

# 1 Osnove

## 1.1 Ponovitev logaritmov

- $\log_a x = \frac{\log_b x}{\log_b a}$
- $\log_b\left(\frac{x}{y}\right) = \log_b x - \log_b y$
- $x = b^y \implies \log_b x = y$
- $\log_2 x = \log x$
- $0 \log 0 = 0$

## 1.2 Entropija

je povprečje vseh lastnih informacij:

$$H(X) = \sum_{i=1}^n p_i I_i = - \sum_{i=1}^n p_i \log p_i$$

Lastnosti: je zvezna, simetrična funkcija (vrstni red  $p_i$  ni pomemben, sestevanje je komutativno). Je vedno večja od 0 ( $p_i \geq 0 \rightarrow -p_i \log p_i \geq 0 \rightarrow H(X) \geq 0$ ) in navzgor omejena z  $\log n$ .

Ce sta dogodka **neodvisna** velja aditivnost:  $H(X, Y) = H(X) + H(Y)$ .

Vec zaporednih dogodkov neodvisnega vira:  $X^l = X \times \dots \times X \rightarrow H(X^l) = lH(X)$ .

# 2 Kodi

## 2.1 Uvod

**Kod** sestavljajo *kodne zamenjave*, ki so sestavljene iz znakov **kodne abecede**. Stevilo znakov v kodni abecedi označujemo z  $r$ .

Ce so  $\{p_1, \dots, p_n\}$  verjetnosti znakov  $\{s_1, \dots, s_n\}$  osnovnega sporočila in  $\{l_1, \dots, l_n\}$  dolzine prejetih kodnih zmanjav, je povprečna dolžina kodne zamenjave

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

## 2.2 Tipi kodov

- **optimalen** - ce ima najmanjšo možno dolžino kodnih zamenjav
- **idealen** - ce je povprečna dolžina kodnih zamenjav enaka entropiji
- **enakomeren** - ce je dolžina vseh kodnih zamenjav enaka
- **enoznacen** - ce lahko poljuben niz znakov dekodiramo na en sam način
- **trenuten** - ce lahko osnovni znak dekodiramo takoj, ko sprejmemo celotno kodno zamenjavo

## 2.3 Kraftova neenakost

Za dolzine kodnih zamenjav  $\{l_1, \dots, l_n\}$  in  $r$  znaki kodne abecede obstaja trenuten kod, iff

$$\sum_{i=1}^n r^{-l_i} \leq 1$$

## 2.4 Povp. dolžina, učinkovitost

Najkrajše kodne zamenjave imamo, ce velja:

$$H_r(X) = L \rightarrow l_i = \lceil -\log_r p_i \rceil$$

Učinkovitost koda:

$$\eta = \frac{H(X)}{L \log_r}, \eta \in [0, 1]$$

Kod je **gospodaren**, ce je  $L$  znotraj:

$$H_r(X) \leq L < H_r(X) + 1$$

kjer je  $H_r(X)$ :

$$H_r(X) = - \sum_{i=1}^n \frac{\log p_i}{\log_r} = \frac{H(X)}{\log_r}$$

## 2.5 Shannonov prvi teorem

Za nize neodvisnih znakov dooline  $n$  obstajajo kodi, za katere velja:

$$\lim_{n \rightarrow \infty} \frac{L_n}{n} = H(X)$$

pri cemer je  $H(X)$  entropija vira  $X$ .

Postopek kodiranja po Shannonu:

1. znake razvrstimo po padajocih verjetnostih
2. določimo stevilo znakov v vsaki kodni zamenjavi ( $l_k$ )
3. za vse simbole izračunamo kumulativne verjetnosti ( $P_k = \sum_{i=1}^{k-1} p_i$ )
4.  $P_k$  pretvorimo v bazo  $r$ . Kodno zamenjavo predstavlja prvih  $l_k$  znakov necellega dela stevila

## 2.6 Fanojev kod

Postopek kodiranja:

1. znake razvrstimo po padajocih verjetnostih
2. znake razdelimo v  $r$  cim bolj enako verjetnih skupin
3. Vsaki skupini priredimo enega od  $r$  znakov kodne abecede
4. Deljenje ponovimo na vsaki od skupin. Postopek ponavljamo, dokler je mogoče

## 2.7 Huffmanov kod

Huffmanov postopek kodiranja poteka od spodaj navzgor (Pri Fanoju je ravno obratno). Pri huffmanovem kodu imamo dve fazi:

1. Združevanje
  - (a) Posici  $r$  najmanj verjetnih znakov in jih zdruzi v sestavljeni znak, katerega verjetnost je vsota verjetnosti vseh znakov

- (b) Preostale znake skupaj z novo sestavljenim znakom spet razvrsti
- (c) Postopek ponavlja dokler ne ostane samo  $r$  znakov

## 2. Razdruževanje

- (a) Vsakemu od preostalih znakov priredi po en znak kodirne abecede
- (b) Vsak sestavljeni znak razstavi in mu priredi po en znak kodirne abecede
- (c) Ko zmanjka sestavljenih znakov, je postopek zaključen

Pred kodiranjem, je vedno pametno preveriti, ce imamo zadostno stevilo znakov. Veljati mora:

$$n = r + k(r - 1), k \geq 0$$

Ce imamo premalo znakov, jih po potrebi dodamo s verjetnostjo  $p = 0$ .

