Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
Bolyai Intézet
Geometria Tanszék

Diplomamunka

Véletlen lineáris kódok hibajavító rátáiról

Mezőfi Dávid Csaba
alkalmazott matematikus hallgató

Szeged, 2016. május 13.

Témavezető: Dr. Nagy Gábor Péter, egyetemi docens

Tartalomjegyzék

Ábrák jegyzéke

Jelölések

$\mathcal{B}(p)$ Bernoulli-eloszlás p paraméterrel.

$\mathcal{B}_n(p)$ Binomiális eloszlás (n, p) paraméterrel.

$\lceil x \rceil$ Az x valós szám felső egészrésze.

$\lfloor x \rfloor$ Az x valós szám alsó egészrésze.

$\operatorname{span} S$ Az S halmazbeli vektorok által kifeszített altér.

$\mathbb{F}_q$ A q-elemű véges test.

$\mathbb{F}_q^n$ A q elemű test feletti n-dimenziós vektortér.

$\mathbb{N}^+$ A pozitív egész számok halmaza: $\mathbb{N}^+ = \{1, 2, 3, \ldots\}$.

$\mathbb{N}_0$ A természetes számok halmaza beleértve a 0-t is: $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$.

PER Egy kód csomaghiba-valószínűsége (az angol *packet error rate* elnevezésből).

$\operatorname{rank}(\mathbf{A})$ Az $\mathbf{A}$ mátrix rangja.

$d(\mathcal{C})$ A $\mathcal{C}$ kód minimumtávolsága.

d A Hamming-távolság (vagy d_H, ha nem derül ki egyértelműen a szövegkörnyezetből).

$h(p)$ A Shannon-függvény (vagy bináris entrópiafüggvény).

$Q(x)$ Annak a valószínűsége, hogy egy standard normális eloszlású véletlen változó x-nél nagyobb értéket vesz fel, azaz $Q(x) = 1 - \Phi(x)$.

BSC A bináris szimmetrikus csatorna rövidítése (az angol *binary symmetric channel* elnevezésből).

NN A legközelebbiszomszéd-dekódolás rövidítése (az angol *nearest neighbour decoding* elnevezésből).

Bevezetés

A diplomamunka a kódoláselmélet témaköréhez kapcsolódik. A kódoláselmélet a következő kommunikációs modellel foglalkozik: egy küldő valamilyen csatornán adatokat akar továbbítani egy fogadónak. A továbbítás során az adatok egy része esetleg megváltozik. Hogyan tudja a fogadó kiválasztani, hogy mely adatok változtak meg, és hogyan tudja a megváltozott adatokat kijavítani?

A küldő összesen 3000 bitnyi információt szeretne továbbítani egy magas zajjal rendelkező bináris szimmetrikus csatornán úgy, hogy átlagosan öt alkalomból legalább négyszer minden fogadott vektort jól javítsunk legközelebbiszomszéd-dekódolással. Célunk olyan hatékony – gazdaságos és megbízható – kód keresése, mely teljesíti a fenti követelményeket.

A dolgozat első fejezete az információelmélet és a csatornakódolás alapvető ismereteit tartalmazza – az entrópia, a csatornakapacitás fogalmának ismerete szükséges a későbbiek megértéséhez –, valamint bemutatásra kerül a bináris szimmetrikus csatorna, illetve egy egyszerű hibajavító kód. A második fejezet első és második szakasza a kódoláselmélet alapvető fogalmait és állításait – kód, minimumtávolság, hibajavító képesség és ezek kapcsolata, linearitás – tartalmazza, melyek elengedhetetlenek a harmadik és negyedik fejezetben tárgyaltakhoz. A második fejezet harmadik szakaszában a legközelebbiszomszéd- és a maximum likelihood dekódolás kapcsolatát vizsgáljuk. A harmadik fejezetben a zajos csatornában fellépő hibák javításának különböző korlátait mutatjuk be, definiáljuk, mit értünk véletlen adott sűrűségű paritásellenőrző mátrixú bináris lineáris kód alatt, majd a legközelebbiszomszéd-dekódolást módosítjuk bitműveletek alkalmazásával a hatékonyság növelésének érdekében. A negyedik fejezetben a definiált véletlen kódok hibajavító rátáit becsüljük szimulációk segítségével, illetve megvizsgáljuk a paritásellenőrző mátrix sűrűsége és a hibajavító ráta, valamint a minimumtávolság közötti összefüggést. A függelék az ismertetett algoritmusoknak, szimulációknak egy használható **SageMath**-beli [S+16] implementációját mutatja be.

1 Információelmélet

Ebben a fejezetben André Neubauer, Jürgen Freudenberger és Volker Kühn *Coding Theory: Algorithms, Architectures and Applications* [CT] című könyve alapján tömör bevezetést kívánunk csatornakódolás információelméleti alapjaiba. Egy egyszerű csatornamodellt is leírunk, amelyet a későbbiekben használni fogunk. Végül pedig a bináris 3-ismétlő kódot mutatjuk be az egyszerű (csatorna)kódolás illusztrációjaként.

Az információ zajos csatornán keresztül történő megbízható átvitele a digitális információs és kommunikációs rendszerek egyik alapvető követelménye. Átvitel alatt most mind térbeli átvitelt – például rádióhullámok segítségével történő kommunikációt –, mind időbeli átvitelt – információ tárolását egy megfelelő tároló eszközön – értünk. A fenti követelmény teljesítésének céljából a modern kommunikációs rendszerek nagy mértékben hagyatkoznak hatékony csatornakódoló módszerekre. A gyakorlati alkalmazások során a kódolási sémáknak nem elég megfelelő karakterisztikával rendelkezniük – tekintettel a csatornában fellépő hibák jelzésére és javítására –, hanem digitális eszközökön hatékonyan implementálhatónak is kell lenniük. A kódolás gyakorlati alkalmazásai közé tartozik az űr- és szatellitkommunikáció, az adatátvitel, a digitális hang- és videóátvitel, a mobil kommunikáció és az adattárolás is – például a számítógép memóriája, a kompakt lemezek, CD-k.

1.1. Entrópia

Az információelmélet egy fontos eredménye, hogy a hibamentes adattovábbítás egy zajos csatornán keresztül elméletileg lehetséges egészen addig, amíg az információ rátája nem haladja meg az úgynevezett csatornakapacitást. Ezen eredmény bemutatásához azonban valamilyen módon mérnünk kell az információt. A Shannon-féle információelméletben ez az információforrás által kibocsátott szimbólumok statisztikai leírásával történik.

Vegyünk egy diszkrét, memóriátlan *információforrást*. Egy adott időpillanatban a forrás egy véletlen diszkrét $X = x_i$ szimbólumot bocsát ki M darab $x_1, x_2, \ldots, x_M$ lehetséges szimbólum közül. A $P_X(x_1), P_X(x_2), \ldots, P_X(x_M)$ valószínűségek határozzák meg,

hogy ezek a szimbólumok milyen arányban fordulnak elő:

$$P_X(x_i) = \mathbb{P}(X = x_i).$$

1.1.1. definíció (entrópia)**.** Az X diszkrét véletlen változóhoz tartozó átlagos információt az

$$I(X) = -\sum_{i=1}^{M} P_X(x_i) \log_2 (P_X(x_i))$$

úgynevezett *entrópia* adja meg, amelyet *bit* egységben mérünk.

1.1.2. definíció (bináris entrópiafüggvény)**.** Egy bináris információforrás esetén, amely az $X = 0$ és az $X = 1$ bináris szimbólumokat bocsátja ki $\mathbb{P}(X = 0) = p$ és $\mathbb{P}(X = 1) = 1 - \mathbb{P}(X = 0) = 1 - p$ valószínűségekkel, az entrópiát az úgynevezett *Shannon-függvény* (vagy bináris entrópiafüggvény) adja meg:

$$h(p) = I(X) = -p \log_2 p - (1-p) \log_2 (1-p).$$

1.2. Csatornakapacitás

Az entrópia fogalmának segítségével lehetőségünk nyílik a csatorna bemenetének és kimenetének kapcsolatának vizsgálatára. Jelölje X a bemeneti (vagy küldött) szimbólumot, míg R a kimeneti (vagy fogadott) szimbólumot. Tegyük fel, hogy M darab $x_1, x_2, \ldots, x_M$ bemeneti és N darab $r_1, r_2, \ldots, r_N$ kimeneti szimbólum lehetséges. A

$$P_{X|R}(x_i \mid r_j) = \mathbb{P}(X = x_i \mid R = r_j);$$
$$P_{R|X}(r_j \mid x_i) = \mathbb{P}(R = r_j \mid X = x_i)$$

feltételes valószínűségek segítségével felírhatjuk a feltételes entrópiákat:

$$I(X \mid R) = -\sum_{i=1}^{M} \sum_{j=1}^{N} \mathbb{P}(X = x_i, R = r_j) \cdot \log_2 \left(P_{X|R}(x_i \mid r_j) \right);$$
$$I(R \mid X) = -\sum_{i=1}^{M} \sum_{j=1}^{N} \mathbb{P}(X = x_i, R = r_j) \cdot \log_2 \left(P_{R|X}(r_j \mid x_i) \right).$$

1.2.1. definíció (kölcsönös információ)**.** A *kölcsönös információ*

$$I(X;R) = I(X) - I(X \mid R) = I(R) - I(R \mid X)$$

a bemenettől a kimenetig a csatornán átküldött információ mennyiségét méri egy adott információforrás esetén.

1.2.2. definíció (csatornakapacitás)**.** Az úgynevezett C *csatornakapacitás* az $I(X;R)$ kölcsönös információ maximuma az X bemenet statisztikai tulajdonságain, azaz a kölcsönös információ egy megfelelően megválasztott $\{P_X(x_i)\}_{i=1}^{M}$ valószínűségeloszlás mellett, tehát

$$C = \max_{\{P_X(x_i)\}_{i=1}^{M}} I(X;R).$$

Ha a bemeneti entrópia $I(X)$ kisebb, mint a csatorna C kapacitása,

$$I(X) < C,$$

akkor az információ tetszőlegesen kicsi hiba mellett továbbítható a zajos csatornán keresztül.

Megjegyzés. Tehát a C csatornakapacitás gyakorlatilag a csatorna információtovábbítási kapacitását méri.

1.3. Bináris szimmetrikus csatorna

A memóriátlan csatornák egy fontos példája a *bináris szimmetrikus csatorna* vagy röviden BSC (*binary symmetric channel*). Ez a csatorna az $X = 0$ vagy az $X = 1$ bináris szimbólumokat továbbítja hiba nélkül $1 - p$ valószínűséggel $(0 < p < 1)$, míg a hibás $R = 1$ vagy $R = 0$ szimbólumokat p valószínűséggel bocsátja ki (lásd az 1.1. ábrát).

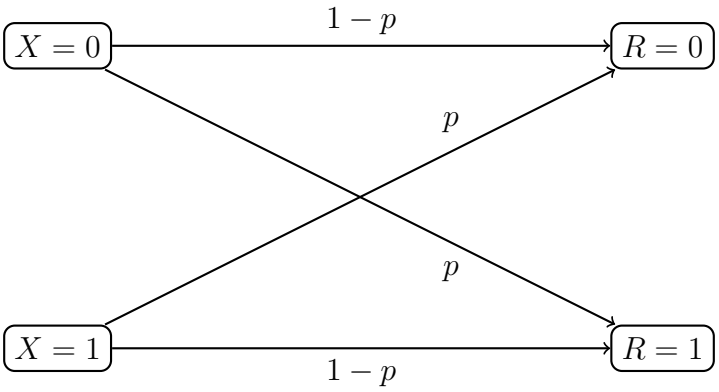
Megjegyzés. A csatorna memóriátlansága azt jelenti, hogy az egyes szimbólumok (hibás) átvitelei független események.

Az $I(X;R)$ kölcsönös információt maximalizálva kapjuk, hogy a bináris szimmetrikus csatorna kapacitása

$$C = 1 - h(p) = 1 + p\log_2 p + (1-p)\log_2(1-p). \tag{1.1}$$

Ez a csatornakapacitás 1-hez tart, ha p tart 0-hoz vagy 1-hez, míg $p = \frac{1}{2}$ esetén a csatorna kapacitása 0.

Megjegyzés. A bináris szimmetrikus csatornával ellentétben, ami diszkrét bemeneti és kimeneti szimbólumokkal operál egy bináris ábécé felett, az úgynevezett AWGN-csatorna



1.1. ábra. Bináris szimmetrikus csatorna p csatornaparaméterrel

(*additive white Gaussian noise*) definíciója folytonos valós értékű véletlen változókon alapszik.

1.4. Egy egyszerű kód

Egy egyszerű kód bevezető példájaként tekintsük a következőt: a

$$0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0$$

sorozatot küldjük át egy $p = 0{,}25$ paraméterű bináris szimmetrikus csatornán, így átlagosan minden negyedik bit hibásan kerül továbbításra. Tegyük fel, hogy a

$$0 \quad 0 \quad \mathbf{0} \quad 0 \quad \mathbf{0} \quad 1 \quad 1 \quad 0$$

bináris sorozatot kapjuk meg a csatorna fogadó oldalán (lásd az 1.2. ábrát).

Egy egyszerű hibajavító (dekódoló) séma implementálásához az úgynevezett bináris *3-ismétlő kódot* alkalmazzuk. Ez az egyszerű kód alkalmas bináris adat kódolására: ha a 0 bináris szimbólumot küldenénk, akkor a „kódoló egység" a 000 kódszóra kódolja szimbólumot, az 1 szimbólum esetén pedig hasonlóan az 111 kódszóra kódol (lásd az 1.3. ábrát).

Így a fenti küldendő bináris sorozatból a

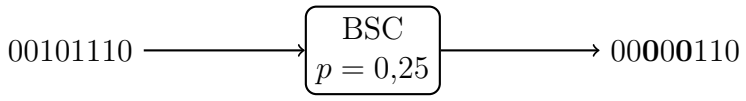
$$000 \quad 000 \quad 111 \quad 000 \quad 111 \quad 111 \quad 111 \quad 000$$

kódszavakból álló bináris sorozatot kapjuk a kódolást követően. Tegyük fel, hogy a bináris szimmetrikus csatornában átlagosan minden negyedik szimbólum hibás továbbítása ($p = 0{,}25$) mellett ebből a bitsorozatból a

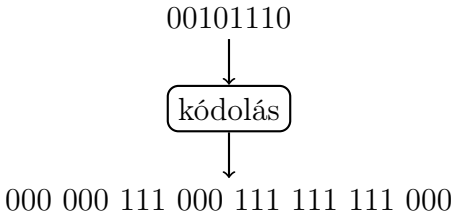
$$0\mathbf{1}0 \quad 000 \quad \mathbf{0}11 \quad 0\mathbf{1}0 \quad 111 \quad \mathbf{0}\mathbf{1}\mathbf{0} \quad 111 \quad 0\mathbf{1}0$$

sorozatot kapjuk.

