Komprese dat

Jan Outrata

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

přednášky

Obsah

Anotace

V předmětu jsou představeny základní i moderní metody bezeztrátové komprese dat a ztrátové komprese multimediálních dat.

Literatura

- Sayood K.: *Introduction to Data Compression*, Fourth Edition. Morgan Kaufmann, 2012. ISBN 978-0124157965
- Salomon D., Motta G.: *Handbook of Data Compression*, 5th edition. Springer, 2010. ISBN 978-1848829022
- Salomon D.: *Data Compression: The complete Reference*, 4th edition. Springer, 2006. ISBN 978-1846286025
- Hankerson D. C., Harris G. A., Johnson P. D.: *Introduction to Information Theory and Data Compression*, Second Edition (Applied Mathematics). Chapman and Hall/CRC, 2003. ISBN 978-1584883135
- Sayood K.: *Lossless compression handbook*. Academic Press, 2003. ISBN 0126208611

Úvod

Komprese dat

= zmenšení velikosti reprezentace obsahu/dat – jeden z účelů kódování dat, (experimentální) vědní obor

Dvě fáze:

1. identifikování a modelování struktury dat s vynecháním redundancí
   - struktura např. opakování vzorů, statistická $\approx$ frekvence/četnost vzorů, korelace mezi vzory, vzory elementární symboly nebo skupiny symbolů, také např. daná zdrojem dat $\rightarrow$ modelování zdroje a syntéza dat (zvuk)
   - také různé modely pro různé části dat
2. kódování dat podle modelu
   - plus případně kódování (části) modelu
   - také predikce hodnoty dle modelu a kódování rozdílu (residua)
   - typicky binární kód

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b}/\text{číslo} = 48\,\text{b}$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b/číslo} = 48\,\text{b}$
2. 7 různých čísel ve dvojkové soustavě $\Rightarrow 3\,\text{b/číslo} = 36\,\text{b}$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b/číslo} = 48\,\text{b}$
2. 7 různých čísel ve dvojkové soustavě $\Rightarrow 3\,\text{b/číslo} = 36\,\text{b}$
3. častější číslo kratší kód $\rightarrow 2 \times 2,\ 1 \times 4,\ 1 \times 6,\ 3 \times 7,\ 2 \times 10,\ 2 \times 11,\ 1 \times 14 \rightarrow$ **0I** pro 7, **III** pro 11, **II0** pro 10, **I0I** pro 2, **I00** pro 14, **000** pro 4 a **00I** pro 6 $\Rightarrow 33\,\text{b} = 2.75\,\text{b/číslo}$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b/číslo} = 48\,\text{b}$
2. 7 různých čísel ve dvojkové soustavě $\Rightarrow 3\,\text{b/číslo} = 36\,\text{b}$
3. častější číslo kratší kód $\rightarrow 2 \times 2,\ 1 \times 4,\ 1 \times 6,\ 3 \times 7,\ 2 \times 10,\ 2 \times 11,\ 1 \times 14 \rightarrow \mathbf{0I}$ pro 7, $\mathbf{III}$ pro 11, $\mathbf{II0}$ pro 10, $\mathbf{I0I}$ pro 2, $\mathbf{I00}$ pro 14, $\mathbf{000}$ pro 4 a $\mathbf{00I}$ pro $6 \Rightarrow 33\,\text{b} = 2.75\,\text{b/číslo}$
4. kódování opakování čísla $\rightarrow \mathbf{0}$ pro žádné, $\mathbf{I0}$ pro jedno a $\mathbf{II}$ pro dvě $\Rightarrow$ $7 \times 3 + 11 = 32\,\text{b} = 2.\overline{6}\,\text{b/číslo}$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b/číslo} = 48\,\text{b}$
2. 7 různých čísel ve dvojkové soustavě $\Rightarrow 3\,\text{b/číslo} = 36\,\text{b}$
3. častější číslo kratší kód $\rightarrow 2 \times 2,\ 1 \times 4,\ 1 \times 6,\ 3 \times 7,\ 2 \times 10,\ 2 \times 11,\ 1 \times 14 \rightarrow \mathbf{0I}$ pro 7, $\mathbf{III}$ pro 11, $\mathbf{II0}$ pro 10, $\mathbf{I0I}$ pro 2, $\mathbf{I00}$ pro 14, $\mathbf{000}$ pro 4 a $\mathbf{00I}$ pro $6 \Rightarrow 33\,\text{b} = 2.75\,\text{b/číslo}$
4. kódování opakování čísla $\rightarrow \mathbf{0}$ pro žádné, $\mathbf{I0}$ pro jedno a $\mathbf{II}$ pro dvě $\Rightarrow$ $7 \times 3 + 11 = 32\,\text{b} = 2.\bar{6}\,\text{b/číslo}$
5. malé rozdíly mezi sousedními čísly $\rightsquigarrow$ predikce $\rightarrow d_1 = x_1 = 2$, $d_i = x_i - x_{i-1} = 0, 2, 2, 1, 0, 0, 3, 0, 1, 0, 3 \rightarrow 4\,\text{b} + 2\,\text{b/číslo} \Rightarrow 26\,\text{b} = 2.1\bar{6}\,\text{b/číslo}$

