a systematic code C with a generator matrix $G = (I|B)$ (I is a $k \times k$ identity matrix), matrix $H = (-B^T|J)$ (J is an $n - k \times n - k$ identity matrix) is a parity-check matrix of C . As a consequence, every linear (n, k) -code has an $n - k \times n$ parity-check matrix of rank $n - k$.

Proof.

Example We know that the following matrix is a generator matrix of Hamming (7, 4)-code.

$$G = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right) = (I|B)$$

Therefore,

$$H = (-B^T|J) = \left(\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right)$$

is a parity-check matrix of the code. Notice that this is not the first parity check matrix of the code we obtained (with columns being binary representations of numbers 1–7). However, this is just the matrix of the system of homogenous equations we obtained from the first parity-check matrix by elementary transformations.

We see an advantage of H being in the form $H = (-B^T|J)$: It gives a way to compute the check symbols (last $n - k$ columns) from the information symbols (first k columns).

Exercise Determine a parity-check matrix H'' of the code over $\mathbf{Z}_3$ given by G'' (above).

Exercise Determine a generator matrix and parity-check matrix for the even-parity code of length 4.

Recall: If $\mathbf{u}$ is the word sent and $\mathbf{v}$ is the word received, we call the difference $\mathbf{e} = \mathbf{v} - \mathbf{u}$ the error pattern. We look for error patterns that have minimum weights. Why: because we adhere to a maximum likelihood decoding: If we receive a code word, we assume no error occurred. If we do not receive a codeword, we assume that the code word with the smallest Hamming distance from the word we received was sent.

Definition Let H be an $m \times n$ parity-check matrix of code C . A **syndrome** of word $\mathbf{v}$ is the word $\mathbf{s}$ defined by

$$\mathbf{s}^T = H\mathbf{u}^T.$$

We observed the following lemma before.

Lemma Let $\mathbf{e}$ be the error pattern of $\mathbf{v}$. Then $\mathbf{e}$ and $\mathbf{v}$ have the same syndrome.

Lemma For an (n, k) -linear code C with minimum weight d ,

$$d \leq n - k + 1.$$

Proof.

Again: Because we want to detect (correct) a possibly large number of errors, we want d large. On the other hand, we want large k (large number of information symbols, i.e. small redundancy). These are contradictory requirements.

A general method for determining the error pattern and thus the word sent from the word received:

Let C be a linear (n, k) -code with a parity-check matrix H with linearly independent rows (such as the one determined from a generator matrix of C , see above).

For every word $\mathbf{e} \in F^n$, consider the set (called coset)

$$\mathbf{e} + C = \{\mathbf{e} + \mathbf{u} \mid \mathbf{u} \in C\}.$$

$\mathbf{e} + C$ is the set of all possible words we may receive when error $\mathbf{e}$ occurs. We know that all words from $\mathbf{e} + C$ have the same syndrome.

Given a syndrome $\mathbf{s}$, we want to find $\mathbf{e}$ with smallest weight that has syndrome $\mathbf{s}$, i.e. for which $H\mathbf{e}^T = \mathbf{s}^T$ (such $\mathbf{e}$ is called a coset leader for $\mathbf{s}$). The system of equations is solvable because H has independent rows.

Procedure:

1. When word $\mathbf{v}$ is received, compute its syndrome $\mathbf{s}^T = H\mathbf{v}^T$.
2. Determine the coset leader for $\mathbf{s}$.
3. Report $\mathbf{u} = \mathbf{v} - \mathbf{e}$ as the word that was sent.

Example

Consider a binary code with the following parity-check matrix:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

There are 4 possible syndromes with the following coset leaders:

syndrome 00, coset leader 00000,

syndrome 01, coset leader 00001,

syndrome 10, coset leader 10001,

syndrome 11, coset leader 10000.

Suppose we receive 11010. The syndrome is $\mathbf{s} = 11$. The corresponding coset leader is 10000. Thus we assume that the word sent was

$$\mathbf{u} = \mathbf{v} - \mathbf{e} = 11010 - 10000 = 11010 + 10000 = 01010.$$

We saw the general ideas behind all linear codes.

We saw examples of some linear codes, among them the perfect Hamming codes.

Several other important codes were designed, including:

Reed-Muller codes (see the copied material, study yourself),
cyclic codes,
BCH codes,
convolutional codes.

Zapoctove prace (pro 1 studenta, pokud neni uvedeno jinak)

- 1 Pojem information divergence a jeho pouziti. Vystupy: Pisemna studie (5-10 stran).
- 2 Pojem asymptotic equipartition property a jeho pouziti. Vystupy: Pisemna studie (5-10 stran).
- 3 Ruzne pristupy ke konstrukci delicich atributu pri konstrukci rozhodovaciho stromu (zejm. information gain). Vystupy: Pisemna studie (5-10 stran).
- 4 Implementace algoritmu pro konstrukci rozhodovaciho stromu z prednasek. Experimenty s realnymi daty (student vybere tabulku dat, program sestoji rozhodovaci strom, student strom okomentuje). Vystupy: Zdrojovy kod, vstupni a vystupni data, dokument popisujici vysledky (1-2 strany).
- 5 Implementace algoritmu pro konstrukci rozhodovaciho stromu z prednasek. Experimenty s variantou algoritmu, která je tolerantni vuci chybam. Vystupy: Zdrojovy kod, vstupni a vystupni datove mnoziny, dokument popisujici vysledky (1-2 strany). 2 studenti
- 6 Huffmanovo kodovani nad kodovou abecedou obsahujici d znaku. Vystupy: Zdrojovy kod, pisemna studie (5-10 stran).

- 7 Axiomatizace pojmu entropie (dukaz jednoznacnosti). Vystupy: Pisemna studie (5-10 stran).
- 8 Axiomatic approach to measuring of information of sign-based image representations (clanek prof. Broneviche)- Vystupy: Pisemna studie (10-15 stran). 2 studenti
- 9 A greedy algorithm for supervised discretization (clanek Butterworth et al.). Vystupy: Zdrojovy kod, pisemna studie (5-10 stran). 2 studenti
- 10 A new metric splitting criterion for decision trees (clanek Simovici). Vystupy: Pisemna studie (10-15 stran). 2 studenti
- 11 Betweenness, metrics and entropy in lattices (clanek Simovici). Vystupy: Pisemna studie (10-15 stran). 2 studenti
- 12 An axiomatization of partition entropy (clanek Simovici). Vystupy: Pisemna studie (10-15 stran). 2 studenti
- 13 A new metric splitting criterion for decision trees (clanek Simovici). Vystupy: Zdrojovy kod, pisemna studie (5 stran). 2 studenti