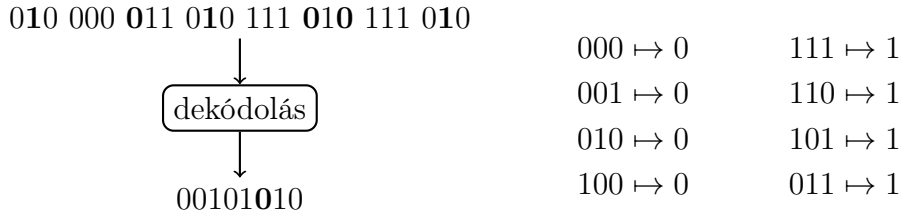
Az 1.4. ábrán bemutatott dekódolási séma az eredeti bitsorozatot közelíti a *többségi döntés* segítségével. Ha a kapott 3-bites kódszóban a 0-k száma nagyobb, mint az 1-eseké, akkor a „dekódoló egység" a 0 szimbólumot adja vissza, különben pedig 1-re dekódolja



1.2. ábra. Kódolás nélküli átvitel BSC-n



1.3. ábra. A 3-ismétlő kód kódolási sémája



1.4. ábra. A 3-ismétlő kód dekódolása: a *többségi döntés*

a kapott szót. Ez a dekódolási algoritmus a kapott

$$0\mathbf{1}0 \quad 000 \quad \mathbf{0}11 \quad 0\mathbf{1}0 \quad 111 \quad \mathbf{0}1\mathbf{0} \quad 111 \quad 0\mathbf{1}0$$

bitsorozatot a

$$0 \quad 0 \quad 1 \quad 0 \quad 1 \quad \mathbf{0} \quad 1 \quad 0$$

sorozatra dekódolja.

Ahogy a példa is mutatja, a 3-ismétlő kód egyetlen hibát képes kijavítani egy kódszóban, többet nem. Ezzel az egyszerű kódolással képesek vagyunk a hibák számát csökkenteni: a kódolás nélküli átvitelhez képest 1-gyel csökkenteni tudtuk (2-ről 1-re) a fellépő hibák számát. Azonban ezt csak a megfelelő áron érhettük el: a 3-ismétlő kód alkalmazásával egy információs szimbólum átvitele a csatornán 3-szor hosszabb időt vesz igénybe.

2 Lineáris blokk-kódok

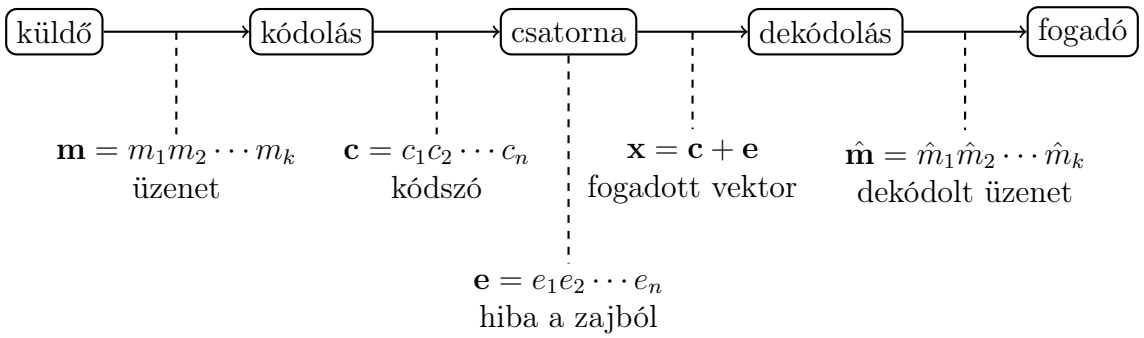
Az ebben a fejezetben ismertetett fogalmakat, állításokat és tételeket Petteri Kaski és Patric R. J. Östergard *Classification Algorithms for Codes and Designs* [CACD], W. Cary Huffman és Vera Pless *Fundamentals of Error-Correcting Codes* [FoECC], Kiss György és Szőnyi Tamás *Véges geometriák* [VG] című könyve, valamint [CT] alapján ismertetjük.

A kódoláselmélet a következő kommunikációs modellel (lásd a 2.1. ábrát) foglalkozik. Egy *küldő* valamilyen *csatornán* adatokat akar továbbítani egy *fogadónak*, jelölje ezt az üzenetet $\mathbf{m}$ (az modellben használt szimbólumok általában az $\mathbb{F}_q$ véges testbeli elemek, sőt gyakran csak a kételemű testet – $\mathbb{F}_2$-t – alkalmazzuk). Amint azt az 1.4. szakaszban is láttuk, ha változtatás nélkül továbbítanánk $\mathbf{m}$-et a csatornán keresztül, a zaj miatt sérülhetne, és a sérülés következtében a fogadó nem tudná helyreállítani az üzenetet.

Az alapvető ötlet az, hogy az eredeti üzenetet redundánssá téve a továbbítás során fellépő hibák javíthatók lesznek. A redundaciát a *kódolás* biztosítja: a kódolást követően $\mathbf{m}$ helyett $\mathbf{c}$ kerül küldésre a csatornán keresztül. Csak az olyan hibák javításával foglalkozunk, melyek a szimbólumok megváltozásából adódnak, nem foglalkozunk szimbólumok hozzáadódásából és elhagyásából keletkező hibákkal. A probléma tehát a következő: a továbbítás során $\mathbf{x}$-hez hozzáadódik egy $\mathbf{e}$ hibavektor, s így a $\mathbf{x} = \mathbf{c} + \mathbf{e}$ módosult vektor kerül dekódolásra. A *dekódolás* során a fogadott $\mathbf{x}$ vektor alapján közelítjük az eredeti üzenetet: a dekódolt üzenet $\hat{\mathbf{m}}$, és remélhetőleg $\hat{\mathbf{m}} = \mathbf{m}$.

Az üzenetek és a kódszavak között bijekció áll fenn, ezért a dekódolás alatt többnyire a $\mathbf{c}$ kódszó $\mathbf{x}$ alapján történő közelítését (nota bene: kódszóra dekódolunk, jelölje ezt $\hat{\mathbf{c}}$) értjük, és remélhetőleg $\hat{\mathbf{c}} = \mathbf{c}$.

Ha egy kódot alkalmazni kívánunk, akkor a kódolás általában meglehetősen kézenfekvő, azonban a hatékony dekódolás legtöbbször sokkal nehezebb feladatnak bizonyul: ezt a fejezet későbbi szakaszában is látni fogjuk.



2.1. ábra. A kommunikációs modell

2.1. Kódoláselméleti bevezető

A kódoláselmélet története mérnöki alkalmazásokban gyökerezik, azonban hamar tiszta matematikai érdeklődési területté nőtte ki magát. Különösen, mivel a kódoláselmélet több alapvető matematikai problémája szorosan kapcsolódik bizonyos kombinatorikai objektumok konstrukciójához.

Legyen $n \in \mathbb{N}^+$ tetszőleges rögzített pozitív egész, és legyen $\mathbf{x}$ az $\mathbb{F}_q^n$ vektortér egy tetszőleges eleme. Kódelméleti kontextusban az $\mathbf{x}$ vektort gyakran *szónak* nevezzük. Az $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ komponenseit *koordinátáknak* nevezzük, míg az $x_i \in \mathbb{F}_q$ elemeket ($i \in \{1, 2, \ldots, n\}$) *koordinátaértékeknek* nevezzük. Ha nem okoz félreértést, akkor az $\mathbf{x}$ szót gyakran csak $x_1 x_2 \cdots x_n$-nel jelöljük.

Megjegyzés. Kódoláselméleti kontextusban szokásosan sorvektorokkal dolgozunk.

Hamming-távolság

A *távolság* fogalma központi szerepet tölt be a kódoláselmélet területén, ezért célszerű bevezetni egy (lehetőség szerint egyszerű) távolságfogalmat az $\mathbb{F}_q^n$ vektortéren. A következő lemma bizonyítása triviális.

2.1.1. lemma. *Tekintsük a*

$$d_H \colon \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N}_0, \quad (\mathbf{x}, \mathbf{y}) \mapsto |\{i \in \{1, 2, \ldots, n\} : x_i \neq y_i\}| \tag{2.1}$$

leképezést. A (2.1)-ben definiált d_H leképezés metrika az $\mathbb{F}_q^n$ vektortéren.

2.1.2. definíció (Hamming-távolság)**.** A (2.1)-ben definiált d_H metrikát *Hamming-távolságnak* nevezzük, és ha nem okoz félreértést, csak d-vel jelöljük.

Megjegyzés. Két vektor (szó) Hamming-távolsága nem más, mint azon koordináták száma, ahol a két vektor különbözik.

Legyen $\mathbf{c} \in \mathbb{F}_q^n$, $r \in \mathbb{N}_0$. Jelölje

$$\mathbb{B}_r(\mathbf{c}) = \left\{\mathbf{x} \in \mathbb{F}_q^n \colon d(\mathbf{c}, \mathbf{x}) \leq r\right\}$$

a $\mathbf{c}$ középpontú r-sugarú gömböt, $\mathbf{0}$ pedig a csupa nulla vektort $\mathbb{F}_q^n$-ben.

2.1.3. definíció (súly)**.** Legyen $\mathbf{x} \in \mathbb{F}_q^n$. Ekkor az $\mathbf{x}$ vektor *súlya* (vagy Hamming-súlya)

$$\mathrm{wt}(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}). \tag{2.2}$$

Megjegyzés. A wt súlyfüggvény az argumentumbeli vektorhoz – (2.2) alapján – éppen annak nem nulla koordinátáinak számát rendeli hozzá, innen a súly elnevezés.

Kód, minimumtávolság, ráta

2.1.4. definíció (kód)**.** Az $\mathbb{F}_q^n$ vektortérnek egy M-elemű $\mathcal{C}$ részhalmazát $\mathbb{F}_q$ feletti (n, M)-*kódnak* nevezzük: $\mathcal{C}$ elemei a *kódszavak*, a kód *hossza* n, a kód *mérete* M.

Mivel az üzenetek és a kódszavak között bijekció szükséges, ezért egy $\mathcal{C}$ (n, M)-kóddal nyilván M különböző üzenetet vagyunk képesek kódolni. Ha $\mathcal{M}$ az üzenetek halmaza, ahol $|\mathcal{M}| = M$, akkor a kódolás egy bijektív $E \colon \mathcal{M} \to \mathcal{C}$ leképezés (E mint az angol *encoding* kifejezés, azaz kódolás).

Megjegyzés. Ha $\mathcal{C}$ egy $\mathbb{F}_2$ feletti kód, akkor bináris kódnak nevezzük.

Egy kód tulajdonságai közül az egyik legfontosabb a minimumtávolság.

2.1.5. definíció (minimumtávolság)**.** Legyen $\mathcal{C} \subseteq \mathbb{F}_q^n$ egy legalább két kódszót tartalmazó kód, azaz $2 \leq |\mathcal{C}|$. Ekkor $\mathcal{C}$ *minimumtávolsága*

$$d(\mathcal{C}) = \min\left\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\right\}.$$

Megjegyzés. Az $|\mathcal{C}| = 1$ esetben szokás a $d(\mathcal{C}) = \infty$ konvenciót alkalmazni.

2.1.6. definíció (ráta)**.** Legyen $\mathcal{C}$ egy $\mathbb{F}_q$ feletti (n, M)-kód. Ekkor a $\mathcal{C}$ kód *rátája*

$$R = \frac{\log_q M}{n}.$$

Példa. Az 1.4. szakaszban bemutatott 3-ismétlő kód esetén $q = 2$, $n = 3$, $M = 2$, hiszen csak 000 és az 111 kódszavak alkották a kódot, így a ráta

$$R = \frac{\log_2 2}{3} = \frac{1}{3}.$$

Hibajavítás: legközelebbiszomszéd-dekódolás

2.1.7. definíció. Legyen $t \in \mathbb{N}^+$ pozitív egész szám, $\mathcal{C} \subseteq \mathbb{F}_q^n$ legalább kételemű kód. A $\mathcal{C}$ kódot *t-hibajavító kódnak* nevezzük, ha bármely két különböző $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ kódszó esetén

$$d(\mathbf{x}, \mathbf{y}) \geq 2t + 1.$$

Megjegyzés. Azaz, ha $\mathcal{C}$ minimum távolsága $d(\mathcal{C})$, akkor $\mathcal{C}$ egy t-hibajavító kód, ahol

$$t = \left\lfloor \frac{d(\mathcal{C}) - 1}{2} \right\rfloor.$$

A következő lemma és következménye megvilágítja a t-hibajavító kód elnevezést.

2.1.8. lemma. *Ha a $\mathcal{C} \subseteq \mathbb{F}_q^n$ kód minimumtávolsága $d(\mathcal{C})$ és $t = \left\lfloor \frac{d(\mathcal{C})-1}{2} \right\rfloor$, akkor a különböző kódszavak köré írt t-sugarú gömbök diszjunktak.*

Bizonyítás. Indirekt bizonyítunk: tegyük fel, hogy $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ olyan különböző kódszavak, melyekre létezik $\mathbf{z} \in \mathbb{B}_t(\mathbf{c}_1) \cap \mathbb{B}_t(\mathbf{c}_2)$. Ekkor a háromszög-egyenlőtlenség szerint

$$d(\mathbf{c}_1, \mathbf{c}_2) \leq d(\mathbf{c}_1, \mathbf{z}) + d(\mathbf{z}, \mathbf{c}_2) \leq 2t < d,$$

amiből következik, hogy $\mathbf{c}_1 = \mathbf{c}_2$, ez pedig ellentmond a feltevésünknek. □

2.1.9. következmény. *Ha $\mathcal{C}$ t-hibajavító kód, akkor tetszőleges $\mathbf{x} \in \mathbb{F}_q^n$ vektorhoz legfeljebb egy olyan $\mathbf{c} \in \mathcal{C}$ kódszó létezik, amelyre $d(\mathbf{c}, \mathbf{x}) \leq t$.*

Ha a küldő csak kódszavakat továbbít, és a továbbítás során legfeljebb t hiba lép fel, akkor a fogadó dekódolni tudja az eredeti üzenetet. Ugyanis a kapott üzenet pontosan egy kódszó köré írt t-sugarú gömbben lesz benne, s ennek a gömbnek a középpontja volt az eredeti üzenet.

Megjegyzés. Ha továbbítás során legfeljebb t hiba lép fel, az azzal ekvivalens, hogy az $\mathbf{e}$ hibavektor legfeljebb t koordinátája lehet nemnulla, azaz $\mathrm{wt}(\mathbf{e}) \leq t$.

Tehát, ha a fogadott vektor $\mathbf{x} = \mathbf{c} + \mathbf{e}$, akkor megkeressük a (Hamming-értelemben) legközelebbi $\hat{\mathbf{c}} \in \mathcal{C}$ kódszót, és erre dekódolunk, azaz

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathcal{C}} d(\mathbf{x}, \mathbf{c}),$$

ahol $\arg\min_{\mathbf{c}\in\mathcal{C}} d(\mathbf{x}, \mathbf{c})$ a $d(\mathbf{x}, \mathbf{c})$ függvény azon $\mathbf{c}$ argumentuma, melyre a függvény minimális. Ezt a módszert *legközelebbiszomszéd-dekódolásnak* (vagy röviden NN-dekódolásnak – az angol *nearest neighbour decoding* elnevezésből) nevezzük.