Huffmanov kod lahko razsirimo tako, da vec osnovnih znakov združujemo v sestavljene znake  $\rightarrow$  bolj učinkoviti kodi. Vendar naletimo na nevarnost kombinacijske eksplozije.

## 2.8 Aritmetični kod

Je **hiter** in **blizu optimalnemu** kodu, ter manj učinkovit kot Huffmanov, vendar se izogne kombinacijski eksploziji. Vsak niz je predstavljen kot realno stevilo  $0 \leq R < 1$ , kar nam pove, da daljši kot bo niz, bolj natančno mora biti podano naravno stevilo  $R$ .

Postopek kodiranja (znakov ni potrebno razvrstiti):

1. Zacetemo z intervalom  $[0, 1)$
2. Izbrani interval razdelimo na  $n$  podintervalov, ki se ne prekrivajo. Sirine podintervalov ustrezajo verjetnostim znakov. Vsak podinterval predstavlja en znak
3. Izberemo podinterval, ki ustreza iskanemu znaku
4. Ce niz se ni koncan, izbrani podinterval ponovno razdelimo (bne 2.tocka)
5. Niz lahko predstavimo s poljubnim realnim stevilom v zadnjem podintervalu

Ko dobimo realni interval, ga samo se pretvorimo v binarnega s pomocjo klasicnega pretvarjanja iz dec v bin stevilski sistem.

2.9 Kod Lempel-Ziv (LZ77)

Stiskanje temelji na osnovi slovarja, tako, da ne potrebujemo racunati verjetnosti za posamezne znake. **Kodirnik** med branjem niza gradi slovar, in **dekodirnik** med branjem kodnih zaamenjav rekonstruira slovar in znake. **Kodiranje:** uporablja drseca okna, znaki se premikajo iz desne na levo. Referenca je podana kot trojcek:

- odmik - razdalja do zacetka enakega podniza v medpomnilniku
- dolzina enakega podniza
- naslednji znak

npr. (0, 0, A) - ni ujemanja, (4, 3, B) - 4 znake nazaj se ponovi 3 znakovni podniz, ki se nato zakljuci s B.

**dekodiranje:** sledimo kodnim zamenjavam

2.10 Deflate

Gre za predelan LZ77. Uporablja pare (odmik, dolzina). Ce ujemanja v kodni tabeli ni, zapise kar znak. Uporablja dve kodni tabeli:

- **tabela za znake in dolzine** - 285 simbolov (0-255 za osnovne znake, 256 konec bloka, 257-285 kodira dolzine) Kodne zamenjave brez dodatnih bitov, se zakodira s Huffmanom.

- **tabela odmikov**

Niz znakov se razdeli na bloke(64k) vsak blok se kodira na enega od treh nacinov:

1. **brez stiskanja** osnovni znaki se prepisejo
2. **stiskanje s staticnim Huffmanom** (verjetnosti podane vnaprej), Huffmanovo drevo ni zakodirano v bloku
3. **stiskanje s Huffmanom** izracunamo verjetnosti za vsak blok

Glava posameznega bloka: 1bit - zadnji/ni zadnji blok + 2bita tip stiskanja + pri (3) se Huffmanovo drevo Ker Huffmanovo drevo ni enolicno, uvedemo kanonicni Huffmanov kod. Postopek:

1. znake razvrstimo najprej po dolzinah kodnih zamenjav in nato po abecedi
2. prvi simbol ima same nicle
3. vsakemu naslednjemu znaku dodelimo naslednjo binarno kodo (prejsnji + 1)
4. ce je kodna zamenjava daljsa od bina-rne kode stevila, na koncu pripnemo nico
5. ponavljaj (3) do konca

Na taksen nacin dosežemo, da je potrebno kodirati samo dolzine kodnih zamenjav.

2.11 Kod Lempel-Ziv (LZW)

Osnovni slovar je podan in ga sporti dopon-jujemo. Alogritem za **kodiranje:**

```
N = ""
ponavljaj:
  preberi naslednji znak z
  ce je [N,z] v slovarju:
    N = [N, z]
  drugace:
    izpisi indeks k niza N
    dodaj [N, z] v slovar
    N = z
  izpisi indeks k niza N
```

Algoritem za **dekodiranje:**

```
preberi indeks k
poisci niz N, ki ustreza indeksu k
izpisi N
L = N
ponavljaj:
  preberi indeks k
  ce je k v slovarju:
    poisci niz N
  drugace:
    N = [L, L(1)]
  izpisi N
  v slovar dodaj [L, N(1)]
  L = N
```

LZW doseze optimalno stiskanje, približa se entropiji.

2.12 Verizno kodiranje ali RLE (run lenght encoding)

Namesto originalnih podatkov, sharnjujemo dolzino verige (fffeef → 3f2e1f). Problemu, ko se podatki ne ponavljajo, se izognemo tako, da izvedemo kombinacijo direktnega kodi-ranja in kodiranja RLE.

