Samoopravné kódy

Ladislav Láska

13. června 2011

Abstrakt

Tento text seznámí čtenáře se základními vlastnostmi samoopravných kódů, jejich konstrukce a také ukáže část teorie za nimi.

Obsah

1 Úvod

Představme si, že chceme komunikovat po nějakém nespolehlivém médiu (třeba telefonu), nebo na nějaké ukládat data (pevný disk), ale rádi bychom o žádná data nepřišli. K takovému účelu nám dobře poslouží **samoopravné kódy**, tedy kódy, které v sobě nesou nějakou redundantní informaci, aby při malém poškození bylo možno stále dekódovat původní data.

V této kapitole nastíním některé triviální kódy, o kterých každý čtenář jistě slyšel. Nejdříve si však musíme říct, co je to vlastně takový kód a jak ho budeme zapisovat.

Definice Data, která chceme poslat budeme nazývat **datová slova**. Data skutečně odeslaná kanálem jsou **kódová slova**. **Parametry kódu** rozumíme dvojici (n, k), kde n je délky kódového slova, k délka datového slova.

Konvence V dalším textu budeme předpokládat, že přenášíme slova v binárním zápise.

1.1 Možné chyby v datech

V datech se mohou dít chyby dvěma způsoby: vynulováním a změnou. Vynulování je, že se každý bit s nějakou pravděpodobností vynuluje (tedy místo poslaných dat se objeví samé nuly - nebo jiné konstanty). Změnou (v binárním schématu překlopením) se rozumí chyba, kde se bit s nějakou pravděpodobností překlopí. Pokud bychom měli jednotlivé bity vícehodnotové (tedy $\mathbb{F} \neq \mathbb{Z}_2$), bylo by zapotřebí složitější definice.

Konvence V dalším textu budeme předpokládat, že chyby se dějí překlopením.

1.2 Paritní kód

Nejjednodušším kódem je kód **paritní**. Funguje následujícím způsobem: Ke kódu doplní jedničku nebo nulu tak, aby byl počet jedniček sudý. Je zřejmé, že takový kód není samoopravný, ale spíš samo-chybudetekující. Umí detekovat právě sudý počet chyb, takže jeho použití je velmi limitováno. V dalších kapitolách se naučíme zkonstruovat efektivnější kódy s jenom o něco větší přidanou informací.

1.3 Opakovací kód

Druhý nejjednodušší kód je **opakovací**. Jednoduše přepošle datové slovo vícekrát za sebou (typicky $3\times$, aby šlo rozpoznat i opravit jednu chybu). To samozřejmě funguje docela dobře, nicméně je zcela neefektivní z hlediska velikosti přenášených dat.

2 Lineární kódy

Lineární kódy jsou takové, že lineární kombinace dvou kódových slova dá další kódové slovo.

Definice Lineární kód (n, k) kód je podprostor vektorového prostoru $\mathbb{F}_q^n$ dimenze k.

Konvence Protože uvažujeme pouze binární kódy, dále předpokládáme $q = 2$.

2.1 Paritní a opakovací kódy jsou lineární

Tvrzení Paritní a opakovací kódy jsou lineární kódy.

Důkaz Mějme nějaká kódová slova u, v. Ověříme, že jejich součet je také kódové slovo. Z toho okamžitě vyplyne, že i každá jiná lineární kombinace je kódové slovo a tedy kód je lineární. *Důkaz provádíme za předpokladu, že $\mathbb{F} = \mathbb{Z}_2$. Platí to ale obecně pro jiná tělesa (v případě parity je potřeba kód vhodně předefinovat).*

1. Paritní kód: Označme si u_d, v_d datové části slova a p, q jejich parity. Je snadné nahlédnout, že počet

jedniček ve výsledném slově w_d bude součet jedniček v obou slovech minus počet míst, kde jsou obě slova rovna jedna. Je zřejmé, že pokud podle součtu jedniček v obou slovech by byla parita $r = p + q$ (snadné ověřit z aritmetiky "sudá+lichá=lichá"atp). Také nahlédneme, že pokud byly jedničky na stejném místě v u i v, parita se nezmění (jedničky byly v obou, teď je 0), tedy parita výsledného slova má skutečně být r. Výsledné slovo je tedy také kódové slovo.

2. Opakovací kód: Triviálně, protože jednotlivé kopie slova se neovlivňují a tedy se každá kopie sečte stejně.

Máme tedy dva příklady lineárních kódů.

2.2 Generující matice

Z vlastností lineárních kódů si můžeme všimnout, že se každé kódové slovo dá vyjádřit jako lineární kombinace bázových slov. To samé platí i pro datová slova. Za cvičení si dokažte následující tvrzení:

Tvrzení Nechť u je datové slov a v je jemu odpovídající kódové slovo. Vyjádříme-li $u = a_1b_1 + ... + a_nb_n$, pak lze vyjádřit $v = a_1b'_1 + ... + a_nb'_n$, kde a_i jsou nějaké koeficienty a b_i, b'_i jsou navzájem odpovídající si bázové vektory ve vektorovém prostoru datových, resp. kódových slov.