Megjegyzés. A $\hat{\mathbf{c}}$ kódszó nem feltétlenül van egyértelműen meghatározva. Ezért implementálás esetén világosnak kell lennie, hogy az $\mathbf{x}$-hez legközelebb eső kódszavak közül melyiket választja a dekódoló algoritmus.

A fentiekben technikailag nem dekódoltunk, „csak" hibát javítottunk, mégis dekódolásnak nevezzük például az NN-dekódolást is. Ezt megtehetjük, hiszen láttuk, hogy az $E\colon \mathcal{M} \to \mathcal{C}$ kódolás bijekció, tehát létezik inverze, így mivel hibajavítás során $\hat{\mathbf{c}} \in \mathcal{C}$ kódszóra javítunk, ebből már egyértelmű az $E^{-1}(\hat{\mathbf{c}}) = \hat{\mathbf{m}} \in \mathcal{M}$ üzenet is, tehát a hibajavítás és a dekódolás lényegében ugyanaz.

2.2. Lineáris kódok

További strukturáltság megkövetelése nélkül egy kód hasznavehetősége meglehetősen korlátozott. Ha egy kódot a gyakorlatban akarunk alkalmazni, akkor tárolnunk kell a kódszavakat, meg kell határoznunk a kód minimumtávolságát, a dekódolásnál pedig ki kell számolnunk a kapott üzenetnek a kódszavaktól vett távolságát. Ezeket a műveleteket a kódok egy speciális osztályára sokkal egyszerűbben tudjuk elvégezni, mint az általános esetben. A továbbiakban ilyen kódokkal foglakozunk.

Legyen $n \in \mathbb{N}^+$ tetszőleges rögzített pozitív egész, és tekintsük az $\mathbb{F}_q^n$ vektorteret.

2.2.1. definíció (lineáris kód)**.** Ha $\mathcal{C}$ egy k-dimenziós altere $\mathbb{F}_q^n$-nek, azaz $\mathcal{C} \leq \mathbb{F}_q^n$, akkor $\mathcal{C}$-t egy $\mathbb{F}_q$ feletti *lineáris* $[n, k]$*-kódnak* nevezzük.

Megjegyzés. A $\mathcal{C}$ lineáris $[n, k]$-kódnak q^k kódszava van, így egy $\mathbb{F}_q$ feletti lineáris $[n, k]$-kód rátája

$$R = \frac{\log_q M}{n} = \frac{\log_q q^k}{n} = \frac{k}{n}.$$

Egy lineáris kód megadásának két leggyakoribb módja a kód generátormátrixának, illetve paritásellenőrző mátrixának megadása.

2.2.2. definíció (generátormátrix)**.** A $\mathcal{C}$ lineáris $[n,k]$-kód *generátormátrixának* nevezünk bármely olyan $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ mátrixot, melynek sorai bázist alkotnak $\mathcal{C}$-ben.

Megjegyzés. Általában egy kódnak több generátormátrixa van.

Ha $\mathcal{C}$ egy lineáris $[n,k]$-kód $\mathbf{G}$ generátormátrixszal, akkor $M = q^k$ különböző üzenetet vagyunk képesek kódolni. Ha praktikusan $\mathcal{M} = \mathbb{F}_q^k$ az üzenetek halmaza, akkor a kódolás az

$$E\colon \mathbb{F}_q^k \to \mathcal{C} \leq \mathbb{F}_q^n, \quad \mathbf{m} \mapsto \mathbf{mG}$$

leképezés. Definíció szerint $\mathbf{G}$ rangja $\operatorname{rank}(\mathbf{G}) = k$, ezért $\dim(\operatorname{Im} E) = k = \dim \mathbb{F}_q^k$, így a lineáris leképezések dimenziótétele szerint $\dim(\operatorname{Ker} E) = 0$, tehát E injektív. Mivel $|\mathcal{C}| = \left|\mathbb{F}_q^k\right| = q^k$, így következik, hogy E bijektív is.

Lineáris kódok esetén a kód minimumtávolságának meghatározása egyszerű.

2.2.3. lemma. *Egy lineáris kód minimumtávolsága megegyezik a legkisebb súlyú nemnulla kódszó súlyával.*

Bizonyítás. Ha $\mathcal{C}$ lineáris kód, akkor $\mathbf{0} \in \mathcal{C}$. Ezért

$$\begin{aligned} d(\mathcal{C}) &= \min\{d(\mathbf{x},\mathbf{y}) : \mathbf{x},\mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\} \leq \\ &\leq \min\{d(\mathbf{x},\mathbf{0}) : \mathbf{x} \in \mathcal{C}, \mathbf{x} \neq \mathbf{0}\} = \min\{\operatorname{wt}(\mathbf{x}) : \mathbf{x} \in \mathcal{C}, \mathbf{x} \neq \mathbf{0}\}, \end{aligned}$$

tehát a minimumtávolság nem nagyobb a legkisebb súlyú nem nulla kódszó súlyánál.

Másrészt ha $\mathbf{x}$ és $\mathbf{y}$ olyan kódszavak, melyekre $d(\mathcal{C}) = d(\mathbf{x},\mathbf{y})$, akkor a linearitás miatt $\mathbf{x}-\mathbf{y} \in \mathcal{C}$. Mivel

$$\operatorname{wt}(\mathbf{x}-\mathbf{y}) = d(\mathbf{x}-\mathbf{y},\mathbf{0}) = d(\mathbf{x}-\mathbf{y},\mathbf{y}-\mathbf{y}) = d(\mathbf{x},\mathbf{y}) = d(\mathcal{C}),$$

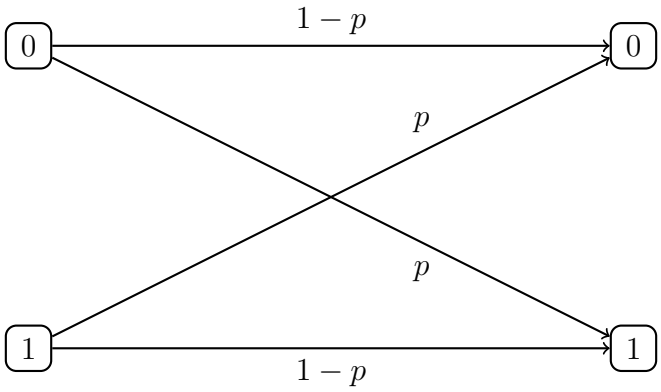
ezért van olyan kódszó, amelynek súlya éppen a minimumtávolság. □

Mivel egy lineáris kód egy vektortér altere, így valamely lineáris transzformáció magját alkotja.

2.2.4. definíció (paritásellenőrző mátrix)**.** A $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ mátrixot a $\mathcal{C}$ lineáris $[n,k]$-kód *paritásellenőrző mátrixának* nevezzük, ha

$$\mathcal{C} = \left\{\mathbf{x} \in \mathbb{F}_q^n \colon \mathbf{Hx}^{\mathrm{T}} = \mathbf{0}\right\}.$$

Megjegyzés. A $\mathbf{H}$ mátrix sorvektorai szintén függetlenek. Általában $\mathcal{C}$-nek szintén több lehetséges paritásellenőrző mátrixa van.



2.2. ábra. A $0 < p < \frac{1}{2}$ csatornaparaméterű BSC

2.3. Maximum likelihood és legközelebbiszomszéd-dekódolás

A dekódolás feladata nem más mint meghatározni melyik kódszót (vagyis $\mathbf{m}$ üzenetet) küldte a küldő, ha az $\mathbf{x}$ vektort kapta a címzett (fogadó). Hatékony (gyors) dekódolási algoritmusok keresése a kódoláselmélet egy intenzíven kutatott területe a gyakorlati alkalmazások miatt. Általában a kódolás könnyű és a dekódolás nehéz, ha a kód mérete elég nagy.

Ebben a szakaszban csatornaként egy p csatornaparaméterű BSC-t fogunk csatornaként alkalmazni a 2.1. ábrán látható kommunikációs modellben (ekkor szükségképpen $q = 2$, azaz a kételemű test – $\mathbb{F}_2$ – feletti kódokat alkalmazunk), ahol $0 < p < \frac{1}{2}$, azaz egy bitet nagyobb valószínűséggel továbbítunk helyesen, mint hibásan (lásd a 2.2. ábrát).

Tegyük fel, hogy $\mathbf{c} \in \mathbb{F}_2^n$ a küldött kódszó és az $\mathbf{x} \in \mathbb{F}_2^n$ a fogadott vektort $\hat{\mathbf{c}} \in \in \mathbb{F}_2^n$-re dekódoljuk. A $\mathbb{P}(\mathbf{c} \mid \mathbf{x})$ feltételes valószínűség jelölje annak az eseménynek a valószínűségét, hogy $\mathbf{c}$ küldött kódszó, feltéve, hogy $\mathbf{x}$ fogadott vektor, illetve hasonlóan $\mathbb{P}(\mathbf{x} \mid \mathbf{c})$ azt a valószínűséget, hogy $\mathbf{x}$ fogadott vektor, feltéve, hogy a $\mathbf{c}$ kódszó lett küldve. Ezek a valószínűségek meghatározhatók a csatorna statisztikája alapján. A két valószínűség közötti kapcsolat leírható a Bayes-tétel segítségével:

$$\mathbb{P}(\mathbf{c} \mid \mathbf{x}) = \frac{\mathbb{P}(\mathbf{x} \mid \mathbf{c})\,\mathbb{P}(\mathbf{c})}{\mathbb{P}(\mathbf{x})},$$

ahol $\mathbb{P}(\mathbf{c})$ annak a valószínűsége, hogy $\mathbf{c}$ került küldésre, illetve $\mathbb{P}(\mathbf{x})$ annak a valószínűsége, hogy $\mathbf{x}$ a fogadott vektor.

MAP-dekódolás

A fenti feltételes valószínűségek alapján két természetes mód kínálkozik a dekódolás során a döntéshozatalra. Egyrészt dekódolhatunk arra a $\hat{\mathbf{c}} = \mathbf{c}$ kódszóra, amelyre a $\mathbb{P}(\mathbf{c} \mid \mathbf{x})$ valószínűség maximális: ezt *maximum a posteriori valószínűség dekódolásnak* (vagy röviden MAP-dekódolásnak az angol *maximum a posteriori probability* elnevezésből) nevezzük. Azaz MAP-dekódolás során

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in\mathcal{C}} \mathbb{P}(\mathbf{c} \mid \mathbf{x}),$$

ahol $\arg\max_{\mathbf{c}\in\mathcal{C}} \mathbb{P}(\mathbf{c} \mid \mathbf{x})$ a $\mathbb{P}(\mathbf{c} \mid \mathbf{x})$ függvény azon $\mathbf{c}$ argumentuma, melyre a függvény maximimális.

ML-dekódolás

Másrészt dekódolhatunk arra a $\hat{\mathbf{c}} = \mathbf{c}$ kódszóra, amelyre a $\mathbb{P}(\mathbf{x} \mid \mathbf{c})$ valószínűség maximális: ezt *maximum likelihood dekódolásnak* (vagy röviden ML-dekódolásnak) nevezzük. Azaz ML-dekódolás során

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in\mathcal{C}} \mathbb{P}(\mathbf{x} \mid \mathbf{c}).$$

Tekintsük a ML-dekódolást egy $0 < p < \frac{1}{2}$ csatornaparaméterű BSC mellett. Ha $\mathbf{x} = x_1x_2\cdots x_n$ és $\mathbf{c} = c_1c_2\cdots c_n$, akkor

$$\mathbb{P}(\mathbf{x} \mid \mathbf{c}) = \prod_{i=1}^{n} \mathbb{P}(x_i \mid c_i),$$

hiszen feltettük, hogy a bithibák függetlenek (a csatorna memóriátlan). A 2.2. ábráról leolvasható, hogy

$$\mathbb{P}(x_i \mid c_i) = \begin{cases} p, & \text{ha } x_i \neq c_i; \\ 1-p & \text{ha } x_i = c_i. \end{cases}$$

Ebből

$$\mathbb{P}(\mathbf{x} \mid \mathbf{c}) = p^{d(\mathbf{x},\mathbf{c})}(1-p)^{n-d(\mathbf{x},\mathbf{c})} = (1-p)^n \left(\frac{p}{1-p}\right)^{d(\mathbf{x},\mathbf{c})}, \tag{2.3}$$

ahol $0 < \frac{p}{1-p} < 1$, hiszen feltettük, hogy $0 < p < \frac{1}{2}$.

Megjegyzés. A 2.1. ábrán szereplő $\mathbf{e} = e_1e_2\cdots e_n$ hibavektorra $\mathbf{x}-\mathbf{c} = \mathbf{e}$, amiből $d(\mathbf{x},\mathbf{c}) = d(\mathbf{x}-\mathbf{c},\mathbf{0}) = \operatorname{wt}\mathbf{e}$. Tehát (2.3) alapján az $\mathbf{e}$ hibavektor BSC esetén egy olyan véletlen n-hosszú bitvektor, melynek súlya (n,p) paraméterű binomiális eloszlást követ, azaz $\operatorname{wt}\mathbf{e} \sim \mathcal{B}_n(p)$.

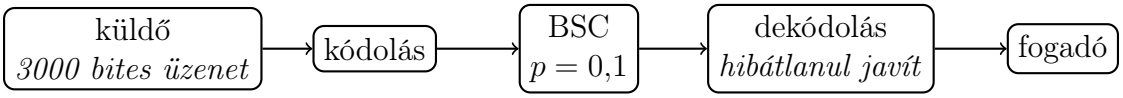
Tehát $\mathbb{P}(\mathbf{x} \mid \mathbf{c})$ maximalizálása ekvivalens $d(\mathbf{x}, \mathbf{c})$ minimalizálásával: azon $\mathbf{c} \in \mathcal{C}$ kódszót keressük, amely legközelebb esik $\mathbf{x}$-hez Hamming-értelemben. Ez azonban éppen a legközelebbiszomszéd-dekódolás, tehát BSC mellett az ML- és az NN-dekódolás ekvivalensek.

Megjegyzés. Elméletileg előfordulhat, hogy olyan $\mathbf{x}$ a fogadott vektor, melyre két különböző $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ kódszó létezik, melyektől $\mathbf{x}$ egyformán minimális távolságra van. A gyakorlatban ennek a valószínűsége elenyésző, így úgy tekintjük, hogy minden esetben egyetlen minimális távolságra eső kódszó létezik a fogadott vektortól.

3 Hibajavítás zajos csatornában

Tekintsük a 3.1. ábrán modellezett feladatot: 3000 bitnyi információt szeretnénk továbbítani egy magas zajjal rendelkező (legyen a csatornaparaméter $p = 0{,}1$) bináris szimmetrikus csatornán úgy, hogy átlagosan öt alkalomból legalább négyszer minden fogadott vektort jól javítsunk NN-dekódolással, mindezt úgy, hogy az alkalmazott kód rátája a lehető legnagyobb legyen. Claude Shannon 1948-ban publikált cikkében megfogalmazott és bizonyított a kódoláselmélet egyik alapvető tétele szerint elég nagy n kódhossz esetén létezik a fentebb megfogalmazott kritériumoknak megfelelő kód, valamint egy felső határt is megfogalmaz az elérhető rátára vonatkozóan: az elérhető ráta legfeljebb akkora, mint a csatorna C kapacitása.