Komprese dat

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

1. číslo ve dvojkové soustavě $\Rightarrow 4\,\text{b/číslo} = 48\,\text{b}$
2. 7 různých čísel ve dvojkové soustavě $\Rightarrow 3\,\text{b/číslo} = 36\,\text{b}$
3. častější číslo kratší kód $\rightarrow 2 \times 2,\ 1 \times 4,\ 1 \times 6,\ 3 \times 7,\ 2 \times 10,\ 2 \times 11,\ 1 \times 14 \rightarrow \mathbf{0I}$ pro 7, $\mathbf{III}$ pro 11, $\mathbf{II0}$ pro 10, $\mathbf{I0I}$ pro 2, $\mathbf{I00}$ pro 14, $\mathbf{000}$ pro 4 a $\mathbf{00I}$ pro $6 \Rightarrow 33\,\text{b} = 2.75\,\text{b/číslo}$
4. kódování opakování čísla $\rightarrow \mathbf{0}$ pro žádné, $\mathbf{I0}$ pro jedno a $\mathbf{II}$ pro dvě $\Rightarrow$ $7 \times 3 + 11 = 32\,\text{b} = 2.\bar{6}\,\text{b/číslo}$
5. malé rozdíly mezi sousedními čísly $\rightsquigarrow$ predikce $\rightarrow d_1 = x_1 = 2$, $d_i = x_i - x_{i-1} = 0, 2, 2, 1, 0, 0, 3, 0, 1, 0, 3 \rightarrow 4\,\text{b} + 2\,\text{b/číslo} \Rightarrow 26\,\text{b} = 2.1\bar{6}\,\text{b/číslo}$
6. vztah mezi čísly $\rightsquigarrow$ predikce $\rightarrow \hat{x}_i = i + 1 \rightarrow$ $d_i = x_i - \hat{x}_i = 0, -1, 0, 1, 1, 0, -1, 1, 0, 0, -1, 1 \rightarrow \mathbf{0}$ pro 0, $\mathbf{I0}$ pro -1 a $\mathbf{II}$ pro $1 \Rightarrow$ $19\,\text{b} = 1.58\bar{3}\,\text{b/číslo}$