2.13 Stiskanje z izgubami

S taksnim nacinom stiskanja lahko dosežemo veliko boljša kompresijska razmerja, vendar izgubimo podatke. Zato ga uporabljamo samo s formati, kjer se ne ukvarjamo z integriteto podatkov(MP3, MPEG, JPEG, ...). Postopki kodiranj znanih formatov:

- **JPEG**
  1. priprava slike → ker je svetlost bol pomembna, je barvna resolucija obicajno zmanjsana ( $YC_R C_B$ )
  2. aproksimacija vsake od treh kom-ponent s 2D DCT
  3. kvantizacija → podatki ki bolj izstopajo so shranjeni manj natančno kot tisti ki so staticni
  4. kodiranje blokov s pomocjo en-tropije
  5. RLE cik-cak po sliki

6. RLE kodiramo z Huffmanom ali Aritmenticnim kodom

• **MP3**

1. Modified DCT
2. odstranitev za cloveka neslisnih frekvenc
3. stereo, ce sta si L in R pretvorimo v mono
4. Huffman na koncu

• **MPEG**

1. uvodno kodiranje → celotna slika JPEG
2. nato pa kodiramo samo spre-membe, ki so se zgodile v sliki JPEG s pomocjo vektorja pre-mika. V primeru, da je prevec raz-lik, se ponovno kodira JPEG slika.

2.14 Kompresijsko razmerje

Izracunamo ga po formuli → stisnjeni binarni zapis C(M) / binarni zapis dokumenta (M):

$$R = C(M)/M$$

3 Kanali

3.1 Uvod

Kanali so strukture, ki opisujejo medsebojno povezanost. Kanal prenasaa informacijo o spe-menljivki X do spremenljivke Y. Matemat-icno ga opisemo s **pogojnimi verjetnostmi**, ki povezujejo izhodne verjetnosti z vhodom.

3.2 Diskretni kanal brez spom-ina

Povezuje diskretni nakljuicni spremenljivki, s koncno mnozico stanj  $X = \{x_1, \dots, x_r\}$  in  $Y = \{y_1, \dots, y_s\}$ . Obema nakljicnima spre-menljivkama pripadajo tudi posamezne ver-jetnosti  $P_X = \{p(x_1), \dots, p(x_r)\}$  in  $P_Y = \{p(y_1), \dots, p(y_s)\}$ . Kjer velja, da je vsota posamezne mnozice verjetnosti enaka 1. Kanal je definiran kot mnozica **pogojnih verjetnosti**

$$p(y_j|x_i).$$

Pogojna verjetnost nam pove verjetnost za dogodek  $y_j$  na izhodu iz kanala, ce je na vhodu v kanal dogodek  $x_i$ . Brez spomina je zato, ker so pogojne vrjetnosti konstantne in torej neodvisne od prehodnih simbolov, velja

$$\sum_j p(y_j|x_i) = 1.$$

Kanal popolnoma podamo z  $r \times s$  pogojnimi verjetnostmi.

3.2.1 Binarni simetrični kanal (BSK)

Gre za poseben primer diskretnega kanala brez spomina. Napaka kanala je  $p$ , saj se z verjetnostjo  $p$  znak prenese v napacnega.

$$P_k = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

3.3 Pogojna entropija

Pogojna entropija spremenljivke  $Y$  pri znanem  $X$  se zapiše kot  $H(Y|X)$ . Vzemimo, da se je zgodil dogodek  $x_i \in X$ . Entropija dogodka  $Y$  je potem

$$H(Y|x_i) = - \sum_{j=1}^s p(y_j|x_i) \log(p(y_j|x_i)).$$

Velja:  $0 \leq H(Y|x_i)$ .  
Ce pa o dogodku  $X$  vemo le da se je zgodil, se lahko spomnemo na vis in uporabimo **vezano verjetnost** dogodkov  $X$  in  $Y$ , ki pravi:

$$p(x_i, y_j) = p(y_j|x_i)p(x_i)$$

Za entropijo:

$$\begin{aligned} H(Y|X) &= \sum_i p(x_i)H(Y|x_i) \\ &= - \sum_{i=1}^r \sum_{j=1}^s p(x_i, y_i) \log p(y_j|x_i) \end{aligned}$$

Splosno velja:  $0 \leq H(Y|X) \leq H(Y)$ , ce poznamo spremenljivko  $X$ , se nedolocenost  $Y$  ne more povecati (lahko se pomanjsa).

3.4 Vezana entropija spremeljivk

Vezana entropija nakljucnih spremenljivk  $X$  in  $Y$  je entropija para  $(X, Y)$ . Pomembne zveze:

- $p(x_i, y_j) = p(y_j|x_i)p(x_i)$ ,
- $\sum_j p(x_i, y_j) = p(x_i)$ ,
- $\sum_i p(x_i, y_j) = p(y_j)$ ,
- $\sum_{i,j} p(x_i, y_j) = 1$

Velja:  $H(X, Y) = H(Y|X) + H(X)$ , kar nam pove, da ce najprej izvemo, kaj se je zgodilo v dogodku  $X$  in potem dobimo se dodatne informacije od dogodku  $Y$ , vemo vse.

3.4.1 Obrat kanala

Ker velja tudi  $H(X, Y) = H(X|Y) + H(Y)$ , kanal lahko **obrnemo** (sepravi vhod  $Y$  in izhod  $X$ ). Pri tem ne obracamo fizicnega procesa, ampak samo verjetnostno strukturo, ki definira kanal. **Pogoj:** poznati moramo vhodne verjetnosti. Iz njih lahko dolocimo izhodne verjetnosti, ki jih lahko uporabimo kot vhodne verjetnosti v obrnjeni kanal. Lastnosti:

- izracun izhodnih verjetnosti  $p(y_j) = \sum_i p(y_j, x_i)p(x_i)$
- obratne pogojne vrjetnosti  $p(x_i, y_j) = p(y_j|x_i)p(x_i) = p(x_i|y_j)p(y_j)$

Za sprejemnika sporočila so obratne pogojne verjetnosti zelo pomembne, saj z njimi lahko iz prejetih znakov doloci verjetnost za vhodne znake.