Ebben a fejezetben a kitűzött feladat megoldásának egy lehetséges megközelítését írjuk le, amit Shannon tétele inspirált: véletlen lineáris kódokat fogunk alkalmazni a feladatra. Ehhez definiálnunk kell, mit értünk véletlen lineáris kód alatt, azaz hogyan képezhetjük őket, valamint mivel ezen családba tartozó kódok véges hosszal rendelkeznek – illetve véges hossz esetén tudjuk számítógépes szimulációval tesztelni őket –, ezért az elérhető rátára a csatornakapacitásnál élesebb felső határ is bemutatásra kerül.



3.1. ábra. A feladat modellje

3.1. Shannon tétele

A következőkben Czédli Gábor *Boole-függvények* [BF] című könyvének gondolatmenetét követjük.

Tekintsük a következő gondolatkísérletet. A küldő folyamatosan szeretne biteket küldeni a fogadónak egy $p = 0{,}1$ paraméterű BSC-n. A bitek véletlenszerűek (származhatnak például pénzdobálásból vagy például egy űrszonda megfigyeléseiből), és szabályos időközönként képződnek, mondjuk másodpercenként egy új bit jön létre. A BSC másodpercenként csak 3 bit továbbítását teszi lehetővé. Ezért a küldő minden k-adik másodpercben az addigra összegyűlt k bitből álló $\mathbf{m}$ üzenetet egy $E\colon \mathbb{F}_2^k \to \mathcal{C} \subseteq \mathbb{F}_2^{3k}$ kódolással kódolja, és $E(\mathbf{m}) = \mathbf{c}$-t haladéktalanul továbbítani kezdi.

Kérdés. Elérhető-e tetszőleges nagy biztonság k és E alkalmas megválasztásával, azaz a hibás dekódolás valószínűsége tetszőlegesen kicsivé tehető-e?

PER-érték

A hibás dekódolás valószínűségét egy kód esetén a kód úgynevezett PER-értéke adja meg, amelyet a következőképp definiálunk.

3.1.1. definíció (PER)**.** Legyen $\mathcal{C}$ egy $\mathbb{F}_q$ feletti (n, M)-kód egy $0 < p < \frac{1}{2}$ paraméterű BSC mellett, tetszőleges dekódolással. Ekkor a $\mathcal{C}$ kód *csomaghiba-valószínűsége* (vagy *PER-értéke* az angol *packet error rate* kifejezésből)

$$\mathrm{PER} = 1 - \frac{1}{M} \sum_{\mathbf{c} \in \mathcal{C}} \mathbb{P}(\hat{\mathbf{c}} = \mathbf{c}).$$

Példa. A fenti gondolatkísérletben $k = 1$-et választva és az 1.4. szakaszban bemutatott $\mathbb{F}_2 \to \{000, 111\} \subseteq \mathbb{F}_2^3$ 3-ismétlő kódot, illetve dekódolásként a *többségi döntést* alkalmazva (könnyen látható, hogy ez ekvivalens az ML-dekódolással) hibásan dekódolunk minden olyan esetben, amikor egynél több hiba lép fel a továbbítás során, tehát a hibás dekódolás valószínűsége

$$\mathbb{P}(\mathrm{wt}\,\mathbf{e} \geq 2) = \mathbb{P}(\mathcal{B}_3(0{,}1) \geq 2) = \binom{3}{2} 0{,}1^2 \cdot 0{,}9 + \binom{3}{3} 0{,}1^3 = 0{,}028.$$

Valóban, a PER-érték definíció szerinti számolása is erre az eredményre vezet. Nyilván $M = 2$, valamint az 1.4. ábráról leolvasható, mely esetleges fogadott vektorokat mely kódszavakra dekódoljuk. A $\mathbb{P}(\hat{\mathbf{c}} = \mathbf{c})$ valószínűséget pedig megkapjuk, ha meghatározzuk a $\mathbf{c}$-re dekódolandó vektorokra, hogy mekkkora valószínűséggel lesz a BSC-n történő

továbbítás során $\mathbf{c}$-ből az adott vektor, majd ezeket a valószínűségeket összegezzük. Például 000-ra dekódoljuk a tőle 1 távolságra lévő vektorokat (ezekből három van), illetve magát 000-t, így

$$\mathbb{P}(\hat{\mathbf{c}} = 000) = 3 \cdot 0{,}1 \cdot 0{,}9^2 + 0{,}9^3 = 0{,}972.$$

Az 111 kódszó esetén analóg módon $\mathbb{P}(\hat{\mathbf{c}} = 111) = 0{,}972$, így

$$\mathrm{PER} = 1 - \frac{1}{2} \cdot 2 \cdot 0{,}972 = 0{,}028.$$

Megjegyzés. A PER-érték meghatározása rávilágít arra, hogy a determinisztikus dekódolás lényegében egy partícionálási feladat: az $\mathbb{F}_q^n$ teret oly módon kell M partícióra osztanunk, hogy minden partícióban pontosan egy kódszó legyen.

Shannon-tétel

A korábban feltett kérdésre, miszerint elérhető-e tetszőleges nagy biztonság k és E alkalmas megválasztásával, meglepő módon az alábbi tétel – melynek technikai jellegű bizonyítását [BF] részletezi – igenlő választ ad.

3.1.2. tétel (Shannon, 1948)**.** *Legyen $0 < p < \frac{1}{2}$ és $0 < R < 1$ rögzített úgy, hogy*

$$R < 1 - h(p) = 1 + p \log_2 p + (1-p) \log_2 (1-p). \tag{3.1}$$

Ekkor tetszőleges pozitív ϵ-hoz létezik olyan $\kappa \in \mathbb{N}$, hogy bármely $\kappa \leq k \in \mathbb{N}$ esetén létezik olyan $\mathcal{C}$ bináris $\left(\left\lfloor \frac{k}{R} \right\rfloor, 2^k\right)$-kód, hogy p paraméterű bináris szimmetrikus csatornát és legközelebbiszomszéd-dekódolást használva bármely $\mathbf{c} \in \mathcal{C}$ küldött kódszó esetén a hibás dekódolás valószínűsége ϵ-nál kisebb lesz.

Megjegyzés. A fenti tételben R – megközelítőleg – a $\mathcal{C}$ kód rátája, a (3.1) egyenlőtlenség jobb oldalán pedig az alkalmazott p paraméterű BSC kapacitása áll. Ha bármely $\mathbf{c} \in \mathcal{C}$ küldött kódszó esetén a hibás dekódolás valószínűsége ϵ-nál kisebb, akkor nyilván a kód PER-értéke is ϵ-nál kisebb lesz, hiszen

$$\mathrm{PER} = 1 - \frac{1}{M} \sum_{\mathbf{c} \in \mathcal{C}} \mathbb{P}(\hat{\mathbf{c}} = \mathbf{c}) = \frac{1}{M} \sum_{\mathbf{c} \in \mathcal{C}} \mathbb{P}(\hat{\mathbf{c}} \neq \mathbf{c}).$$

Shannon tételének bizonyításából kitűnik, hogy a kódszavakat véletlenszerűen megválasztva várhatóan jó kódoláshoz jutunk a gazdaságosság (ráta) és a megbízhatóság

(PER-érték) szempontjából. Nagyobb k-kra ugyan az így nyert kódolás és a hozzá tartozó dekódolás az exponenciálisan növekvő memória- és időigény miatt gyakorlatilag megvalósíthatatlan, mégis ez a tétel indította el a *kivitelezhető* jó kódolások keresésére irányuló kutatásokat (például BCH-, Reed–Solomon-, Reed–Muller- és LDPC-kódok).

3.2. Véletlen lineáris kódok

Az előző szakaszban megismert Shannon-tétel eredménye azt sugallja, hogy a jelen fejezet bevezetőjében kitűzött feladat kritériumai teljesíthetőek. Azonban a hatékony kódoláshoz célszerű – amint azt a 2.2. szakaszban is kiemeltük – lineáris kódokat alkalmazni: a kódolás így kézenfekvő, a minimumtávolság meghatározása és a kódszavak tárolása is jelentősen egyszerűsödik. A következő bekezdésben összefoglaljuk a keresett lineáris kód teljesítendő kritériumait.

Feladat. Keressünk olyan

- bináris lineáris $[n, k]$-kódot
- minél nagyobb $R = \frac{k}{n}$ rátával, amelyre
- egy $p = 0{,}1$ paraméterű BSC és
- NN-dekódolás mellett
- összesen 3000 bites üzenetet kódolva
- annak a valószínűsége, hogy a továbbítást követően minden fogadott vektort helyesen javítunk legalább 0,8.

A Shannon-tétel alapján állíthatjuk, hogy egy *jó* kód rátájának felső határa az alkalmazott csatorna kapacitása. Most $p = 0{,}1$ mellett a csatorna kapacitása – ahogy az 1.3. szakaszban láttuk – (1.1) alapján

$$C = 1 - h\,(0{,}1) = 1 + 0{,}1 \log_2 0{,}1 + 0{,}9 \log_2 0{,}9 \approx 0{,}531. \tag{3.2}$$

Megjegyzés. A 3000 bites üzenetet előírő kritériumon valamelyest „lazíthatunk”: ha bináris lineáris $[n, k]$-kódot alkalmazunk, akkor az üzeneteknek k hosszú bitvektoroknak kell lenniük, így ahhoz, hogy összesen legalább 3000 bitnyi információt továbbíthassunk $\left\lceil \frac{3000}{k} \right\rceil$ darab üzenetblokkot kell küldenünk.

Véletlen lineáris kód és paritásellenőrző mátrixa

Az előző szakasz befejező bekezdésében, a Shannon-tétel bizonyítására vonatkozó megjegyzések szerint a kódszavakat véletlenszerűen megválasztva várhatóan jó kódoláshoz jutunk. Hogyan válasszuk a kódszavakat véletlenszerűen egy lineáris $[n, k]$-kód esetén? Amint a 2.2. szakaszban láttuk, egy lineáris kód megadásának egy lehetséges módja a paritásellenőrző mátrixának megadása.

3.2.1. definíció. A $\mathcal{C} \leq \mathbb{F}_2^n$ lineáris $[n, k]$-kódot *véletlen δ sűrűségű paritásellenőrző mátrixú bináris lineáris $[n, k]$-kódnak* nevezzük, ha $\mathcal{C}$ úgy lett generálva, hogy paritásellenőrző mátrixául egy olyan $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ véletlen bináris mátrixot választottunk, melyre igaz, hogy az 1-esek száma $\mathbf{H}$ elemei között $((n-k)\,n, \delta)$ paraméterű binomiális eloszlást követ, valamint teljesül, hogy $\operatorname{rank}(\mathbf{H}) = n - k$.

Megjegyzés. A δ *sűrűség* elnevezés nem indokolatlan, hiszen $\mathbf{H}$-ban a fentiek szerint az 1-esek számának várható értéke $(n-k)\,n\delta$, amiből következik, hogy $\mathbf{H}$ sűrűsége (a nem nulla elemek számának aránya a mátrix méretéhez) várhatóan $\frac{(n-k)n\delta}{(n-k)n} = \delta$.

Formálisan legyen

$$X = \sum_{i=1}^{n-k} \operatorname{wt} \mathbf{e}_i \mathbf{H},$$

ahol $\mathbf{e}_i \in \mathbb{F}_2^{n-k}$ jelöli szokásosan az i-ik egységvektort: ekkor X valóban a $\mathbf{H}$ mátrixban szereplő 1-esek számát jelöli. Ha $\mathbf{H} = [h_{i,j}]_{(n-k)\times n}$ minden $h_{i,j}$ eleme pedig független δ paraméterű Bernoulli-eloszlású véletlen változó, azaz $h_{i,j} \sim \mathcal{B}(\delta)$, akkor a függetlenség miatt $X \sim \mathcal{B}_{(n-k)n}(\delta)$.

Tehát generálhatunk úgy a 3.2.1. definícióban szereplő kódokat, hogy először felírunk egy véletlen $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ bináris mátrixot oly módon, hogy minden eleméről (nem feltétlenül igazságos) *pénzfeldobással* – azaz egy δ paraméterű Bernoulli-eloszlású véletlen változó segítségével – döntjük el, hogy 0 vagy 1 legyen, majd az így kapott mátrixra természetesen ellenőrizni kell a rangfeltételt (ha ez nem teljesül, felírunk egy új mátrixot), ha ez teljesül, akkor az így kapott $\mathbf{H}$ lesz a $\mathcal{C}$ kód paritásellenőrző mátrixa.

Hibátlan javítás és PER

Legyen $\mathcal{C}$ egy bináris lineáris $[n, k]$-kód $\mathbf{G}$ generátormátrixszal, $M = \left\lceil \frac{3000}{k} \right\rceil$, és tegyük fel, hogy egy $0 < p < \frac{1}{2}$ paraméterű BSC-n keresztül M darab véletlen csomagot szeretnénk továbbítani (azaz a küldendő üzenet hossza megközelítőleg 3000 bit), a küldendő üzenetcsomagokat pedig egyenletes eloszlás szerint választjuk $\mathbb{F}_2^k$-ból.

Jelölje $\mathbf{c}_i \in \mathbb{F}_2^n$ az i-ik üzenetcsomaghoz, $\mathbf{m}_i$-hez rendelt kódszót, míg $\hat{\mathbf{c}}_i \in \mathbb{F}_2^n$ a hiba-

javítást követően kapott kódszót ($i \in \{1, 2, \ldots, M\}$). Tekintsük a következő eseményt:

$$A^* = \{\forall\, i \in \{1, 2, \ldots, M\} : \hat{\mathbf{c}}_i = \mathbf{c}_i\},$$

azaz A^* jelöli azt az eseményt, hogy minden kapott üzenetet jól javítunk. Így már formalizálhatjuk a kitűzött feladat utolsó két kritériumát: legyen $\mathbb{P}(A^*) \geq 0{,}8$.