Komprese dat

- využití („zneužití“) omezení reprodukční techniky a příjemce obsahu (člověka) pro vynechání nevyužitelných informací (obraz, video, zvuk)
- data . . . znaky textu, vzorky obrazu (body) a videa (body v čase), zvuku (úrovně v čase), aj., digitální (digitalizovaná) forma, narůstající objem – např. obraz foto 10 Mpx 24 bpp ~ 30 MB, video HDTV 1920×1080 12 bpp, 25 fps ~ 590 Mb/s, zvuk CD 44.1 kHz, 16 bps, stereo ~ 1.3 Mb/s
- vývoj úložných a přenosových technologií nestačí, navíc (fyzikální) omezení
- umožnění tzv. multimediální revoluce – komprese textu, obrazu, videa, zvuku při uložení a přenosu

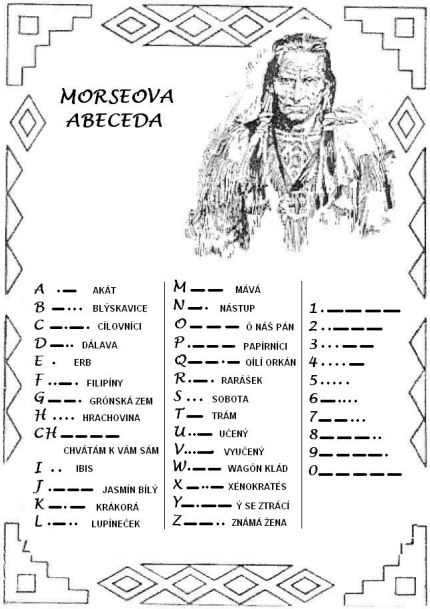
$\rightarrow$ všudypřítomná – počítače, spotřební elektronika, komunikační a distribuční sítě, . . .

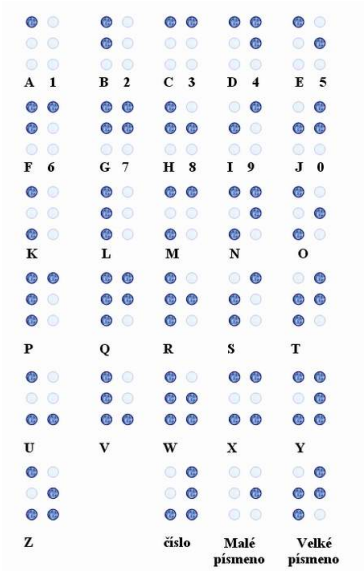
Příklady z minulosti

- morseovka: písmena (a číslice a interpunkce) kódována do posloupností teček a čárek, častější (e, t) kratšími pro zmenšení průměrné délky textu
- Braillovo písmo: do 2×3 matice teček kódována písmena (a číslice, interpunkce aj., Grade 1) a častá slova (a jejich zkratky, Grade 2)

Obrázek: Morseovka a Braillovo písmo

Komprese dat





Kompresní techniky a metody

= dva algoritmy: kompresní pro kompresi originálních dat na komprimovaná a dekompresní (rekonstrukční) pro dekompresi komprimovaných dat na dekomprimovaná (rekonstruovaná)

- standardy: ISO, ITU-T aj.

Bezeztrátové (lossless)

= dekomprimovaná data stejná jako data originální = žádná ztráta informace v datech

- např. pro text, programové (binární) soubory, citlivé záznamy (bankovní, zdravotní), nereprodukovatelná data (snímky v čase) aj.
- statistické: Huffmanovo a aritmetické kódování
- kontextové: PPM
- slovníkové: LZ*
- jiné: BWT, ACB, obrazové (JPEG-LS, JBIG)

Kompresní techniky a metody

Ztrátové(lossy)

= při kompresi vynechání nějaké informace v originálních datech → dekomprimovaná data (obecně) odlišná od originálních dat = ztráta informace z originálních dat – zkreslení dat

- vyšší míra komprese než u bezeztrátových za cenu vyšší míry zkreslení dat
- např. pro obraz, video, zvuk (hudba, řeč) – zkreslení dat vede k artefaktům při reprodukci obsahu
- vzorkování a kvantizace: skalární a vektorová
- diferenční kódování: DPCM, delta modulace
- transformační a podpásmové kódování: Fourierova, Z a kosinová transformace, wavelety
- aplikace: obraz – JPEG, fraktály, video – H.*, MPEG, zvuk – MDCT, G.*, MPEG, LPC, CELP