3.5 Medesebojna informacija

Pove nam, koliko o eni spremenljivki izvemo iz druge spremenljivke, definicija:

$$I(X; Y) = H(X) - H(X|Y)$$

Lastnosti:

- $I(X; Y) = H(X, Y) - H(X|Y) - H(Y|X)$
- $I(X; Y) = H(X) - H(X|Y)$
- $I(X; Y) = H(Y) - H(Y|X)$
- $I(X; Y) = H(X) + H(Y) - H(X, Y)$
- $I(X; Y)$  je simetricna glede na  $X$  in  $Y$
- $I(X; Y) = - \sum_i \sum_j p(x_i, y_j) \log \frac{p(x_i)p(y_j)}{p(x_i, y_j)}$
- $I(X; Y) \geq 0$
- $I(X; X) = H(X)$

3.6 Kapaciteta kanala

Kapaciteta kanala je največja možna medsebojna informacija, ki jo lahko prenesemo od vhoda na izhod.

$$C = \max_{P(X)} I(X; Y)$$

3.6.1 Kapaciteta kanala BSK

Lastnosti:

- $C = \max_{P(X)} (H(Y) - H(Y|X))$
- $p(x_0) = \alpha, p(x_1) = 1 - \alpha$
- $I(X; Y) = H(Y) - H(Y|X) = \dots = H(Y) - H(p, 1 - p)$
- $\frac{dI(X; Y)}{d\alpha} = 0$
- $H(Y) = 1 \Rightarrow C$  je max
- $C = I(X; Y)|_{\alpha=1/2} = 1 - H(p, 1 - p)$

3.6.2 Kapaciteta kanala BSK z brisanjem

Definicija:

$$P_k = \begin{pmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{pmatrix}$$

Lastnosti:

- $C = 1 - p$
- $p(x_0) = \alpha, p(x_1) = 1 - \alpha$
- $p(y_0) = (1 - p)\alpha, p(y_1) = p, p(y_2) = (1 - p)(1 - \alpha)$
- $I(X; Y) = (1 - p)H(\alpha, 1 - \alpha)$
- $\frac{dI(X; Y)}{d\alpha} = 0 \Rightarrow \alpha = 1/2$

3.7 Shannonov drugi teorem

Shannon je ugotovil, da nam združevanje znakov v nize daje več možnosti za doseganje zanesljivega prenosa. Naj bo  $M$  število različnih kodnih zamenjav, ki jih lahko oblikujemo z nizi dolžine  $n$ . Potem je **hitrost koda** (prenosa) definirana kot:

$$R = \frac{\max H(X^n)}{n} = \frac{\log M}{n}$$

Hitrost je največja takrat, ko so dovoljene kodne zamenjave na vhodu enako verjetne. Shannonov teorem pravi, da je možna skoraj popolna komunikacija s hitrostjo, enako kapaciteti kanala. **Teorem:**

Za  $R \leq C$  obstaja kod, ki zagotavlja tako preverjanje informacije, da je verjetnost napake pri dekodiran poljubno majhna. Za  $R > C$  kod, ki bi omogočal preverjanje informacije s poljubno majhno verjetnostjo napake, **ne** obstaja.

Ce so znaki neodvisni, velja:

$$\log(H(X^n)) = n \log H(X) \Rightarrow R = H$$

Za  $R \leq \frac{\log 2^{nC}}{n} = C$  je možno najti kodne zamenjave, ki omogočajo zanesljivo komunikacijo.

4 Varno kodiranje

4.1 Uvod

Omejili se bomo na enostavne linearne bločne kode za BSK. Dolžina bloka je  $k$  znakov, abeceda je enaka abecedi kanala, torej imamo  $M = 2^k$  blokov  $x_1, \dots, x_k, x_i \in \{0, 1\}$ . Za potrebe varovanja dodamo se nekaj varnostnih znakov, celotna dolžina vsake od  $M$  kodnih zamenjav je potem  $n$ . Namesto enega posljemo  $n$  enakih znakov. Boljši pristop pa je, da naredimo kode, kjer se povečujeta  $n$  in  $k$  hitreje od razilke  $n - k$ .

4.2 Kontrolne vsote

Varnost komunikacije povečamo tako, da dodamo nekaj bitov za preverjanje parnosti (paritetni biti). Nastavljeni so tako, da je vsota bitov v aritmetiki po modulu 2 fiksna vrednost (0 ali 1).  
*Spomnimo se arsa*

+/-/XOR	0	1
0	0	1
1	1	0

 AND  $\sim \times$ .

npr. 00|0, 01|1, 10|1, 11|1 (detektiramo samo eno napako).

4.2.1 Pravokotni kodi

Zapišemo ga v obliki pravokotnika, gledamo sodost po vrsticah in po stolpcih.