A BSC tulajdonságai, illetve a hibajavító algoritmusunk (NN-dekódolás, ami ekvivalens az ML-dekódolással BSC mellett) következtében a különböző csomagok – vektorok – javításai egymástól függetlenek, így a teljes valószínűség tétele szerint

$$\begin{aligned}\mathbb{P}(A^*) &= \prod_{i=1}^{M} \mathbb{P}(\hat{\mathbf{c}}_i = \mathbf{c}_i) = \prod_{i=1}^{M} \left(\sum_{\mathbf{c} \in \mathcal{C}} \mathbb{P}(\hat{\mathbf{c}} = \mathbf{c}) \cdot \mathbb{P}(\mathbf{c} = \mathbf{c}_i) \right) = \\ &= \prod_{i=1}^{M} \left(\frac{1}{2^k} \sum_{\mathbf{c} \in \mathcal{C}} \mathbb{P}(\hat{\mathbf{c}} = \mathbf{c}) \right) = (1 - \mathrm{PER})^M = (1 - \mathrm{PER})^{\left\lceil \frac{3000}{k} \right\rceil},\end{aligned}$$

hiszen a $\mathbf{c}_i$ kódszavakat egyenletes eloszlás szerint választottuk $\mathcal{C}$-ből. Ebből a kód PER-értéke és a kritérium közötti kapcsolat:

$$\begin{aligned}(1 - \mathrm{PER})^M &= \mathbb{P}(A^*) \geq 0{,}8; \\ 1 - \mathrm{PER} &\geq 0{,}8^{\frac{1}{M}}; \\ \mathrm{PER} &\leq 1 - 0{,}8^{\frac{1}{M}} = 1 - 0{,}8^{\frac{1}{\left\lceil \frac{3000}{k} \right\rceil}} \leq 1 - 0{,}8^{\frac{k}{3000}},\end{aligned}$$

tehát egy $[n, k]$-kód pontosan akkor felel meg a fenti kritériumnak, ha a PER-értéke kisebb vagy egyenlő, mint $1 - 0{,}8^{\frac{1}{M}}$, ahol $M = \left\lceil \frac{3000}{k} \right\rceil$.

Felső becslés az elérhető dimenzióra véges kódhossz esetén

A Shannon tételében garantáltan létező kód hossza nagyon nagy lehet, azonban a gyakorlatban nagy hosszal rendelkező kódok NN-dekódolása csak kódok speciális családjaira kivitelezhető hatékonyan. Véletlen lineáris kódok esetén – korábbi észrevételünk szerint ezek között lehet érdemes keresni – nem áll rendelkezésünkre hatékony dekódolási algoritmus.

Ezért hasznos lehet a dimenzióra tett felső becslés véges kódhossz esetén, hogy szűkíthessük a *jó* kódok keresési terét (nevezzük *jó* kódnak a kitűzött feladat kritériumainak eleget tevő kódot). Ez a felső határ kerül bemutatásra a következőkben, melyet Yuriy Polyanskiy, H. Vincent Poor és Sergio Verdú bizonyítottak 2010-ben (a bizonyítást [CCR] részletezi).

3.2.2. tétel (Polyanskiy – Poor – Verdú, 2010). *Tekintsünk egy bináris szimmetrikus csatornát olyan p paraméterrel, melyre $p \notin \left\{0, \frac{1}{2}, 1\right\}$, és legyen $\mathcal{C} \leq \mathbb{F}_2^n$ egy bináris lineáris $[n,k]$-kód ϵ PER-értékkel. Ekkor*

$$k \leq n\left(1 - h\left(p\right)\right) - \sqrt{np\left(1-p\right)} \log_2 \frac{1-p}{p} Q^{-1}\left(\epsilon\right) + \frac{1}{2}\log_2 n + O\left(1\right).$$

Megjegyzés. A tétel állítása a kód dimenziójára vonatkozik, de ebből könnyedén nyerhetünk a rátára vonatkozó felső becslést:

$$R = \frac{k}{n} \leq 1 - h\left(p\right) - \sqrt{\frac{p\left(1-p\right)}{n}} \log_2 \frac{1-p}{p} Q^{-1}\left(\epsilon\right) + \frac{1}{2} \cdot \frac{\log_2 n}{n} + O\left(\frac{1}{n}\right),$$

ahol $1 - h\left(p\right)$ éppen a p paraméterű BSC kapacitása.

A kitűzött feladat esetében $p = 0{,}1$, így a tétel szerint egy ϵ PER-értékű n-hosszú lineáris $[n,k]$-kód esetében

$$\begin{aligned} k &\leq n\left(1 - h\left(0{,}1\right)\right) - \sqrt{n \cdot 0{,}1 \cdot 0{,}9} \log_2 \frac{0{,}9}{0{,}1} Q^{-1}\left(\epsilon\right) + \frac{1}{2}\log_2 n + O\left(1\right) \approx \\ &\approx 0{,}531n - 0{,}951\sqrt{n} Q^{-1}\left(\epsilon\right) + \frac{1}{2}\log_2 n + O\left(1\right). \end{aligned}$$

3.3. Algoritmus a legközelebbiszomszéd-dekódolásra

Az előző szakaszban említettük, hogy az NN-dekódolás csak kódok speciális családjaira kivitelezhető hatékonyan. Ezért ebben a szakaszban a módosított NN-dekódoláshoz szükséges segédfüggvények algoritmusai kerülnek bemutatásra, mely módosított dekódolás lényege, hogy a bitvektorokat állandó hosszúságú *szeletekre* darabolva a szeletek egy-egy bináris számrendszerben leírt egész számot határoznak meg, így a bitvektor ezen egészek vektoraként reprezentálható, majd a Hamming-távolságot bitművelettel (bitoperációval) határozzuk meg.

A következőkben két bitműveletet is felhasználunk: a *bitszintű szorzást* és a *bitszintű kizáró vagyot* – ezek jól ismert, természetes számokon (beleértve a 0-t is) értelmezett műveletek.

Bitszintű szorzás. Legyen $x, y \in \mathbb{N}_0$. Ekkor x és y bitszintű szorzatát (AND) $x \wedge y$

jelöli. Például $x = 5$ és $y = 3$ esetén

$$x = 101_2;$$
$$y = 011_2;$$
$$x \wedge y = 001_2 = 1.$$

Bitszintű kizáró vagy. Legyen $x, y \in \mathbb{N}_0$. Ekkor a bitszintű kizáró vagyot (XOR) $x \oplus y$ jelöli x és y esetén. Például $x = 5$ és $y = 3$ esetén

$$x = 101_2;$$
$$y = 011_2;$$
$$x \wedge y = 110_2 = 6.$$

Megjegyzés. A XOR-műveletet (bitszintű kizáró vagyot) értelmezhetjük a következőképpen is: ha a két egész szám kettes számrendszerbeli alakjaira bitvektorokként tekintünk, akkor a művelet eredménye a két vektor összege lesz ($\mathbb{F}_2$ felett).

Bitvektorok és egészek

Tekintsük az $\mathbb{F}_2$ feletti n-dimenziós vektorteret, $\mathbb{F}_2^n$-t, és legyen b egy rögzített pozitív egész szám. Tetszőleges $x_1x_2\cdots x_n = \mathbf{x} \in \mathbb{F}_2^n$ vektor esetén tekintsük az első b koordináta által alkotott $x_1x_2\cdots x_b$ blokkot, a második b koordináta által alkotott $x_{b+1}x_{b+2}\cdots x_{2b}$ blokkot, és így tovább, egészen az ℓ-edik ilyen blokkig, ahol

$$\ell = \left\lceil \frac{n}{b} \right\rceil. \tag{3.3}$$

Megjegyzés. Az utolsó ilyen – a fentiek alapján képzett – blokkot nem feltétlenül alkotja b darab $\mathbf{x}$-beli koordináta: az utolsó blokk ℓ (3.3)-beli definíciója miatt éppen $x_{(\ell-1)b+1}x_{(\ell-1)b+2}\cdots x_n$ lesz.

Minden $i \in \{1, 2, \ldots, \ell\}$ esetén definiáljuk az

$$a_i = \sum_{j=0}^{b-1} \tilde{x}_{(i-1)\cdot b+j+1} \cdot 2^j \tag{3.4}$$

számot, ahol $\tilde{x}_j \in \{0, 1\} \subset \mathbb{N}$ az $x_j \in \mathbb{F}_2$ koordinátaértékből (szokásos módon) képzett egész szám, ha $j \in \{1, 2, \ldots, n\}$, különben pedig legyen $\tilde{x}_j = 0$.

Megjegyzés. Vegyük észre, hogy a (3.4)-ben definiált a_i éppen az i-edik blokk jobbról

balra való felírásával keletkezett kettes számrendszerbeli szám.

A definíció alapján világosan látszik, hogy bármely $i \in \{1, 2, \ldots, \ell\}$ esetén a_i egy b-bites egész szám, azaz $0 \leq a_i \leq 2^b - 1$.

A (3.4)-beli a_i-k felhasználásával definiáljuk a

$$\iota_b \colon \mathbb{F}_2^n \to \left\{0, 1, \ldots, 2^b - 1\right\}^\ell, \quad \mathbf{x} \mapsto \iota_b(\mathbf{x}) = \mathbf{a} = (a_1, a_2, \ldots, a_\ell)$$

leképezést. A 3.3.1. algoritmus bemutatja a ι_b leképezés egy lehetséges implementációját.

3.3.1. algoritmus. A ι_b leképezés (bitvektorból egész koordinátájú vektor)

Bemenet: b pozitív egész, $\mathbf{x} \in \mathbb{F}_2^n$ bitvektor.
Kimenet: a $\iota_b(\mathbf{x}) = \mathbf{a} \in \left\{0, 1, \ldots, 2^b - 1\right\}^\ell$ vektor.

```
1:  procedure BitvektorEgészvektorrá(b, x)
2:      a ← 0
3:      i ← 1
4:      j ← 0
5:      a ← 0
6:      for k ← 1, k ≤ n, k ← k + 1 do
7:          if j = b then
8:              a_i ← a
9:              i ← i + 1
10:             j ← 0
11:             a ← 0
12:         a ← a + x̃_k · 2^j
13:         j ← j + 1
14:     a_ℓ ← a
15:     return a
```

Megjegyzés. Definícióból adódóan a ι leképezés injekció, sőt a XOR-műveletnél tett megjegyzés alapján könnyen ellenőrizhető, hogy tetszőleges $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ esetén, ha

$$\iota_b(\mathbf{x}) = \mathbf{a} = (a_1, a_2, \ldots, a_\ell);$$
$$\iota_b(\mathbf{y}) = \mathbf{b} = (b_1, b_2, \ldots, b_\ell),$$

akkor

$$\iota_b(\mathbf{x} + \mathbf{y}) = \iota_b(\mathbf{x}) \oplus \iota_b(\mathbf{x}) = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_\ell \oplus b_\ell). \tag{3.5}$$

Súlytáblák

A XOR-műveletre vonatkozó megjegyzésben láttuk, hogy természetes módon tekinthetünk a természetes számokra bitvektorokként is, így értelmezhető egy egész szám Hamming-súlya.

3.3.2. definíció (egész szám Hamming-súlya)**.** Legyen $b \in \mathbb{N}^+$ rögzített pozitív egész szám és $a \in \{0, 1, \ldots, 2^b - 1\}$, ahol a kettes számrendszerbeli alakja $(\overline{a_1 a_2 \cdots a_b})_2$, ahol $a_i \in \{0, 1\}$ ($i \in \{1, 2, \ldots b\}$). Ekkor az *a szám Hamming-súlya*

$$\operatorname{int wt} a = \sum_{i=1}^{b} a_i.$$

Egy b-bites a természetes szám Hamming-súlyát könnyen meghatározhatjuk az AND (bitszintű és) művelet segítségével, ahogy az a 3.3.3. algoritmus is mutatja.

3.3.3. algoritmus. Egész szám Hamming-súlyának meghatározása

Bemenet: b pozitív egész, $a \in \{0, 1, \ldots, 2^b - 1\}$.
Kimenet: $\operatorname{int wt} a$, az a szám Hamming-súlya.

```
1: procedure EgészHammingSúly(b, a)
2:     w ← 0
3:     for k ← 0, k ≤ b − 1, k ← k + 1 do
4:         if a ∧ 2^k ≠ 0 then
5:             w ← w + 1
6:     return w
```

Megjegyzés. A 3.3.3. algoritmus valóban az a szám Hamming-súlyát adja vissza, hiszen $a \wedge 2^k$ pontosan akkor nem 0, ha a-ban a 2^k helyiértéken 1-es szerepel.

Így egy adott b esetén felírhatunk egy olyan 2^b méretű tömböt, a *súlytáblát* (az elemek sorszámozását 0-val kezdve), hogy minden i címkéjű elem éppen az i szám Hamming-súlya, azaz $\operatorname{int wt} i$. Formálisan legyen $b \in \mathbb{N}^+$ rögzített pozitív egész szám, és tekintsük a

$$W_b \colon \{0, 1, \ldots, 2^b - 1\} \to \{0, 1, \ldots, b\}, \quad a \mapsto W_b(a) = \operatorname{int wt} a$$

leképezést. A súlytábla meghatározásának algoritmikus leírása a 3.3.4. algoritmusban látható.

Így a korábbiak összevetéséből adódik, hogy tetszőleges $\mathbf{x} \in \mathbb{F}_2^n$ bitvektor esetén, ha

3.3.4. algoritmus. A súlytábla meghatározása

Bemenet: b pozitív egész szám.
Kimenet: a W_b súlytábla.

```
1: procedure SúlytáblátGenerál(b)
2:     W_b ← üres                                  ▷ üres az üres tömb.
3:     for a ← 0, a ≤ 2b − 1, a ← k + 1 do
4:         W_b[a] ← EgészHammingSúly(b, a)         ▷ W_b[a] a tömb a címkéjű eleme.
5:     return W_b
```

$\iota_b(\mathbf{x}) = (\alpha_1, \alpha_2, \ldots, \alpha_\ell)$, akkor

$$\operatorname{wt}(\mathbf{x}) = \sum_{i=1}^{\ell} \operatorname{int}\operatorname{wt}\alpha_i = \sum_{i=1}^{\ell} W_b(\alpha_i) \tag{3.6}$$

NN-dekódolás bitműveletekkel

Mivel a számítógépes implementáció esetén kódoláselméletben absztrakt objektumokkal – vektorterekkel, vektorokkal – dolgozunk, ezért a kapcsolódó műveletek is számításigényesnek bizonyulhatnak az egyszerűbb műveletekkel, mint például a bitműveletekkel szemben.

Így a következőképpen módosíthatjuk a standard NN-dekódolást. Tegyük fel, hogy a $\mathcal{C} \leq \mathbb{F}_2^n$ lineáris $[n, k]$-kódot alkalmazzuk egy p paraméterű BSC mellett és legyen b egy pozitív egész szám. Ekkor $\mathcal{C}$ minden $\mathbf{c}$ elemére meghatározhatjuk annak egészvektoros reprezentációját, azaz $\iota_b(\mathbf{c})$-t és tároljuk az egészvektorokat egy olyan tömbben, ahol a címkék a megfelelő $\mathcal{C}$-beli kódszavak, valamint minden $a \in \{0, 1, \ldots, 2^b - 1\}$ esetén kiszámoljuk $W_b(a) = \operatorname{int}\operatorname{wt} a$-t, azaz meghatározzuk a súlytáblát, majd a kiszámolt értékeket szintén tároljuk (ezt már egy standard tömbben is megtehetjük).

