Samoopravné kódy

aneb na nás si chyby nepřijdou

(verze 1.2 z 2021-04-15)

0. Dým nad prérií

Dva indiáni si vyměňují zprávy kouřovými signály: Aleshanee na jednom kopci, Bodaway na druhém. Jsouce indiány pokrokovými, užívají binární abecedu: malý obláček kouře je nula, velký jednička. Jenže nad prérií vane vítr neposeda a s obláčky si pohrává: tu a tam maličký nafoukne či velký zmenší. Občas proto indiáni přijmou opačný bit, než byl původně vyslán. Naštěstí aspoň platí zákon zachování počtu obláčků: nikdy se jeden obláček nerozdělí na dva, ani nesplynou dva do jednoho.

Naši indiáni si ale poradí: budou tvořit takové zprávy, aby se na nich poznalo, že během přenosu došlo k chybě, a zprávu prostě pošlou znovu. Nebo aby dokonce poznali, kde přesně chyba nastala, a tím pádem ji dokázali opravit. Pochopitelně se jim to může dařit jen tehdy, je-li chyb omezený počet – kdyby vítr výtržník mohl obláčky měnit neomezeně, nesla by užitečnou informaci jen délka zprávy.

1. Kódování a dekódování

Situaci popíšeme formálněji:

- Aleshanee chce vyslat nějakou zprávu (posloupnost bitů) $x_1, \ldots, x_n$.
- Pro zvýšení odolnosti ji *zakóduje* do nějaké delší zprávy $y_1, \ldots, y_m$ a tu odešle.
- Vítr cestou změní až k bitů dle vlastního výběru.
- Bodaway přijme nějakou modifikovanou zprávu $\hat{y}_1, \ldots, \hat{y}_m$.
- Nakonec Bodaway zprávu *dekóduje.* To může znamenat:
  1. *Detekuje chyby* – pokud zakódovaná zpráva přišla poškozená, pozná to. A pokud přišla v pořádku, získá původní zprávu $x_1, \ldots, x_n$.
  2. *Opraví chyby* – ať už zakódovaná zpráva přišla poškozená nebo v pořádku, získá původní zprávu $x_1, \ldots, x_n$.

  Chování pro víc než k chyb nedefinujeme.

Máme tedy 3 parametry: délku původní zprávy n, délku zakódované zprávy m a maximální počet chyb k, které umíme detekovat/opravit. Samozrejmě bychom chtěli zprávy prodlužovat co nejméně, tedy aby pro dané n a k bylo m co nejmenší.

Příklad: *Paritní kód* funguje tak, že ke zprávě přidáme jeden bit tak, aby celkový počet jedniček ve zprávě byl sudý. Tomuto bitu se říká *paritní bit* a jeho hodnota je nutně $(x_1 + \ldots + x_n) \bmod 2$. Všimněte si, že tento kód detekuje chybu v libovolném jednom bitu: kdykoliv se změní jediný bit, celkový počet jedniček se změní mezi lichým a sudým. Máme tedy detekční kód s parametry $m = n + 1$ a $k = 1$.

Příklad: Pokud $m = n$, pak evidentně nejsme schopni poznat žádnou chybu. Jelikož různé původní zprávy musíme zakódovat do různých vyslaných zpráv (jinak by dekódování nebylo jednoznačné), musí být kódovací zobrazení prosté. Jenže původních i zakódovaných zpráv je stejně, takže to musí být bijekce. Tudíž zakódováním může vzniknout každá n-tice bitů, a proto změna v jednom bitu vyrobí ze zprávy jinou korektní zprávu. Takže paritní kód je nejlepší možný.

Úkol 1.1: Vymyslete kód, který detekuje 2 chyby.

Úkol 1.2: Vymyslete kód, který opraví 1 chybu.

Úkol 1.3: Vymyslete kód, který detekuje 3 chyby.

Úkol 1.4: Vymyslete kód, který detekuje až 100 chyb.

Úkol 1.5: Vymyslete kód, který opraví až 100 chyb.

Úkol 1.6: Uvažme trochu jinou situaci: když přijde poškozený bit, poznáme, že je poškozený (představte si kaňku na papíře). Detekce chyb je v takovém případě triviální, ale navrhněte kód, který umí 1 chybu opravit. Tomu se říká *mazací kód*.

Dál pokračujte až poté, co vyřešíte aspoň část úloh z této kapitoly.

2. Geometrie zpráv

Kódování, změny zpráv po cestě (těm budeme říkat prostě *šum*) a dekódování můžeme také popisovat geometricky. Na zakódované zprávy $y_1, \ldots, y_m$ se můžeme dívat jako na *body* v nějakém m-rozměrném prostoru, jehož každá souřadnice nabývá pouze hodnot 0 a 1. Tomuto prostoru se říká *Hammingův prostor* a dal by se popsat jako m-rozměrná krychle.