Míry kompresních algoritmů

- asymptotická časová a paměťová složitost algoritmů komprese a dekomprese
- experimentální časová a paměťová náročnost algoritmů – jejich implementací na referenčních datech
- míry komprese
  - kompresní poměr (compression ratio) = poměr velikosti originálních a komprimovaných dat, také jako procento velikosti komprimovaných dat z velikosti originálních dat
  - compression rate = průměrná velikost komprimovaných dat na vzorek originálních dat, např. pixel u obrazu – bitů/pixel, sekunda u videa a zvuku – bitů/s
  - na referenčních datech
- míry zkreslení (distortion) – rozdíl mezi originálními a dekomprimovanými daty, více způsobů měření „přesnosti (fidelity)“ a „kvality“ obsahu, viz dále, na referenčních datech

Modely dat

Fyzický

= popis zdroje dat – např. měřených, popis měřidla

- u ztrátové komprese zvuku (řeči) – popis syntezátoru a syntéza dat
- obecně příliš složitý nebo nemožný

Pravděpodobnostní model

= empiricky zjištěný statistický popis zdroje dat

- pro statistické a kontextové bezeztrátové kompresní metody
- ignorantní: výskyt každé hodnoty na výstupu zdroje dat je nezávislý na výskytu ostatních hodnot a je se stejnou pravděpodobností – nejjednodušší
- dostupná pravděpodobnost výskytu nezávisle se vyskytujících hodnot
- pravděpodobnost:
  - frekvence/četnost výskytu výsledku experimentu (hodnot na výstupu zdroje dat) – n opakování experimentu, n_i výskytů výsledku $\omega_i \in \Omega, i \in \{1, 2, \ldots, N\}$ (Ω ... prostor výsledků (sample space)) $\rightarrow$ frekvence/četnost výskytu výsledku ω_i: $f(\omega_i) = f_i = \frac{n_i}{n} =$ přibližná hodnota/odhad pro pravděpodobnost výskytu výsledku ω_i: $P(\omega_i) = p_i = \lim_{n\to\infty} \frac{n_i}{n}$, událost (event) $A \subseteq \Omega$, výskyt události = výskyt kteréhokoliv výsledku události, $f(A) \geq 0 \Rightarrow P(A) \geq 0$ (1), $P(\Omega) = 1$ (2), $B \subseteq \Omega, A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B)$ (3), $\sum_i P(\omega_i) = 1$

Modely dat

Pravděpodobnostní model

- pravděpodobnost:
  - míra víry (belief) v událost – a priori pravděpodobnost $P(A)$ události A před výskytem události (získání informace) B, a posteriori pravděpodobnost $P(A|B)$ po/za předpokladu, sdružená (joint) pravděpodobnost $P(A, B)$ výskytu obou událostí A, B, Bayesovo pravidlo $P(A|B) = \frac{P(A,B)}{P(B)}$, (statisticky) nezávislé události $\ldots P(A, B) = P(A)P(B)$, tj. při $P(A|B) = P(A)$, pro případy, kdy experiment není možné provést
  - míra („velikost“) události (jako množiny) – jako jiné míry (1) a (3), normalizace (2) = axiomy, z nich např. $P(\bar{A}) = 1 - P(A)$, $P(\emptyset) = 0$, $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ aj., pro nediskrétní prostor výsledků