Tegyük fel, hogy a $\mathbf{c} \in \mathcal{C}$ kódszót továbbítottuk, majd az $\mathbf{x} \in \mathbb{F}_2^n$ vektort kaptuk a továbbítást követően. Ekkor mivel $\mathbb{F}_2$ felett vagyunk, ezért

$$d(\mathbf{x}, \mathbf{c}) = \operatorname{wt}(\mathbf{x} - \mathbf{c}) = \operatorname{wt}(\mathbf{x} + \mathbf{c}).$$

Legyen $\iota_b(\mathbf{x} + \mathbf{c}) = (\beta_1, \beta_2, \ldots, \beta_\ell)$ (ahol ℓ-t (3.3)-nak megfelelően határozzuk meg). Ekkor (3.6) szerint

$$\operatorname{wt}(\mathbf{x} + \mathbf{c}) = \sum_{i=1}^{\ell} W_b(\beta_i).$$

Ha $\iota_b(\mathbf{x}) = (\alpha_1, \alpha_2, \ldots, \alpha_\ell)$ és $\iota_b(\mathbf{c}) = (\gamma_1, \gamma_2, \ldots, \gamma_\ell)$, akkor (3.5) szerint $\iota_b(\mathbf{x} + \mathbf{c}) =$

$= \iota_b(\mathbf{x}) \oplus \iota_b(\mathbf{c})$, tehát

$$\mathrm{wt}(\mathbf{x}+\mathbf{c}) = \sum_{i=1}^{\ell} W_b(\beta_i) = \sum_{i=1}^{\ell} W_b(\alpha_i \oplus \gamma_i),$$

amiből következik, hogy az NN-dekódolás a

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathcal{C}} d(\mathbf{x},\mathbf{c}) = \arg\min_{\mathbf{c}\in\mathcal{C}} \sum_{i=1}^{\ell} W_b(\alpha_i \oplus \gamma_i)$$

kódszót adja vissza.

Mivel a súlytáblát, illetve a kódszavak egészvektoros reprezentációit tároltuk, és csak összeadást, illetve bitoperációt (XOR) használtunk, ezért számítógépes implementáció esetén ez a módszer nem használ absztrakt objektumokat. Sőt ha, többféle kódot alkalmazunk, akkor a súlytáblát elég egyszer meghatározni, hiszen az csak b-től függ. A fentiek algoritmikus leírása a 3.3.5. algoritmusban látható.

3.3.5. algoritmus. Dekódolás bitoperációkkal

Bemenet: b pozitív egész szám, a $\mathcal{C}$ bináris lineáris $[n,k]$-kód.
Kimenet: a $\mathcal{C}$ kód kódszavainak egészvektoros reprezentációja.

procedure KÓDSZAVAKEGÉSZVEKTOROKKÁ($b,\mathcal{C}$)
  $\mathfrak{C}_b \leftarrow$ ÜRES
  for $\mathbf{c}\in\mathcal{C}$ **do**
    $\mathfrak{C}_b[\mathbf{c}] \leftarrow$ BITVEKTOREGÉSZVEKTORRÁ($b,\mathbf{c}$)
  return $\mathfrak{C}_b$

Bemenet: b pozitív egész, $\mathcal{C}$ bináris lineáris $[n,k]$-kód, $\mathbf{x}\in\mathbb{F}_2^n$ fogadott vektor.
Kimenet: $\hat{\mathbf{c}}\in\mathcal{C}$, $\mathbf{x}$ NN-dekódoltja.

$W_b \leftarrow$ SÚLYTÁBLÁTGENERÁL(b)
$\mathfrak{C}_b \leftarrow$KÓDSZAVAKEGÉSZVEKTOROKKÁ($b,\mathcal{C}$)
procedure BITOPDEKÓDOL($b,\mathcal{C},\mathbf{x}$)
  $\mathbf{a} \leftarrow$ BITVEKTOREGÉSZVEKTORRÁ($b,\mathbf{x}$) ▷ $\mathbf{a} = (a_1, a_2, \ldots, a_\ell)$.
  $\ell \leftarrow \left\lceil \frac{n}{b} \right\rceil$
  $S \leftarrow$ ÜRES
  for $\mathbf{c}\in\mathcal{C}$ **do**
    $(\gamma_1,\gamma_2,\ldots,\gamma_\ell) \leftarrow \mathfrak{C}_b[\mathbf{c}]$
    $S[\mathbf{c}] \leftarrow \sum_{i=1}^{\ell} W_b[a_i \oplus \gamma_i]$
  $\hat{\mathbf{c}} \leftarrow \arg\min_{\mathbf{c}\in\mathcal{C}} S$
  return $\hat{\mathbf{c}}$

A KÓDSZAVAKEGÉSZVEKTOROKKÁ függvény gyorsítható, ha $\mathcal{C}$ bináris lineáris $[n,k]$-kód: először a zérusvektorra határozzuk meg az egészvektoros reprezentációt – alkossa

első lépésként ez az egy egészvektor $\mathfrak{C}_b$-t. Ezt követően felírjuk $\mathcal{C}$ egy bázisát, majd pedig rendre a báziselemekkel vett „összegekkel" bővítjük minden lépésben $\mathfrak{C}_b$-t: lásd a 3.3.6. algoritmust.

3.3.6. algoritmus. Kódszavak egészvektoros reprezentációja lineáris kód esetén

Bemenet: b pozitív egész szám, $\mathcal{C}$ bináris lineáris $[n, k]$-kód, $\mathbf{G}$ a $\mathcal{C}$ kód generátormátrixa.

Kimenet: a $\mathcal{C}$ kód kódszavainak egészvektoros reprezentációja.

1: **procedure** KÓDSZAVAKEGÉSZVEKTOROKKÁ($b, \mathcal{C}, \mathbf{G}$)
2: $B \leftarrow \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k\}$ ▷ $\mathbf{G}$ sorvektorai.
3: $\mathbf{g}_0 \leftarrow \mathbf{0}$ ▷ $\mathbf{0} \in \mathbb{F}_2^n$.
4: $\mathfrak{C}_b \leftarrow$ ÜRES
5: $\mathfrak{C}_b[\mathbf{g}_0] \leftarrow \mathbf{0}$ ▷ $\mathbf{0} \in \mathbb{N}_0^\ell$.
6: **for** $i \leftarrow 1, i \leq k, i \leftarrow i + 1$ **do**
7: $\mathbf{a} \leftarrow$ BITVEKTOREGÉSZVEKTORRÁ($b, \mathbf{g}_i$)
8: **for** $\mathbf{c} \in \operatorname{span}\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{i-1}\}$ **do** ▷ Pontosan a $\mathfrak{C}_b$-ben már címkeként szereplő kódszavak.
9: $\mathfrak{C}_b[\mathbf{c} + \mathbf{g}_i] \leftarrow \mathfrak{C}_b[\mathbf{c}] \oplus \mathbf{a}$
10: **return** $\mathfrak{C}_b$

4 Szimulációk véletlen lineáris kódokra

Ebben a fejezetben először megvizsgáljuk, hogy a véletlen kódok PER-értékének becslése mennyi időt vesz igénybe, majd a feladat kritériumainak eleget tevő véletlen lineáris kódok keresésének egy lehetséges módszere kerül bemutatásra, valamint rávilágítunk az átlagos PER-érték és a δ sűrűségparaméter közötti kapcsolatra, végül az átlagos minimum-távolság és a sűrűség közötti kapcsolatot tárgyaljuk.

Emlékeztetőül: feladatunk a következő kritériumoknak eleget tevő véletlen δ sűrűségű paritásellenőrző mátrixú bináris lineáris $[n, k]$-kódok keresése:

- az $R = \frac{k}{n}$ ráta legyen minél nagyobb;
- egy $p = 0{,}1$ paraméterű BSC és
- NN-dekódolás mellett (a 3.3. szakaszbeli bitműveleteket alkalmazó algoritmust használva)
- összesen $M = \left\lceil \frac{3000}{k} \right\rceil$ darab üzenetet kódolva
- annak a valószínűsége, hogy a továbbítást követően minden fogadott vektort helyesen javítunk legalább 0,8, azaz a kód PER-értékére $\mathrm{PER} \leq 1 - 0{,}8^{\frac{k}{3000}}$.

Megjegyzés. Az utolsó kritériumon változtattunk: $1 - 0{,}8^{\frac{1}{M}}$ helyett $1 - 0{,}8^{\frac{k}{3000}}$-t választottuk felső korlátnak az egyszerűbb számolás végett, azonban ezzel nem követünk el nagy hibát, hiszen $1 - 0{,}8^{\frac{1}{M}} \approx 1 - 0{,}8^{\frac{k}{3000}}$ a később vizsgált dimenziótartományban.

4.1. PER-becslés és futási ideje

Tekintsünk egy véletlen δ sűrűségű paritásellenőrző mátrixú $\mathcal{C}$ bináris lineáris $[n, k]$-kódot, melynek a kitűzött feladatra való alkalmasságát szeretnénk vizsgálni. Legyen

$\mathrm{PER}_{3000} = 1 - 0{,}8^{\frac{k}{3000}}$, így $\mathcal{C}$-ről azt mondjuk, hogy teljesítette a kritériumokat – azaz *jó* kód –, ha a PER-értékének becslésekor a kapott érték kisebb vagy egyenlő, mint PER_{3000}.

PER-érték becslése

Legyen M tetszőleges pozitív egész szám. A fentebb definiált $\mathcal{C}$ kód PER-értékét a következőképpen becsüljük (lásd a 4.1.1. algoritmust): az egyszerűség kedvéért csak a $\mathbf{0}$ kódszót továbbítjuk M-szer, azaz M alkalommal függetlenül generálunk $\mathbf{e}$ hibavektorokat, ahol $\operatorname{wt}\mathbf{e} \sim \mathcal{B}_n(p)$, majd $\mathbf{e}$-t dekódoljuk. Ha a nullvektorra dekódoltunk (javítottunk), akkor *jól* dekódoltunk, különben *rosszul*. Végül $\mathcal{C}$ PER-értékét a *rossz* dekódolások számának és M-nek az arányával becsüljük.

4.1.1. algoritmus. Bináris lineáris $[n, k]$-kód PER-értékének becslése

Bemenet: $n \in \mathbb{N}^+$, $0 < p < \frac{1}{2}$.
Kimenet: $\mathbf{e} \in \mathbb{F}_2^n$ hibavektor, ahol $\operatorname{wt}\mathbf{e} \sim \mathcal{B}_n(p)$.

procedure HIBAVEKTOR(n, p)
    $(e_1, e_2, \ldots, e_n) = \mathbf{e} \leftarrow \mathbf{0}$ ▷ $\mathbf{0} \in \mathbb{F}_2^n$
    for $i \leftarrow 1, i \leq n, i \leftarrow i + 1$ **do**
        $e_i \leftarrow \mathcal{B}(p)$
    return $\mathbf{e}$

Bemenet: b pozitív egész szám, $\mathcal{C}$ bináris lineáris $[n, k]$-kód, M a küldendő csomagok száma, p a csatornaparméter.
Kimenet: a $\mathcal{C}$ kód PER-értékének becsült értéke.

$W_b \leftarrow$ SÚLYTÁBLÁTGENERÁL(b)
$\mathfrak{C}_b \leftarrow$ KÓDSZAVAKEGÉSZVEKTOROKKÁ($b, \mathcal{C}$)
procedure PERBECSLÉS($b, \mathcal{C}, M, p$)
    $r \leftarrow 0$
    for $i \leftarrow 1, i \leq M, i \leftarrow i + 1$ **do**
        $\mathbf{e} \leftarrow$ HIBAVEKTOR(n, p)
        if BITOPDEKÓDOL($b, \mathcal{C}, \mathbf{e}$) $= \mathbf{0}$ **then** ▷ $\mathbf{0} \in \mathbb{F}_2^n$
            $r \leftarrow r + 1$
    return $1 - \frac{r}{M}$

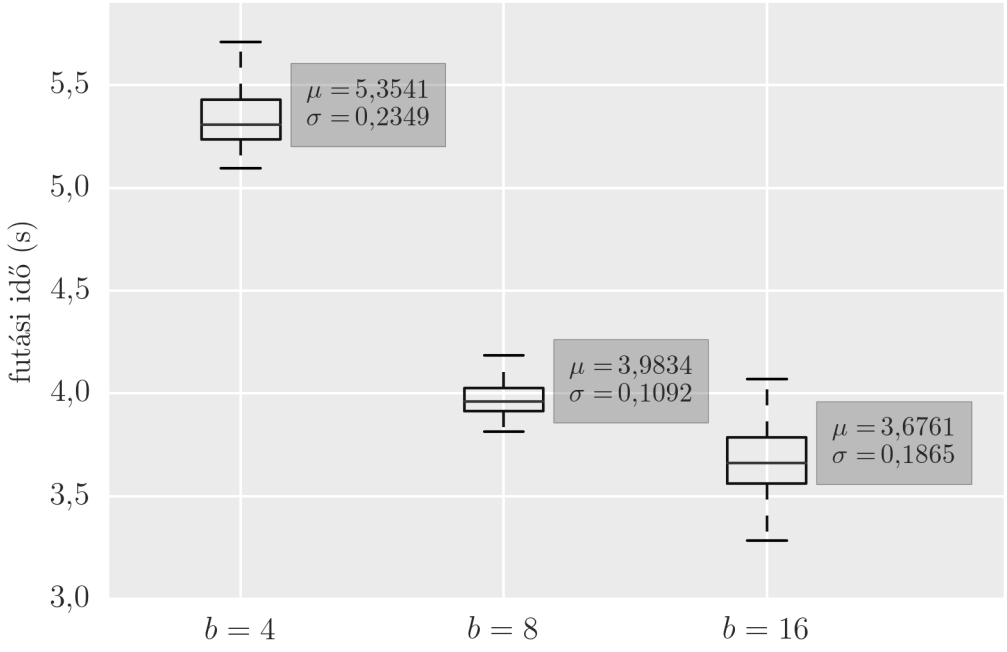
Megjegyzés. A fenti módszer elméleti szempontból nem precíz módon közelíti a kód PER-értékét, hiszen definíció szerint a PER egy átlagvalószínűség, a 4.1.1. algoritmus azonban csak a $\mathbf{0}$ kódszóra közelíti a hibás javítás valószínűségét. Azonban a futtatott szimulációk tapasztalatai azt mutatják, hogy különböző kódszavakra sem tapasztalható szignifikáns különbség a becsült PER-értékekre, így gyakorlati szempontból kizárólag a $\mathbf{0}$ kódszó alkalmazása kielégítő, és nem utolsó sorban az implementációt is jelentős mértékben egyszerűsíti.

PER-becslések futási idejei

A PER-értékek becsléseire vonatkozó futásidők vizsgálatát 100-100 darab kódra 12-, 13-, illetve 14-dimenziós kódok esetén (a kódhossz minden esetben a dimenzió háromszorosa, azaz $R = \frac{1}{3}$) különböző b értékek mellett végezzük el: $b \in \{4, 8, 16\}$, ahol küldendő üzenet hossza 3000 bit, tehát $M = \left\lceil \frac{3000}{k} \right\rceil$, illetve $p = 0{,}1$. A következők szerint végezzük a méréseket.