Pro dvě zprávy $\mathbf{a} = a_1, \ldots, a_m$ a $\mathbf{b} = b_1, \ldots, b_m$ můžeme dokonce měřit jejich vzdálenost. Říká se jí *Hammingova vzdálenost* a značí se $\varrho(\mathbf{a}, \mathbf{b})$. Definujeme ji jako počet pozic i, na kterých je $a_i \neq b_i$.

Úkol 2.1: Hammingova vzdálenost se chová, jak se na vzdálenost sluší a patří (matematik by řekl, že je to *metrika*). Je vždy nezáporná. Také $\varrho(\mathbf{a}, \mathbf{b}) = 0$ právě tehdy, když $\mathbf{a} = \mathbf{b}$. Je symetrická: $\varrho(\mathbf{a}, \mathbf{b}) = \varrho(\mathbf{b}, \mathbf{a})$. Dokažte, že splňuje trojúhelníkovou nerovnost: $\varrho(\mathbf{a}, \mathbf{c}) \leq \varrho(\mathbf{a}, \mathbf{b}) + \varrho(\mathbf{b}, \mathbf{c})$ pro každé $\mathbf{a}$, $\mathbf{b}$ a $\mathbf{c}$.

Pokud vyšleme nějakou zprávu $\mathbf{y}$ a nastane nejvýše k chyb, můžeme přijmout přesně ty zprávy $\hat{\mathbf{y}}$, pro něž je $\varrho(\mathbf{y}, \hat{\mathbf{y}}) \leq k$. To si můžeme představit jako kouli v Hammingově prostoru. (Pozor, má trochu divný tvar :))

Zdá se tedy, že pro schopnost detekovat/opravit chyby je zásadní, aby jednotlivé zakódované zprávy od sebe byly dostatečně daleko.

Definice: Nechť C je množina všech zpráv, které mohou vzniknout zakódováním. Budeme jí říkat prostě *kód* a příslušným zprávám (bodům Hammingova prostoru) *kódová slova.*

Všimněte si, že na tom, která původní zpráva odpovídá kterému kódovému slovu, moc nesejde – vhodné zobrazení může usnadnit algoritmy pro kódování a dekódování, ale schopnost poradit si s chybami tím není nijak ovlivněna. Víme nicméně, že kódových slov je stejně jako původních zpráv, tedy $|C| = 2^n$.

Definice: Pro kód C definujeme jeho *kódovou vzdálenost* d jako minimum z Hammingových vzdáleností kódových slov. Tedy $d = \min\{\varrho(\mathbf{a}, \mathbf{b}) \mid \mathbf{a}, \mathbf{b} \in C \wedge \mathbf{a} \neq \mathbf{b}\}$.

Definice: Aby se nám o vlastnostech kódů dobře mluvilo, budeme říkat (n, m, d)-*kód* každému kódu, který má parametry n a m a kódovou vzdálenost *alespoň* d.

Úkol 2.2: Jak souvisí maximální počet chyb k, které jsme schopni *detekovat*, s kódovou vzdáleností d?

Úkol 2.3: Jak souvisí maximální počet chyb k, které jsme schopni *opravit*, s kódovou vzdáleností d?

Úkol 2.4: Jak souvisí maximální počet kaněk k, které jsme schopni *opravit*, s kódovou vzdáleností d?

Úkol 2.5: Podívejte se na kódy (množiny kódových slov) z minulé kapitoly. Jaké jsou jejich kódové vzdálenosti? Nedovedou ve skutečnosti detekovat/opravit víc chyb, než jsme si mysleli?

Úkol 2.6: Dokažte o některém ze sestrojených kódů, že je nejlepší možný. Tedy že pro dané n a d nejde dosáhnout menšího m.

Úkol 2.7: Ukažte, jak z libovolného (n, m, d)-kódu pro $d > 1$ vyrobit $(n, m-1, d-1)$-kód.

Úkol 2.8: Ukažte, jak z libovolného (n, m, d)-kódu pro liché d vyrobit $(n, m+1, d+1)$-kód. Mohlo by to souviset s úkolem 1.3.

3. Lineární kódy

Některé kódy se chovají jako vektorové (pod)prostory. Představme si Hammingův prostor jako vektorový prostor $\mathbb{Z}_2^m$, tedy prostor dimenze m nad tělesem $\mathbb{Z}_2$. Násobení vektoru skalárem je nudné (můžeme násobit jen nulou nebo jedničkou), ale získali jsme užitečné sčítaní vektorů (po složkách, každá složka modulo 2, tedy vlastně XOR posloupností bitů).