1	0	1
0	1	1
0	1	0

4.2.2 Trikotni kodi

Vsota elementov v stolpcu in vrstici s paritetnim bitom vred mora biti soda. (ravno tako vsota paritetnih bitov)

1	0	1
0	0	
1		

4.3 Hammingova razdalja

Hammingova razdalja med kodnima zamenjava nam pove število znakov, na katerih se razlikujeta. Kodni zamenjavi sta enaki, ce je razdalja 0, razdalja med različnimi kodi mora biti vsaj 1, drugace je kod **singularen**. Razdalja je podana kot **minimalna** Hammingova razdalja med dvema kodnima zamenjavama. Število napak, ki jih kod zazna:

d ≥ e + 1 → e\_max = d - 1

d ≥ 2f + 1 → f\_max = ⌊(d-1)/2⌋

4.3.1 Hammingov pogoj

Za bloke dolzine n lahko zgradimo 2^n različnih kodnih zamenjav. Ce zelimo zagotoviti odpornost na napake, mora biti razdalja d > 1. Uporabni kodi imajo st. kodnih zamenjav M = 2^k < 2^n. Hammingov pogoj: da bi lahko dekodirali vse kodne zamenjave, pri katerih je prislo do e ali manj napak mora veljati:

M ≤ 2^n / Σ\_{i=0}^e (n choose i)

4.4 Linearni blocni kodi

Kode oznacimo kot dvojcek (n, k). n predstavlja število vseh bitov, k podatkovnih, n - k pa st. paritetnih. O linearnih blocnih kodih govorimo, kadar:

- je vsota vsakega para kodnih zamenjav spet kodna zamenjava.
- da produkt kodne zamenjave z 1 in 0 spet kodno zamenjavo.
- vedno obstaja kodna zamenjava s samimi ničlami

Oznacimo jih z L(n, k). **Hammingova razdalja** linearnega koda je enaka številu enic v kodni zamenjavi z najmanj enicami. Naj bodo podatkovni biti oznaceni kot z1, z2 in z3, varnostni pa kot s1, s2 in s3:

z1	z2	s3
z3	s2	
s1		

Potem vrednosti zlozimo v vektor, in opravimo kodno zamenjavo.

x̃ = (x1, x2, x3, x4, x5, x6) = (z1, z2, z3, s1, s2, s3)

Velja:

z1 + z2 + s1 = 0 = x1 + x2 + x4  
z3 + s2 + z2 = 0 = x2 + x3 + x5  
s3 + s3 + z1 = 0 = x1 + x3 + x6

4.4.1 Generatorska matrika

Generiranje kodne zamenjave lahko opisemo z generatorsko matriko.

x̃ = z̃G = [1 0 0 1 0 1; 0 1 0 1 1 0; 0 0 1 0 1 1]

V splošnem podatkovni vektor 1 × k množimo z generatorsko matriko k × n, da dobimo kodno zamenjavo 1 × n. Matrika mora imeti linearno neodvisne vrstice. Kod, cigar generatorska matrika ima to obliko, je **sistematicni kod** - prvih k znakov koda je enakih sporočilu (podatkovnim bitom), ostalih n - k znakov pa so paritetni biti.

Za diskretne kanale brez spomina jo vedno lahko zapišemo v obliki G = (Ik|A).

4.4.2 Matrika za preverjanje sodosti

Linearne enacbe lahko zapišemo z matriko za preverjanje sodosti:

H = [1 1 0 1 0 0; 0 1 1 0 1 0; 1 0 1 0 0 1]

Lastnosti:

- x̃H^T = 0
- GH^T = 0
- G = (Ik|A) ⇒ H = (A^T|In-k)
- vsota dveh kodnih zamenjav je nova kodna zamenjava.

4.5 Sindrom v kanalu

Predpostavimo da se med posiljanjem v kanalu zgodi napaka:

z → x = zG → err → y = x + e → s = yH^T

Napako pri prenosu preprosto ugotavljamo tako, da pogledamo, ce je s = 0. Vendar to nam ne garantira da pri prenosu ni prislo do napake. Sindrom izracunamo na naslednji nacin(vektor velikosti 1 × n - k):

yH^T = (x + e)H^T = eH^T = s

Ker je verjetnost za napako obicajno p << 1, je niz s t napakami veliko verjetnejši od niza s t + 1 napakami.

4.5.1 Standardna tabela

Imejmo ponavljalni kod (0|00) in (1|11). Sestavimo matriki G in H.

G = [1|11] in H = [1 1 0; 1 0 1]

Imamo 4 mozne sindrome: (00), (01), (10), (11). Na izhodu lahko dobimo 2^n = 8 različnih nizov.

Mozne nize na izhodu in njihove sindrome obicajno razvrstimo v std. tabelo:

sindrom	popravljalnik	
00	000	111
01	001	110
10	010	101
11	100	011

V isti vrstici so nizi, ki dajo enak sindrom. V prvi vrstici so vedno kodne zamenjave, ki imajo sindrom 0. Skrajno levo je vedno niz, ki ima najmanj enic, saj je najbolj verjeten. Imenujemo ga popravljalnik. Ostale nize dobimo tako, da popravljalnik pristevamo k kodnim zamenjavam v prvi vrsti. Popravljanje je sedaj enostavno: izracunamo sindrom, popravljalnik odstejemo(pristejemo) od prejetega niza.