Důkaz (cvičení)

Takovou operaci můžeme vyjádřit maticí přechodu mezi bázemi těchto vektorových prostorů.

Definice Matici G nazveme **generujicí** maticí (n, k) kódu, právě tehdy, když pro každé datové slovo u a jeho odpovídající kódové slovo v platí: $v = u \cdot G$. Generující matice je ve **standardním tvaru**, když ji lze psát jako $G = (I_k|M)$, kde I_k je jednotková matice $k \times k$ a M je libovolná matice $k \times n - k$.

Poznámka Je vidět, že definici lze "transponovat", a také se tak někdy používá. Nenechte se proto zmást. Takto mi to ale přijde intuitivnější, o to víc při odvozování kódů z polynomů (bude později).

2.3 Prověrková matice

Definice Matici H nazveme **prověrkovou** maticí (n, k) kódu, právě tehdy, když pro každé kódové slovo u a nekódové slovo v platí: $\mathbf{0} = H \cdot u$ a $\mathbf{0} \neq H \cdot v^T$.

Tvrzení Pro nějakou generující matici G tvaru $G = (I_k|M)$ řádu $n \times k$, je jí odpovídající prověrková matice je tvaru $H = (-M^T|I_{n-k})$.

Důkaz *(zatím bez důkazu)*

2.4 Příklady konstrukce

V následujících pár odstavcích si ukážeme, jak se jednoduše dají vytvořit generující matice pro jednoduché kódy z první kapitoly.

Stěžejním pozorováním je, že pokud sestavíme datovou bázi elementární, jednička na i-té pozici v datovém slově přičte do kódového slova i-tý řádek z matice.

2.4.1 Generující matice parity

Víme, že paritní matice upravuje počet jedniček na sudý. Chceme tedy, aby se elementární vektory báze (e_i) zobrazily na $(e_i|1)$. Ze znalosti násobení matic je tedy zjevné, že generující matice paritního (4,3) kódu

je:

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

2.4.2 Generující matice opakování

Podobnou metodou, jako jsme sestavili matici parity, můžeme sestavit matici opakování. Vektory elementární báze zobrazíme na 3 po sobě jdoucí, identické vektory: $(e_i) \to (e_i|e_i|e_i)$. Matice opakovacího (9,3) kódu bude vypadat:

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

2.5 Hammingova vzdálenost

Abychom mohli lépe měřit "výkonnost"kódových slov, zavedeme si pojem Hammingovy vzdálenosti a ukážeme jeho spojitost se schopnostmi korekce kódu.

Definice **Hammingova vzdálenost** d dvou slov je rovna počtu bitů, ve kterých se slova liší.

Věta Nechť C je lineární kód. Pak:

1. Kód odhaluje r a méně chyb, právě když je hammingova vzdálenost každých dvou kódových slov rovna alespoň $r + 1$.
2. Kód opravuje r a méně chyb, právě když je hammingova vzdálenost každých dvou kódových slov rovna alespoň $2r + 1$.

Důkaz Tvrzení je zřejmé z faktu, že hammingova vzdálenost tvoří metriku nad prostorem kódu C, dokonce nad každým jeho nadprostorem F. Můžeme si tedy představit okolo kódových slov koule poloměru r a chybná slova jako jejich prvky. Pak je zřejmé, že pokud žádné kódové slovo nepadne do koule jiného kódového slova, pro dané r kód chybu odhalí. Pokud navíc koule namají průniky, lze chybu i opravit.

2.6 Hammingův kód

Jak vidíme, paritní kód je sice úsporný, ale nedosahuje příliš dobrých výsledků. Opakovací kód zase dosahuje dobrých výsledků, ale cena za jeho použití je příliš veliká. Představíme si tedy Hammingův kód, mimochodem první ze samoopravných kódu.

2.6.1 Konstrukce hammingova kódu

Nejdříve si ukážeme, jak takový kód zkonstruovat. Budeme konstruovat obecný $(2^n - 1, 2^n - n - 1)$ kód. Mějme tedy kód dlouhý $2^n - 1$. Umístíme celkem n paritních bitů na místa rovna mocninám dvojky. Každý bit na pozici $2^i - 1$ bude pokrývat paritou 2^i dalších bitů (počínaje jím samým), stejný počet dalších bitů vynechá a opět stejný počet pokryje, což se opakuje až do konce slova. Názorně tabulka pro $(7, 4)$-kód.