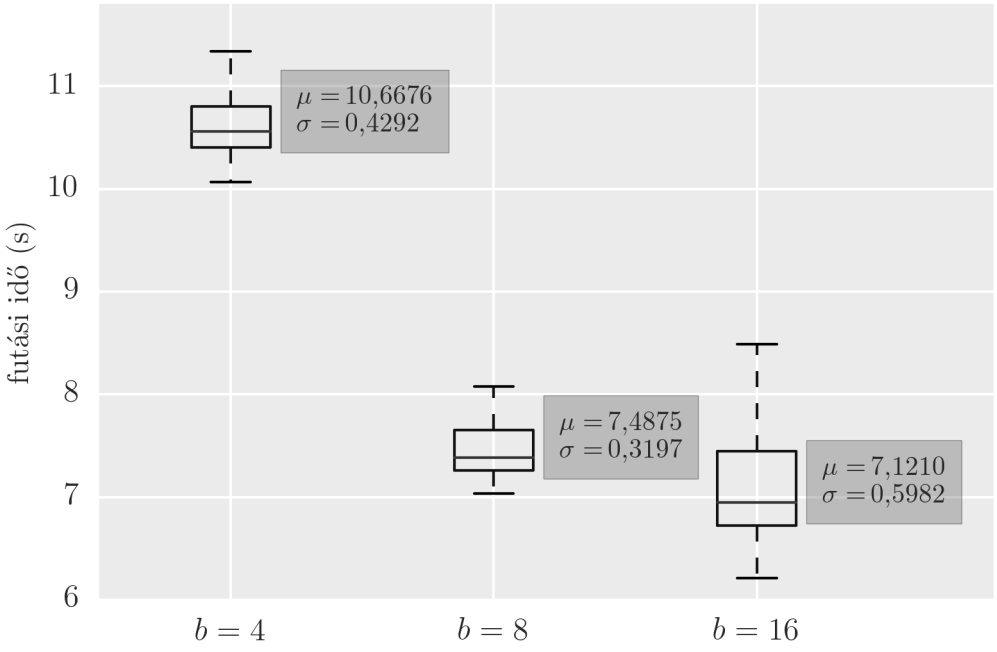
	n	$[\delta_{\min}, \delta_{\max}]$	M
$d = 12$	36	[0,0278, 0,9722]	250
$d = 13$	39	[0.0256, 0.9744]	231
$d = 14$	42	[0.0238, 0.9762]	215

4.1. táblázat. A becslések paraméterei



4.2. ábra. A PERBecslés algoritmus futási ideje $k = 12$ dimenzió esetén

1. Minden $k \in \{12, 13, 14\}$ dimenzió esetén generálunk egy 100-elemű véletlen tömböt, ahol a tömb elemei elemei $\left(\frac{R}{k}, 1 - \frac{R}{k}\right)$-n egyenletes eloszlást követnek ($\frac{k}{R}$ éppen a kódok hossza). A generált tömb minimális elemét jelölje $\delta_{\min}$, maximális elemét $\delta_{\max}$ (a dimenziónkénti paramétereket lásd a 4.1. táblázatban).



4.3. ábra. A PERBecslés algoritmus futási ideje $k = 13$ dimenzió esetén

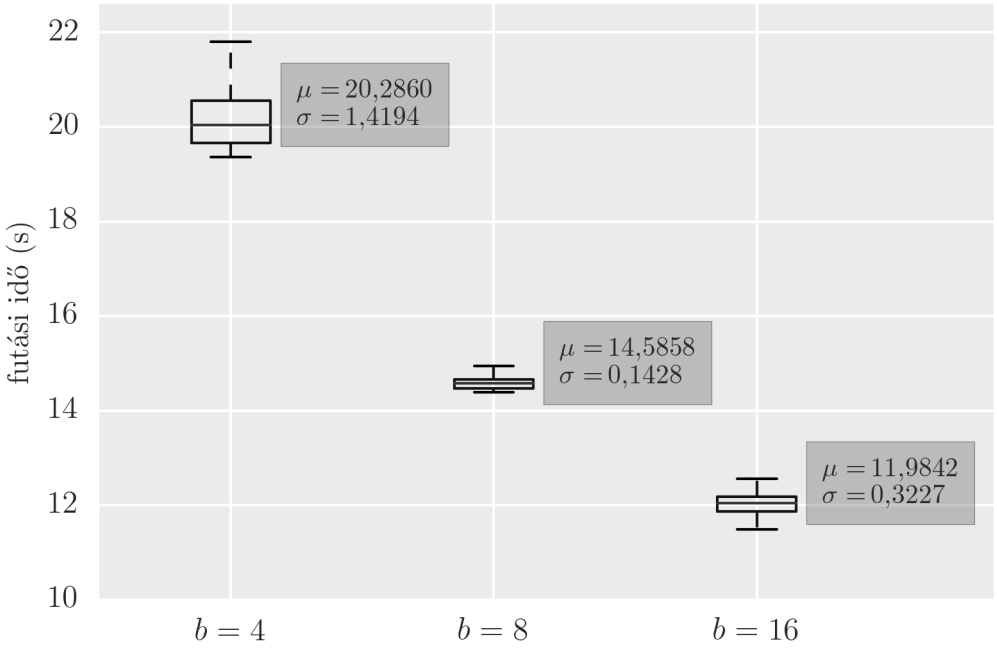
2. A tömb minden δ elemére generálunk egy véletlen δ sűrűségű paritásellenőrző mátrixú $\mathcal{C}$ bináris lineáris $\left[\frac{k}{R}, k\right]$-kódot.

3. Minden $b \in \{4, 8, 16\}$-ra futtatjuk a PER-közelítést $\left\lceil \frac{3000}{k} \right\rceil$ darab csomagra $p = 0{,}1$ mellett, azaz futtatjuk a PERBecslés $\left(b, \mathcal{C}, \left\lceil \frac{3000}{k} \right\rceil, 0{,}1\right)$ függvényt mind a 100 darab $\mathcal{C}$ kódra, és mérjük a futásidőket.

A 4.2. ábrán a $k = 12$ esetben, a 4.3. ábrán a $k = 13$ esetben, míg a 4.4. ábrán a $k = 14$ esetben látható a futási idők statisztikai analízise (μ jelöli a futási idők átlagát, míg σ a korrigált szórást).

Megjegyzés. Az $\frac{R}{k}$ és az $1 - \frac{R}{k}$ alsó és felső határok tapasztalati értékek: a $\delta \in \left(\frac{R}{k}, 1 - \frac{R}{k}\right)$ sűrűségek esetén „gyorsan” generálhatunk megfelelő ranggal rendelkező δ sűrűségű paritásellenőrző mátrixot.

Az ábrák összevetése alapján a futási idők exponenciálisan növekednek a dimenzióval, adott dimenzió esetén b növekedésével csökken a futási idő, sőt megjegyzendő, hogy minden esetben $b = 8$ esetben a legkisebb a szórás.



4.4. ábra. A PERBecslés algoritmus futási ideje $k = 14$ dimenzió esetén

4.2. Jó véletlen lineáris kódok keresése

Hogyan keressünk *jó* véletlen δ sűrűségű paritásellenőrző mátrixú bináris lineáris $[n, k]$-kódokat? Ezen kérdés megválaszolására nyújtunk egy egyszerű megközelítést a következőkben.

Egy egyszerű keresési módszer

A következő módszert alkalmazzuk a kitűzött feladat kritériumait kielégítő véletlen bináris lineáris $[n, k]$ kódok keresésére. Legyen $\delta \in \left(\frac{1}{n}, 1 - \frac{1}{n}\right)$ tetszőleges, b pozitív egész rögzítettet, $p = 0{,}1$ a csatornaparaméter.

1. Generáljunk 100 darab véletlen δ sűrűségű paritásellenőrző mátrixú $\mathcal{C}$ bináris lineáris $[n, k]$-kódot.

2. Az előzőekben generált 100 kód mindegyikére becsüljük a PER-értéket 100 küldött csomagra, azaz futtassuk PERBecslés $(b, \mathcal{C}, 100, 0{,}1)$-et.

3. Válasszuk ki 10 legjobb kódot, azaz azon 10 kódot, melyekre a becsült PER-értékek a legkisebbek.

4. A kiválasztott 10 kódra becsüljük újra a PER-értéket 1000 küldött csomaggal, azaz most PERBecslés $(b, \mathcal{C}, 1000, 0{,}1)$-et futtatjuk.

Megjegyzés. Az előző szakaszban tett megállapításaink alapján a $b = 8$ paramétert alkalmazzuk a szimulációk során.

Az előző szakaszbeli megfigyelések, valamint a szimulációk futtatásakor tapasztaltak alapján $k = 16$ dimenzió felett a futási idők kezelhetetlenné válnak, sőt $k = 15$ és $k = 16$ esetekben is már nagyon nagyok.

A szimulációk eredményei

A szimulációk során a következő kódparamétereket használjuk:

- $k \in \{12, 13, 14, 15, 16\}$ dimenziók;
- $R \in \{0{,}25, 0{,}3, 0{,}33\}$ ráták és
- $\delta \in \{0{,}05, 0{,}1, \ldots, 0{,}4\}$ sűrűségek.

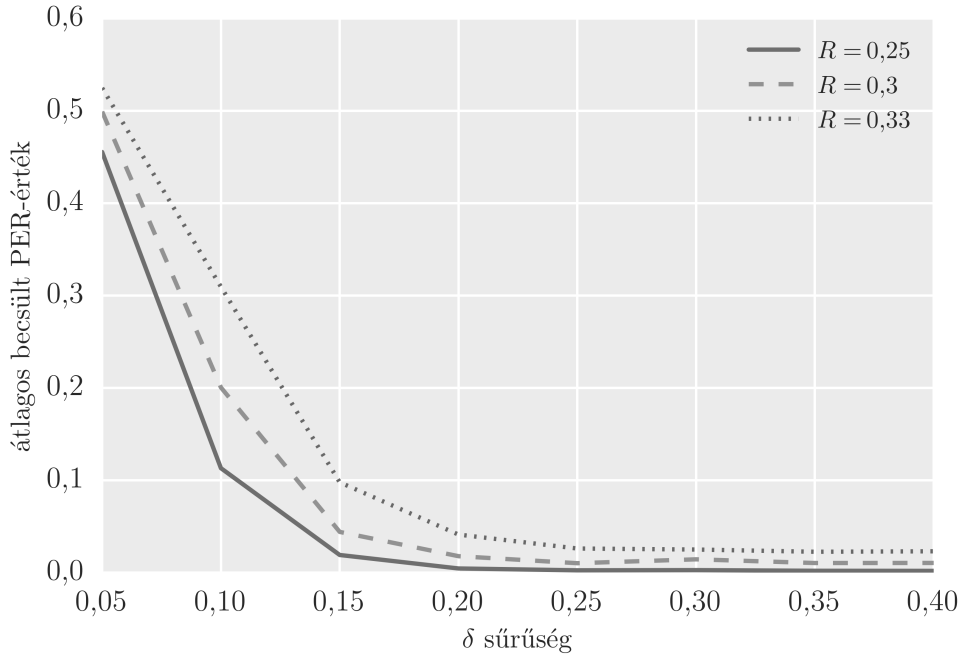
Megjegyzés. Ha $k \in \{12, 13, 14, 15, 16\}$, akkor minden k esetében $\mathrm{PER}_{3000} \approx 0{,}001$, így valamelyest lazítva a kritériumokon mondhatjuk, hogy $\mathcal{C}$ kód *jó*, ha a becsült PER-értéke kisebb vagy egyenlő, mint 0,001.

Természetes módon merül fel a következő kérdés.

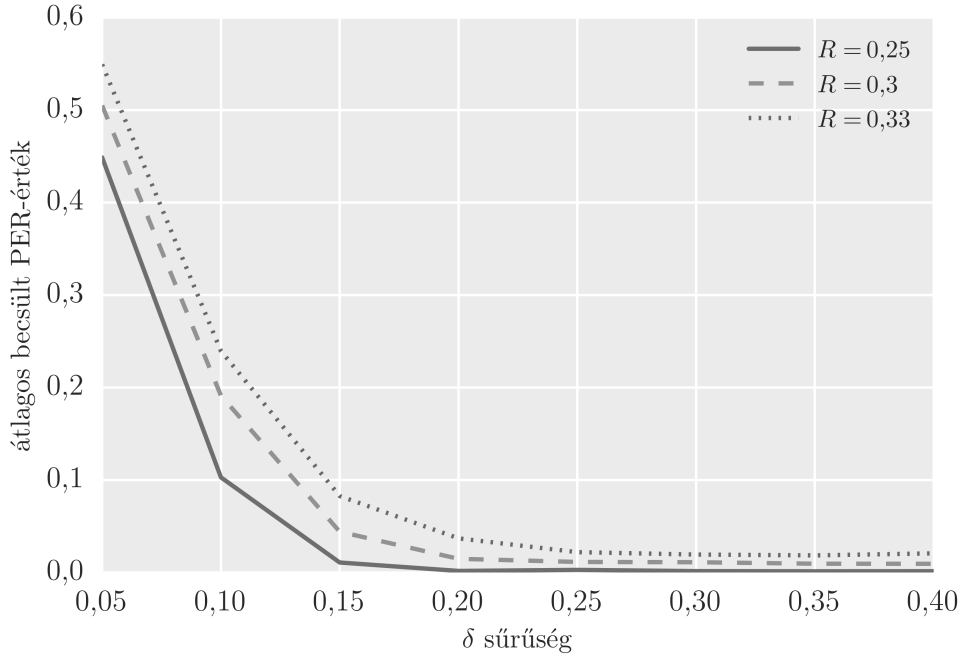
Kérdés. Függ-e egy kód (becsült) PER-értéke a δ sűrűségparamétertől?

A 4.5., a 4.6. és a 4.7. ábrákon láthatók 100 csomagra futtatott szimulációk eredményei a $k \in \{12, 13, 14\}$ esetekben, melyekről leolvasható, hogy *igen*, függ egy kód PER-értéke a δ sűrűségparamétertől, azonban $\delta = 0{,}25$-től már nem tapasztalható szignifikáns változás. Ezt a tapasztalatot felhasználva a számításigényesebb $k = 15$, illetve $k = 16$ esetekben már csak $\delta = 0{,}4$-re futtatunk szimulációkat.