4.6 Hammingov kod

Hammingovi kodi so družina linearnih blocnih kodov, ki lahko popravijo eno napako. Najlazuje jih predstavimo z matriko za preverjanje sodosti, v kateri so vsi stolpci nenicelni vektorji. Spadajo med **popolne kode** - sfere z radijem 1 okrog kodnih zamenjav ravno napolnijo prostor z 2^n točkami.

Kod z m varnostnimi biti ima kodne zamenjave dolzine 2^m - 1. Oznaka koda je potem H(2^m - 1, 2^m - 1 - m). Ce stolpce v matriki H interpretiramo kot števila v binarni obliki, nam oznaka stolpca doloca položaj napake. Stolpci v Hammingovem kodu so lahko poljubno razmetani. Pomembno je le to, da nastopajo **vs**a števila od 1 do 2^m - 1.

Hammingov kod je lahko:

- **leksikografski** - oznake stolpcev si sledijo po vrsti
- **sistematicni** - oznake stolpcev so pomesane

V Hammingovem kodu se za varnostne bite obicajno vzamejo tisti stolpci, ki imajo samo **eno** enico.

4.6.1 Dekodiranje

Dekodiranje leksikografskega Hammingovega koda je preprosto:

1. izracunamo sindrom s = yH^T
2. ce je s = 0, je x' = y
3. ce s ≠ 0, decimalno število S predstavlja mesto napake.

Za kod, ki pa ni leksikografski potrebujemo tabelo povezav med indeksi sindromov in stolpci(sepravi pogledamo, na kateri indeks se slika izracunani sindrom).

## 4.7 Ciklicni kodi

Ciklicni kod  $C(n, k)$  je linearni blocni kod, v katerem vsak krozni premik kodne zamenjave da drugo kodno zamenjavo. Zapišemo jih s polinomi po padajocih potencah (ravno tako jih sestevamo po mod 2).

### 4.7.1 Zapis s polinomi

Imejmo osnovni vektor:

$$\begin{aligned} x &= (x_{n-1}, x_{n-2}, \dots, x_0) \Leftrightarrow \\ x(p) &= x_{n-1}p^{n-1} + x_{n-2}p^{n-2} + \dots + x_0 \end{aligned}$$

Izvedemo premik za eno mesto:

$$\begin{aligned} x' &= (x_{n-2}, \dots, x_0, x_{n-1}) \Leftrightarrow \\ x'(p) &= x_{n-2}p^{n-2} + \dots + x_0p + x_{n-1} \end{aligned}$$

Velja zveza:  $x'(p) = px(p) - x_{n-1}(p^n - 1)$ .

V mod 2 aritmetiki:

$$\Rightarrow x'(p) = px(p) + x_{n-1}(p^n - 1).$$

V mod( $p^n + 1$ ) aritmetiki:

$$\Rightarrow x'(p) = px(p) \bmod (p^n + 1).$$

**Pozor:** aritmetiko po mod 2 izvajamo na **istih** stopnjah polinoma (na bitih), aritmetiko po mod ( $p^n + 1$ ) pa na **polinomu**.

Izvajanje kroznega prekmika za  $i$  mest:

$$x^i(p) = p^i x(p) \bmod (p^n + 1)$$

### 4.7.2 Generatorski polinomi

Vrstice generatorske matrike lahko razumemo kot kodne zamenjave. Za ciklicne kode v splošnem velja: **Generatorski polinom** je stopnje  $m$ , kjer je  $m$  stevilo varnostnih bitov, in ga označimo kot:

$$g(p) = p^m + g_{m-1}p^{m-1} + \dots + g_1p + 1$$

Za sistematični kod velja:  $G = [I_k | A_{k, n-k}]$ . Generatorska matrika:

$$G = \begin{bmatrix} 1 & g_{m-1} & \dots & g_1 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & g_{m-1} & \dots & g_1 & 1 & 0 & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 & g_{m-1} & \dots & g_1 & 1 \end{bmatrix}$$

Sistematicni lahko dobimo z linearnimi operacijami nad vrsticami. Velja:

$$p^n + 1 = g(p)h(p)$$

Sepravi vsak polinom, ki polinom  $p^n + 1$  deli brez ostanka, je generatorski polinom.

### 4.7.3 Polinom za preverjanje sodosti

Velja:  $x(p)h(p) \bmod (p^n + 1) = 0 \Rightarrow \sum_{i=0}^{n-i} x_i h_{j-i} = 0$

V matricni obliki:  $\vec{x}H^T = H\vec{x}^T = 0$

$$\begin{bmatrix} h_0 & \dots & h_k & 0 & \dots & 0 & 0 \\ 0 & h_0 & \dots & h_k & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & h_0 & \dots & h_k \end{bmatrix} \begin{bmatrix} x_{n-1} \\ \vdots \\ x_0 \end{bmatrix} = 0$$

### 4.7.4 Kodiranje z množenjem

Kodne zamenjave so večkratniki generatorskega polinoma. Velja:

$$x(p) = z(p)g(p) \bmod (p^n + 1)$$

, kjer je  $z(p)$  polinom, ki ustreza podatkovnemu vektorju  $\vec{z}$  Kod, ki smo ga dobili z množenjem, ustreza generatorski matriki, ki ima v vrsticah koeficiente  $p^{k-1}g(p), \dots, pg(p), g(p)$ , zato ni sistematičen.