Pozice	0	1	2	3	4	5	6
Význam	*p0*	*p1*	d0	*p3*	d1	d2	d3
Pokrývá paritou	•		•		•		•
		•	•			•	•
				•	•	•	•

Je snadno vidět, že kromě paritních bitů jsou všechny bity pokryty dvoumi paritními bity.

2.6.2 Minimální hammingova vzdálenost hammingova kódu

Věta Hammingův kód má minimální hammingovu vzdálenost 3.

Důkaz Sporem: Mějme dvě kódová slova, která se liší v méně než třech bitech. Alespoň jeden z bitů musí nutně být datový, protože jinak by nebyla kódová. Pokud se slova liší v právě jednom datovém bitu, určitě se musí lišit i ve dvou paritních. Pokud se slova liší ve dvou datových bitech, určitě se liší alespoň v jednom dalším paritním (je vidět například z obrázku). Pokud se slova liší ve třech datových bitech, není co dokazovat.

2.7 Minimální kódová vzdálenost a prověrková matice

Nyní ukážeme, jak snadno zjistit minimální hammingovu vzdálenost podle prověrkové matice.

Definice **Hammingovu váhu** slova u, $\mathbf{wt}(u)$, definujeme jako hammingovu vzdálenost od nulového vektoru.

Věta Minimální hammingova vzdálenost kódu d je rovna nejmenšímu počtu lineární závislých sloupců prověrkové matice H.

Důkaz Uvědomme si, že při násobení matice s nějakým přijatým vektorem v jednotlivé bity ve vektoru volí celé sloupce z původní matice. Přitom víme, že pro nějaké korektní kódové slovo c platí:

$$\mathbf{0} = H \cdot c^T = \sum_{i=1}^{n} H_i c_i^T \tag{3}$$

Kde H_i je i-tý sloupec H. Budeme-li uvažovat pouze $c_i \neq 0$, zbylé sloupce představují lineárně závislou množinu a tudíž musí být minimální vzdálenost d alespoň počet lineárně závislých sloupců.

Uvažujem dále nejmenší množinu lineárně závislých sloupců $\{H_j : j \in S\}$, kde S je indexová množina pro sloupce. Potom můžeme psát:

$$\sum_{i=1}^{n} H_i c_i^T = \sum_{j \in S}^{n} H_j c_j^T + \sum_{j \notin S} H_j c_j^T = \mathbf{0} \tag{4}$$

Nyní uvažujem vektor c' takový, že $c'_j = 0 \Leftrightarrow j \notin S$. Takový je určitě kódovým slovem, protože výše zmíněná suma pro něj platí. Takže určitě $d \leq \mathbf{wt}(c')$, což je ale nejmenší počet lineárně závislých sloupců H a tvrzení platí.

TODO: Ověřit, trošku formálněji popsat.

2.8 Syndromové dekódování

Zatím jsme se naučili kódy konstruovat. Také jsme si ukázali generující a prověrkovou matici. Nyní si ukážeme, jak efektivně dekódovat chybně přenesená slova (pro kód, který korekci umožňuje).

Obecně se nám bude hodit k dekódování tabulka syndromů a možných slov, který jim odpovídají. Tu sestavíme následovně: syndrom **0** má každé kódové slovo. Dále pro každý syndrom a kódové slovo najdeme slovo odpovídající danému syndromu od něj odvozené. To uděláme přičtením syndromu ke kódovému slovu (a doplníme nulami na stejnou velikost).

TODO: Proč? Slovák si myslí, že je to pravda. Já nevím, proč by to pravda byla v obecném případě. Nejspíš je to tím, že takové slovo představuje například zbytek po dělení v polynomiálním kódu, ale neumím to dokázat.

TODO: Příklad.

Je snadné vypozrovat, že v odečtením libovolných dvou slov na řádku dává kódové slovo - důkaz nechám jako snadné cvičení čtenářovi. Každá jednička v těchto slovech tedy vyjadřuje, kolik bitů je potřeba změnit

ve slově, aby se z něj stalo platné kódové slovo. Označme si tedy slova s nejmenším počtem jedniček v daném řádku a nazvěme je **vedoucími reprezentanty**. Při odečtení tohoto slova od přijatého slova dostáváme platné kódové slovo a to právě odpovídající vyslanému slovu, pokud nedošlo k příliš mnoha chybám.

3 Polynomiální kódy

Už při konstrukci hammingova kódu jsme narazili na problém, že popis nemusel být zrovna triviální. V dalších odstavcích ukáži, jak se dají samoopravné kódy generovat pomocí polynomů.

Definice Nechť $p(x) = a_0x^0 + ... + a_{n-k}x^{n-k} \in \mathbb{Z}_2[x]$ je polynom a $a_0 \neq 0 \neq a_{n-k}$. Pak **polynomiální kód generovaný polynomem** $p(x)$ je kód, jehož slova jsou polynomy stupně menšího než n a bezezbytku dělitelné $p(x)$.