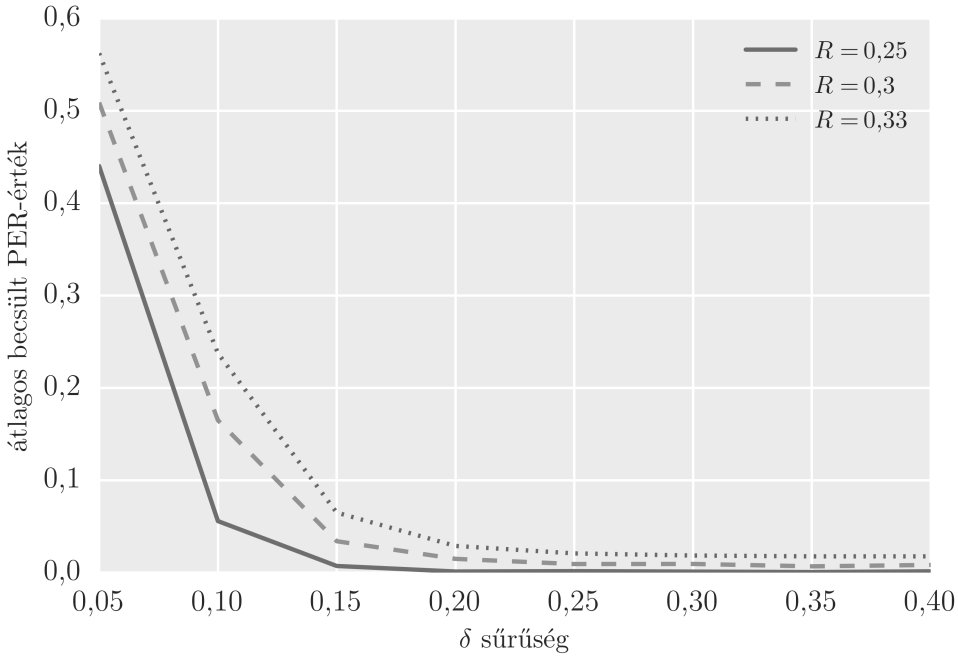
A $\delta = 0{,}4$ sűrűségre a 4.8. táblázatban láthatók a szimulációk során a 10 legjobb kód közül legjobban teljesítő (legkisebb becsült PER-értékű) kódok PER-értékei a vizsgált dimenziók és ráták mellett. Ebből kitűnik, hogy $\delta = 0{,}4$-re *jó* kódot csak az $R = 0{,}25$ ráta mellett találtunk.



4.5. ábra. Az átlagos PER-becslések sűrűségenként 100 kódra $k = 12$ dimenzió esetén



4.6. ábra. Az átlagos PER-becslések sűrűségenként 100 kódra $k = 13$ dimenzió esetén



4.7. ábra. Az átlagos PER-becslések sűrűségenként 100 kódra $k = 14$ dimenzió esetén

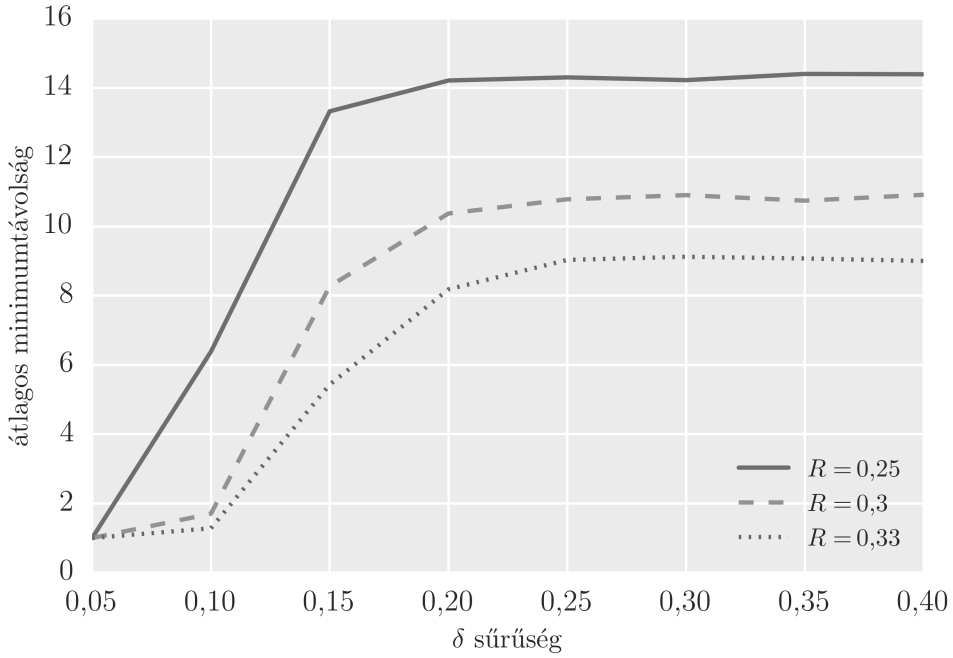
	$k = 12$	$k = 13$	$k = 14$	$k = 15$	$k = 16$
$R = 0{,}25$	0,0000	0,0000	0,0010	0,0000	0,0000
$R = 0{,}3$	0,0030	0,0060	0,0020	0,0030	0,0040
$R = 0{,}33$	0,0170	0,0120	0,0140	0,0140	0,0110

4.8. táblázat. A $\delta = 0{,}4$ sűrűségű legjobban teljesítő kódok becsült PER-értéke (1000 csomagra)

4.3. Minimumtávolság és sűrűség

A 2.1. szakaszban láttuk, hogy egy kód minimumtávolsága nagyban meghatározza a kód hibajavító képességét. Az előző szakaszban ismertetett eredmények alapján véletlen δ sűrűségű paritásellenőrző mátrixú bináris lineáris $[n, k]$-kódok esetében δ-tól függ a kód (becsült) PER-értéke. Fennáll-e hasonló viszony δ és a kód minimumtávolsága között?

A 4.9. és a 4.10. ábrákon látható $\delta \in \{0{,}05, 0{,}1, \ldots, 0{,}4\}$ sűrűségenként 100-100 véletlen δ sűrűségű paritásellenőrző mátrixú bináris lineáris $[n, k]$-kód átlagos minimumtávolsága, ahol $n = \left\lfloor \frac{k}{R} \right\rfloor$. Látszik, hogy az átlagos PER-érték függ a sűrűségtől, valamint hogy a PER-érték becsléséhez hasonló módon $\delta \approx 0{,}2$-től nem tapasztalunk szignifikáns különbséget az átlagos minimumtávolságok között (rátánként).



4.9. ábra. Az átlagos minimumtávolságok sűrűségenként 100 kódra $k = 15$ dimenzió esetén

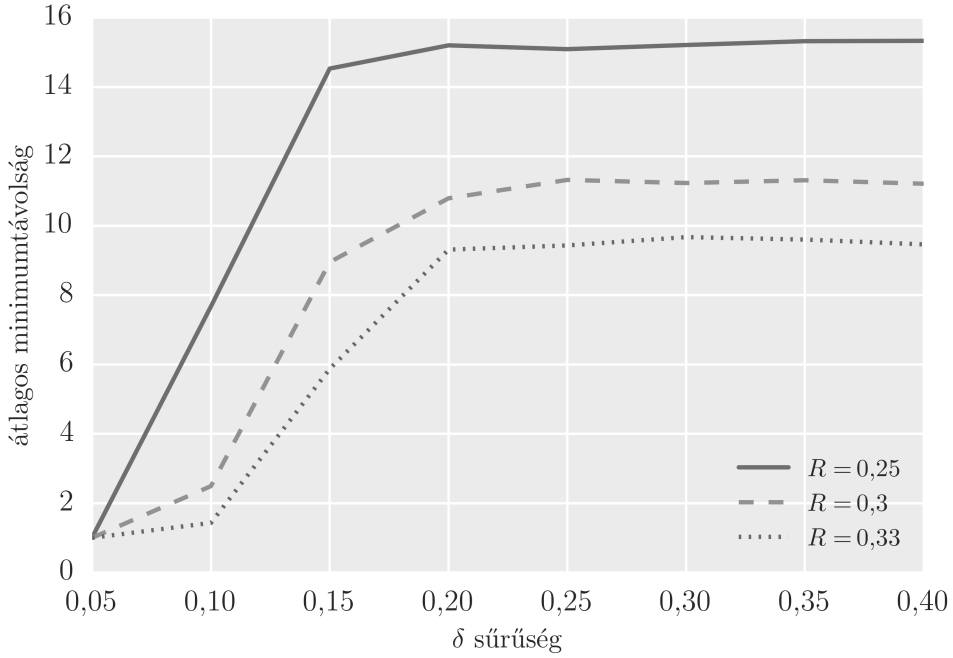
Minimumtávolság és paritásellenőrző mátrix

A 4.9. és a 4.10. ábrákat jobban megvizsgálva feltűnhet, hogy a $\delta = 0{,}05$ esetben mindkét dimenzió esetében körülbelül 1 az átlagos minimumtávolság. Milyen kapcsolat állhat ez és a paritásellenőrző mátrix struktúrája között?

Tegyük fel, hogy a véletlen $\mathcal{C}$ lineáris $[n, k]$-kódot a véletlen $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ paritásellenőrzőmátrix alapján generáltuk, illetve tegyük fel, hogy $d(\mathcal{C}) = 1$. Ekkor a 2.2.3. lemma szerint létezik $\mathcal{C}$-nek 1 súlyú kódszava, jelölje ezt $\mathbf{c}$, és tegyük fel, hogy $\mathbf{c}$-nek az i-ik koordinátája az 1 súlyt adó nem nulla elem, azaz $\mathbf{c} = \mathbf{e}_i$. A paritásellenőrző mátrix 2.2.4. definíciója szerint $\mathbf{H}\mathbf{c}^{\mathrm{T}} = \mathbf{0}$, amiből következik, hogy $\mathbf{H}$-nak az i-ik oszlopa csupa 0 kell, hogy legyen.

A fordított irány triviálisan teljesül: ha $\mathbf{H}$-nak az i-ik oszlopa csupa 0, akkor $\mathbf{H}\mathbf{e}_i^{\mathrm{T}} = 0$, tehát $\mathbf{e}_i \in \mathcal{C}$, és így $d(\mathcal{C}) = 1$.

Tehát egy lineáris kód minimumtávolsága pontosan akkor 1, ha létezik olyan paritásellenőrző mátrixa, melynek van csupa nulla oszlopa.



4.10. ábra. Az átlagos minimumtávolságok sűrűségenként 100 kódra $k = 16$ dimenzió esetén

Függelék

A 3. és a 4. fejezetekben tárgyalt algoritmusok implementációja látható itt **SageMath**-ben [S+16] megvalósítva. A 4. fejezetben leírt *jó* kódok keresésére alkalmazott egyszerű módszer a **tournament_qualification** és a **tournament_final** függvényekkel került megvalósításra.

```
wt_filename="WEIGHT_TABLES_SAVED"
def int_wt(i,bitlength):
    return long(sum(0!=(i&(2**k)) for k in range(bitlength)))
def build_wt_table(bitlength):
    return [int_wt(i,bitlength) for i in range(2**bitlength)]
WEIGHT_TABLES={}
for i in [4,8,16]:
    WEIGHT_TABLES[i]=build_wt_table(i)
save(WEIGHT_TABLES,wt_filename)
def bitvec2intrep(bitvec,bitlength):
    intrep=[]
    k=0
    a=0
    for i in range(len(bitvec)):
        if k==bitlength:
            intrep.append(long(a))
            a=long(0)
            k=0
        if 0!=bitvec[i]:
            a+=long(2**k)
        k+=1
    intrep.append(long(a))
    return intrep
def err_vec(length,p):
    return vector(GF(2),[Integer(random()<p) for i in range(length)])
def rnd_ldpc_code(length,dim,delta):
    pc_mat_space=MatrixSpace(GF(2),length-dim,length)
    while True:
        pc_mat=pc_mat_space.random_element(density=delta)
```

```
30          if pc_mat.rank()==length-dim:
31              break
32      code=codes.LinearCodeFromCheckMatrix(pc_mat)
33      code.my_parity_check_matrix=pc_mat
34      return code
35  class MyCode:
36      def __init__(self,length,dim,density,bitlength):
37          self.code=rnd_ldpc_code(length,dim,density)
38          self.length=self.code.length()
39          self.dim=self.code.dimension()
40          self.gen_mat=self.code.generator_matrix()
41          self.check_mat=self.code.my_parity_check_matrix.transpose()
42          self.bitlength=bitlength
43          self.elements=list(self.code)
44          gens=[bitvec2intrep(x,self.bitlength) for x in self.gen_mat.rows()
                ]
45          word_length=len(gens[0])
46          V=[[long(0) for i in range(word_length)]]
47          for i in range(self.dim):
48              V=V+[[long(v[j])^^long(gens[i][j]) for j in range(word_length)
                    ] for v in V]
49          self.intreps=V
50      def __repr__(self):
51          return "MyCode␣object␣of␣length␣%d␣and␣dimension␣%d"%(self.length,
                self.dim)
52      def encode(self, msg):
53          return msg*self.gen_mat
54      def decode_bitop(self,recv):
55          recv_intrep=bitvec2intrep(recv,self.bitlength)
56          n_recv_ints=len(recv_intrep)
57          wt_list=[sum(WEIGHT_TABLES[self.bitlength][c[i]^^recv_intrep[i]]
                for i in range(n_recv_ints)) for c in self.intreps]
58          c_min=wt_list.index(min(wt_list))
59          return self.elements[c_min]
60      def per_est(self,n_pkgs,p):
61          ret=0
62          for i in range(n_pkgs):
63              if self.decode_bitop(err_vec(self.length,p)).is_zero():
64                  ret+=1
65          return 1.0-ret.n()/n_pkgs
66  BITLENGTH=8
67  p=0.1
```

```
68 def tournament_qualification(length,dim,density,sample_size_for_simulation
       =100,N_in=100,N_out=10):
69     global BITLENGTH,p
70     my_codes=[MyCode(length,dim,density,BITLENGTH) for i in range(N_in)]
71     pers=[]
72     nr_of_zero_pers=0
73     for c in my_codes:
74         per=c.per_est(sample_size_for_simulation,p)
75         pers.append((c,per))
76         if per==0: nr_of_zero_pers+=1
77     pers_mean=mean([x[1] for x in pers])
78     pers_std=std([x[1] for x in pers])
79     return [sorted(pers,key=lambda x: x[1])[:N_out],pers_mean,pers_std]
80 def tournament_final(code_list,sample_size_for_simulation=1000):
81     global BITLENGTH,p
82     pers=[]
83     for c in code_list:
84         per=c[0].per_est(sample_size_for_simulation,p)
85         pers.append((c[0],per))
86     pers_mean=mean([x[1] for x in pers])
87     return min(pers,key=lambda x: x[1])
```

Irodalomjegyzék

[BF] Czédli Gábor, *Boole-függvények*, Polygon, Szeged, 2009.

[CACD] Petteri Kaski – Patric R. J. Östergard, *Classification Algorithms for Codes and Designs*, Springer-Verlag, Berlin, 2006.

[CT] André Neubauer – Jürgen Freudenberger – Volker Kühn, *Coding Theory: Algorithms, Architectures and Applications*, John Wiley & Sons, Chichester, 2007.

[FoECC] W. Cary Huffman – Vera Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University Press, New York, 2003.

[VG] Kiss György – Szőnyi Tamás, *Véges geometriák*, Polygon, Szeged, 2001.

[CCR] Yuriy Polyanskiy – H. Vincent Poor – Sergio Verdú, *Channel Coding Rate in the Finite Blocklength Regime*, IEEE Transactoins on Information Theory, 2010.

[S+16] *SageMath, the Sage Mathematics Software System (Version 7.1)*, The Sage Developers, 2016, `http://www.sagemath.org`.

Nyilatkozat

Alulírott Mezőfi Dávid Csaba kijelentem, hogy a diplomamunkában foglaltak saját munkám eredményei, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel. Tudomásul veszem, hogy diplomamunkámat a Szegedi Tudományegyetem könyvtárában a kölcsönözhető könyvek között helyezik el, és az interneten is nyilvánosságra hozhatják.

Szeged, 2016. május 13. .....

aláírás