### 4.7.5 kodiranje z deljenjem

Kodiranje na osnovi deljenja ustvari sistematičen ciklicen kod. Kodna zamenjava je zato sestavljena iz sporočila (podatkovnega bloka) in varnostnega bloka znakov,  $x = (z|r)$ . Polinom podatkovnega bloka je:

$$z(p) = z_{k-1}p^{n-1} + \dots + z_1p^1 + z_0p^0$$

Ce pa polinom pomnožimo s  $p^m$ , dobimo na desni  $m$  nicel.

$$p^m z(p) = z_{k-1}p^{k-1} + \dots + z_1p^{m+1} + z_0p^m$$

To ustreza bloku  $z$ , premaknjenem za  $m$  znakov v levo,  $(z_{k-1}, \dots, z_0, 0, \dots, 0)$ .

V splošnem nastavek seveda ne bo deljiv, velja pa  $p^m z(p) = g(p)t(p) + r(p)$ , kjer je  $t(p)$  kolicnik,  $r(p)$  pa ostanek, s stopnjo manj od  $m$ .

Ostanek lahko zapišemo v obliki niza, kot  $(0, \dots, 0, r_{m-1}, \dots, r_0)$ .

Polinom  $p^m z(p) + r(p) = g(p)t(p)$  je deljiv z  $g(p)$  in je zato ustrazna kodna zamenjava. Kodno zamenjavo tako dobimo, ce ostanek deljenja z generatorskim polinomom pristevamo k osnovnemu nastavku,  $(z_{k-1}, \dots, z_0 | r_{m-1}, \dots, r_0)$ .

### 4.7.6 Strojna izvedba kodirnika

Uporabljeni so trije tipi elementov: pomnilna celica tipa  $D$ , sestevalnik (XOR), množenje s konstanto  $(1 \mid 0)$ . Poznamo kodiranje na osnovi deljenja in na osnovi množenja. (insert pics here). Pri kodiranju se sepravi najprej na izhod posiljajo kar vhodni znaki, potem v naslednjih korakih se vsebina pomnilnih celic od zadaj naprej.

### 4.7.7 Dekodiranje

Dekodiranje ciklicnih kodov sloni na linearnih blocnih kodih. Vzemimo, da je pri prenosu prislo do napake  $y = x + e$ , ali pa zapisano v polinomski obliki  $y(p) = x(p) + e(p) = z(p)g(p) + e(p)$ .

- Najprej izracunamo sindrom. Ekvivalent enacbe  $s = yH^T$  v polinomskem zapisu je  $y(p) = q(p) * g(p) + s(p)$ , oz.  $s(p) = y(p) \bmod g(p)$ .

- Ce je ostanek deljenja  $y(p)$  z  $g(p)$  različen od nic, je prislo do napake.

Iz  $s(p) = y(p) \bmod g(p)$  sledi, da je v primeru, ko je napaka na zadnjih  $m$  mestih, stopnja  $e(p)$  manj kot  $m$  in velja kar  $e(p) = s(p)$ . Za ostale napake pa lahko izkoristimo ciklicnost kodov:

- Naredimo trik, osnovno enacbo premaknemo za  $i$  mest:

$$p^i y(p) = p^i x(p) + p^i e(p)$$

- Ce najdemo pravi  $i$ , bo veljalo  $p^i e(p) = s(p)$

- Pravi  $i$  je tisti, pri katerem bo  $e(p)$  imel najmanj enic

### 4.7.8 Klasifikacija napak

Napaki, ki se pojavi na izhodu odposlane kodne zamenjave neodvisno od morebitnih napak na sosednjih znakih, pravimo **posam-icna** ali **neodvisna** napaka. Do posamicnih napak pride zaradi motenj, ki so krajše od casa posiljanja enega znaka.

Povezanim napakam na vec zaporednih znakih pravimo **izbruh**. Dolžina izbruha je stevilo znakov med prvim in zadnjim napacno sprejetim znakom. Do izbruha pride, ce je trajanje motenj daljše od casa posiljanja enega znaka.

Ciklicni kodi so posebej primerni za **ugotavljanje izbruhov napak** .

### 4.7.9 Zmoznosti ciklicnih kodov

Odkrivanje napak s ciklicnimi kodi, kjer velja  $1 < \text{st}(g(p)) < n$ :

- Kod odkrije vsako posamicno napako:  $e(p) = p^i$

- Za dolocene generatorske polinome odkrije tudi dve posamicni napaki do dolzine bloka  $n = 2^m - 1$

- Odkrije poljubno stevilo lihih napak, ce  $p + 1$  deli  $g(p)$

- Odkrije vsak izbruh napak do dolzine  $m$

- Odkrije vse razen  $2^{-(m-1)}$  izbruhov dolzine  $m + 1$

- Odkrije tudi vse razen delez  $2^{-m}$  izbruhov daljsih od  $m + 1$

Popravljanje napak s ciklicnimi kodi, kjer velja  $1 < \text{st}(g(p)) < n$ :

- Izracun sindroma

- Ciklicno prilaganje sindroma prenesenemu blok  $y$ .

- Popravijo lahko do  $e = \lfloor \frac{d-1}{2} \rfloor$  posamicnih napak, kjer je  $d$  Hammingova razdalja koda.