Zatím zahoďmě stranou, jak dobré takové kódování bude. Podívejme se, jak zkonstruujeme kódové slovo z datového. Nejdříve je potřeba si uvědomit, že každé datové slovo délky k můžeme zapsat jako polynom stupně maximálně k, tedy bity slova jsou jeho koeficienty. Takový polynom většinou nebude dělitelný $p(x)$, takže ho jím musíme pronásobit. Kódové slovo pro datové slovo $m(x)$ tedy bude polynom $v'(x) = p(x)m(x)$. Všimneme si ale, že to má jistou nevýhodu. V takovém kódovém slově není přímo obsažena zpráva. Zprávu tedy zakódujeme jinak: $v(x) = x^{n-k}m(x)+r(x)$, kde $r(x) := x^{n-k}m(x)$ **div** $p(x)$, aby byl náš polynom dělitelný. Také lze snadno vypozorovat, že zbytek nebude mít řád větší než $n - k$, tedy nižší koeficienty budou představovat "paritu", vyšší koeficienty potom samotnou zprávu.

3.1 Linearita polynomiálních kódů

Tvrzení Každý polynomiální kód je lineární.

Důkaz Mějme polynom $p(x)$ generující nějaký kód C a jeho dvě kódová slova $v(x), w(x)$. Je zřejmé, že libovolná lineární kombinace $a \cdot v(x) + b \cdot w(x)$ je dělitelná $p(x)$, pokud $v(x)$ i $w(x)$ jsou dělitelné $p(x)$, což z předpokladu jsou a kód je tedy lineární.

3.2 Dekódování

Dekódování je velmi jednoduché, prakticky je to otázka dělení polynomů. Korektně přenesená data se poznají tak, že jejich zbytek po dělení příslušným polynomem je 0. Nekorektní data lze dekódovat například syndromovým dekódováním (protože každý polynomiální kód je lineární).

3.3 Od polynomu k maticím

Protože jsme si ukázali, že polynomiální kódy jsou lineární, musí také pro každý polynomiální kód existovat generující a prověrková matice. Jak ji ale zjistit?

Půjdeme na to od lesa. Protože víme, že jsou lineární kódy lineární (uzavřené na sčítání), stačí nám zjistit, kam se zobrazí bázové vektory. Těmi jsou a_ix^i. Nechť tedy podle v_i je sloupcový vektor, na který se zobrazila báze a_ix^i, tedy po vynásobení $p(x)$. Generující matice je pak sestavena spojením těchto vektorů. Prověrkovou matici lze sestavit podle tvrzení o vztahu prověrkové a generující matice.

Pozor ale na to, že pokud chceme mít matici ve standardním tvaru, je potřeba nejen násobit $p(x)$, ale provést fintu se zbytkem, jako jsme udělali při zavádění kódu.

3.4 Polynom pro paritu

Pozorování Polynom $p(x) = 1 + x$ generuje $(n, n - 1)$ kód parity.

Důkaz Jak to zjistíme? Buď si odvodíme pravidlo, pro dělitelnost polynomem (což v tomto případě lze - polynom $q(x)$ je dělitelný $1 + x$ právě tehdy, když $q(1) = 0$), nebo si můžeme zkusit postavit generující matici ve standardním tvaru. Podíváme se tedy na generující matici pro $(4, 3)$ kód. Musíme tedy zobrazit

bázové vektory $1, x, x^2$ pomocí našeho polynomu. Také provedeme zmíněnou fintu a vynásobíme si bázi x, takže parita bude koeficient a_0. K tomu je zapotřebí vypočítat zbytky po dělení p(x):

$$x = 1 \mod x + 1 \tag{5}$$
$$x^2 = 1 \mod x + 1 \tag{6}$$
$$x^3 = 1 \mod x + 1 \tag{7}$$

Při dělení na papíře vyjde -1. Nezapomínejte na to, že pracujeme v $\mathbb{Z}_2$. Vektory tedy zobrazíme takto (tedy vynásobení x a přičtení zbytku):

$$1 \sim x + 1 = a(x) \tag{8}$$
$$x \sim x^2 + 1 = b(x) \tag{9}$$
$$x^2 \sim x^3 + 1 = c(x) \tag{10}$$
$$\tag{11}$$

A přepíšeme do matice. Je samozřejmé, že můžeme sloupce zpřeházet. Většinou jsme zvyklí psát paritu na konec (naše schéma jí má jako konstantní člen, takže by byla na začátku). U takto jednoduchého kódu ji tedy můžeme přehodit na konec:

$$G = \begin{pmatrix} a_1 & a_2 & a_3 & a_0 \\ b_1 & b_2 & b_3 & b_0 \\ c_1 & c_2 & c_3 & c_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{12}$$

Jak si můžeme všimnout, generující matice je stejná, jako matice paritního kódu na začátku textu.