Definice: O kódu C řekneme, že je *lineární*, pokud pro všechna $\mathbf{a}, \mathbf{b} \in C$ je $\mathbf{a}+\mathbf{b} \in C$.

Úkol 3.1: Ukažte, že (neprázdný) lineární kód nutně obsahuje nulový vektor $\mathbf{0}$ (zprávu ze samých nul). Tím pádem každý lineární kód je vektorovým podprostorem.

Definice: *Hammingova váha* vektoru $\mathbf{a}$ se značí $w(\mathbf{a})$ a je rovna počtu jedničkových bitů. Tedy platí $w(\mathbf{a}) = \varrho(\mathbf{a}, \mathbf{0})$.

Úkol 3.2: Které z našich kódů jsou lineární?

Úkol 3.3: Dokažte, že pro každý lineární kód platí, že $d = \min\{w(\mathbf{y}) \mid \mathbf{y} \in C, \mathbf{y} \neq \mathbf{0}\}$. (To se může hodit k výpočtu kódových vzdáleností u těch kódů, kde jsme to neuměli přímo.)

4. Polynomiální kódy

Hezké kódy se dají odvodit z polynomů. Na to ale nejprve musíme rozšířit abecedu: místo nul a jedniček budeme posílat symboly z nějaké konečné množiny Σ. Této množině budeme říkat *abeceda* a její velikost značit q.

Naše úvahy o kódových vzdálenostech stále fungují (rozmyslete) a pokud Σ je těleso, můžeme pořád pracovat s linearitou kódů. Značení kódů rozšíříme přirozeným způsobem:

Definice: $(n, m, d)_q$-kód říkáme kódu nad q-prvkovou abecedou, který má q^n kódových slov délky m a kódovou vzdálenost d.

Úkol 4.1: Jak vypadá analogie paritního kódu pro obecnou abecedu? Jaké má parametry?

Úkol 4.2: Mějme nějaký $(n, m, d)_q$-kód pro $q = 2^t$. Přeložíme ho do binární abecedy tak, že každý symbol zapíšeme ve dvojkové soustavě pomocí t bitů. Jak se změní parametry kódu?

Úkol 4.3: Uvažujme Hammingův prostor dimenze m nad q-prvkovou abecedou. Kolik bodů obsahuje koule o poloměru r?

Od této chvíle až do konce kapitoly bude abeceda vždy nějaké konečné těleso. Připomeneme si pár tvrzení o polynomech, která platí nad každým tělesem.

Odbočka: Musíme rozlišovat různé způsoby, jak se polynomy mohou rovnat. $P \equiv Q$ značí formální rovnost: polynomy P a Q mají stejné členy (až na pořadí a případné

členy s nulovými koeficienty). Naproti tomu $P = Q$ znamená rovnost jako funkce: při dosazení jakéhokoliv x bude $P(x) = Q(x)$.

Lemma: Polynom $P(x) \equiv p_0x^0 + p_1x^1 + \ldots + p_tx^t$ má nejvýše t kořenů, kromě případu, kdy $P \equiv 0$.

Náčrtek důkazu: Pokud a je kořenem polynomu P, pak je polynom P dělitelný polynomem $x - a$. Tedy má-li P kořeny $a_1, \ldots, a_k$, platí $P(x) \equiv (x - a_1) \cdot \ldots \cdot (x - a_k) \cdot Q(x)$ pro nějaký polynom $Q \not\equiv 0$. Tento součin ovšem obsahuje x v alespoň k-té mocnině, takže k nemůže být větší než t.

Lemma: Nechť P a Q jsou polynomy stupně nejvýš t, které se shodnou ve více než t bodech, tedy $P(a_i) = Q(a_i)$ pro navzájem různá čísla $a_0, \ldots, a_t$. Potom $P \equiv Q$.

Důkaz: Uvažme polynom $R = P - Q$. Jistě je $R(a_i) = P(a_i) - Q(a_i) = 0$ pro všechna i. Takže R má alespoň $t + 1$ kořenů, ovšem zároveň stupeň nejvýše t. Podle předchozího lemmatu tedy musí být $R \equiv 0$, takže $P \equiv Q$..

Konstrukce: Vezmeme různé polynomy P a Q stupně t nad q-prvkovým tělesem. Sestrojíme jejich „grafy“ $(P(a_1), \ldots, P(a_q))$ a $(Q(a_1), \ldots, Q(a_q))$, kde $a_1, \ldots, a_q$ je nějaké očíslování prvků tělesa. Podle předchozího lemmatu tyto grafy mají společných nejvýše t bodů, takže jejich Hammingova vzdálenost činí aspoň $q - t$.

Úkol 4.4: Využijte předchozí konstrukci k výrobě kódů s velkou kódovou vzdáleností. Abecedu si zvětšete podle potřeby.