- Popravijo lahko tudi izbruhe napak do dolzine  $e = \lfloor \frac{m}{2} \rfloor$

#### 4.7.10 CRC

Ali Cyclic Redundancy Check, temelji na ciklikih kodih. Po standardu velja:

- Registri v **LSFR** so na začetku nastavljeni na 1; osnovni CRC ne loci sporocil, ki imajo različno število vodilnih ničel. Ta sprememba, ki je ekvivalentna negiranjju prvih  $m$  bitov, to težavo odpravi.
- Na koncu sporočila dodamo  $m$  - bitov, odvisno od implementacije LSFR. Pri nasi se to ne dela!
- **Operacija XOR** na fiksnem ostanku deljenja, običajno je to kar negacija vseh bitov.
- **Vrstni red bitov v bajtu** - nekateri serijski protokoli najprej oddajo najmanj pomembne bite (najmanj pomembni bit ima najvisjo stopnjo polinoma).
- **Vrstni red bajtov** - pomnilniška organizacija, odvisna od arhitekture (LE, BE).
- Notacija CRC polinomov - biti označujejo prisotnost faktorja. Večkrat se izpusca en izmed faktorjev  $p^m$  ali 1.

Ciklični kodi so odlični za detekcijo napak. Za popravljanje napak pa danes obstajajo boljši kodi.

#### 4.7.11 Prepletanje

Motnje so mnogokrat v obliki izbruhov. V takih primerih pride na dolocenih kodnih zamenjavah do velikega števila napak, na drugih pa napak ni. S prepletanjem bitov se da napake porazdeliti med več kodnih zamenjav. Resitev:

- Kodne zamenjave v kodirnik vpisujemo vrstico po vrstico, oddaja pa jih stolpec po stolpec. Obratno je na strani dekodirnika.
- Naceloma je vzorec skoraj naključen. Matriko prepletanja poznata kodirnik in dekodirnik.

- Dodamo zakasnitev, izmenično signali potujejo gor/dol, ena veja je zakasnjena.

Dejanske resitve so bolj kompleksne: več vej, zakasnitve tudi do 20 vej.

#### 4.7.12 Konvolucijski kodi

Primerni za popravljanje napak. Konvolucijske kode generiramo z linearnimi premikalnimi registri, ki so sestavljeni iz pomnilnih celic D in vrat XOR. Spadajo pod nelinearne kode.

## 5 Analiza signalov

Pri analizi signalov in sistemov je izjemno pomembna kolicina frekvenca.

### 5.1 Invariantnost sinusoid

Vzemimo zvezni signal, ki prehaja skozi linearni medij (sistem) kot je na primer električno vezje.

V splošnem bo signal na izhodu drugačen od signala na vhodu (zvok, ki ga poslušamo pod vodo je bistveno bolj popačen od tistega, ki ga poslušamo na zraku)

Pomembno pri signalih pa je, da se vhodni signal v obliki sinusoid

$$x(t) = A \sin(2\pi\nu t + \theta)$$

popaci v izhodni signal z drugačno amplitudo in fazo  $\theta$ , vendar ohrani frekvenco  $\nu$ . Razlog, da se frekvenca ohrani je v tem, da linearne sisteme lahko zapisemo v obliki elementarnih operacij, kot so (množenje s konstanto, odvajanje, integracija, zakasnitev, vsota).

### 5.2 Fourierova transformacija

Vsako periodično funkcijo (ce je dovolj lepa), lahko zapisemo kot kombinacijo sinusoid. V kombinaciji z invariantnostjo sinusoid to pomeni, da lahko:

- vsako funkcijo razstavimo na sinusoid

- obravnavamo obnasanje vsake sinusoid v sistemu posebej

- na koncu združimo locene rezultate

Ta koncept se danes uporablja pri vsaki analizi signalov.

## 5.3 Resonanca

Do resonance pride, ko je frekvenca vsiljenega nihanja enaka frekvenci lastnega nihanja. Takrat pride do ojačitve amplitud. Resonanca je pomembna lastnost električnih vezij, s katero zagotovimo nihanja, nastavljanje radijskih sprejemnikov na pravo postajo, odstranimo sum.

### 5.4 Modulacija in frekvenčni premik

Iz analize vemo, da nelinearne operacije nad signali (kvadriranje, množenje) privedejo do pomembnih transformacij v frekvenčnem prostoru.

Iz osnovne trigonometrije vemo:

$$\begin{aligned} \sin(2\pi\nu_1 t) \sin(2\pi\nu_2 t) &= \\ \frac{1}{2} [\cos(2\pi(\nu_1 - \nu_2)t) - \cos(2\pi(\nu_1 + \nu_2)t)] \\ \cos(2\pi\nu t) &= \sin(2\pi\nu t + \pi/2) \end{aligned}$$

Produkt sinusoid s frekvencama  $\nu_1$  in  $\nu_2$  lahko torej zapisemo kot vsoto sinusoid s frekvenco  $\nu_1 + \nu_2$  in sinusoid s frekvenco  $\nu_1 - \nu_2$ .

To lastnost izkorisca amplitudna modulacija (radijske postaje AM) in frekvenčni premik, s katerim lahko zagotovimo hkraten prenos več signalov po istem mediju.

### 5.5 Teorem vzorčenja

Signal moramo vzorčiti vsaj s frekvenco  $2\nu_c$ , ce je najvisja opazena frekvenca v signalu  $\nu_c$ . Na tem zaključku sloni vsa danasnja tehnologija.