Modely dat

Pravděpodobnostní model

- náhodná proměnná/veličina: měřitelné $X : \Omega \mapsto \mathbb{R}$ ($\mathbb{R}$ ... obor reálných čísel), realizace $X(\omega) = x$, např. $P(X(\omega) \leq x) = P(X \leq x)$, diskrétní a spojitá
- rozdělení pravděpodobnosti: distribuční funkce/kumulovaná pravděpodobnost (cumulative distribution function) $F_X(x) = P(X \leq x)$, $x_1 \geq x_2 \Rightarrow F_X(x_1) \geq F_X(x_2)$, $P(X = x) = F_X(x) - F_X(x^-)$ pro $F_X(x^-) = P(X < x)$, rozdělení/distribuce/hustota pravděpodobnosti (probability distribution/density function) $f_X(x)$ ... diference/derivace $F_X(x)$ pro diskrétní/spojitou X, pro diskrétní typicky $f_X(x) = P(X = x)$, např. binomické, Poissonovo, uniformní, normální (Gaussovo), aj.
- sdružená (joint) distribuční funkce $F_{X_1X_2\ldots X_n}(x_1, x_2, \ldots, x_n) = P(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_n \leq x_n)$, sdružené rozložení pravděpodobnosti $f_{X_1X_2\ldots X_n}(x_1, x_2, \ldots, x_n)$, marginální pro jednotlivé X_i, X_1, X_2 nezávislé, jestliže $F_{X_1X_2}(x_1, x_2) = F_{X_1}(x_1)F_{X_2}(x_2)$ (a tedy i $f_{X_1X_2}(x_1, x_2) = f_{X_1}(x_1)f_{X_2}(x_2)$)

Modely dat

Pravděpodobnostní model

- střední hodnota (expected value) náhodné proměnné X: $E[X] = \sum_i x_i P(X = x_i)$ pro diskrétní X, $E[X] = \int_{-\infty}^{\infty} x f_X(x) dx$ pro spojitou X, statistický průměr (mean, statistical average) $\mu_X = E[X]$, rozptyl (variance) $\sigma_X^2 = E[(X - \mu_X)^2] = E[X^2] - \mu_X^2$, standardní odchylka (standard deviation) $\sigma_X = \sqrt{\sigma_X^2}$, X_1, X_2 nekorelované, jestliže $E[(X_1 - \mu_1)(X_2 - \mu_2)] = 0$
- náhodný/stochastický proces: měřitelné $X : \Omega \mapsto \mathcal{F}, \mathcal{F} : \mathbb{R} \mapsto \mathbb{R}$, realizace $X(\omega) = x(t), -\infty < t < \infty$ funkce času, ensemble $X(t) = \{x_\omega(t)\}$, střední hodnota ensemble, vzorek (sample) $X(t_0)$ ensemble = náhodná proměnná
- problém nulové pravděpodobnosti/frekvence (zero probability/frequency problem): kompresní metody předpokládají u modelu všechny uvažované pravděpodobnosti/frekvence nenulové $\rightarrow$ místo nulových nastavení velice malých

Modely dat

Markovův model (Andrei A. Markov)

- výskyt hodnoty x_j na výstupu zdroje dat je závislý na výskytu (některých, ne nutně bezprostředně) předchozích hodnot $x_i, i < j$
- vychází z pravděpodobnostního modelu
- v bezeztrátové kompresi Markovův řetěz s diskrétním časem: posloupnost hodnot x_j (náhodné proměnné X_j) následuje Markovův model/proces k-tého řádu, jestliže $P(x_j|x_{i_1}, x_{i_2}, \ldots, x_{i_k}) = P(x_j|x_{j-1}, x_{j-2}, \ldots), i_1, i_2, \ldots, i_k < j$ (znalost některých předchozích k hodnot je stejná jako znalost všech předchozích hodnot), posloupnosti s_j hodnot $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ = stavy modelu/procesu/řetězu, $P(x_j|s_j)$ = pravděpodobnosti přechodu mezi stavy
- nejběžnější model 1. řádu: $P(x_j|x_i = P(x_j|x_{j-1}, x_{j-2}, \ldots), i < j$
- s_j, $P(s_j)$, $P(x_j|s_j)$ ... stavový diagram
- různé modely podle formy závislosti, se zvyšujícím se k vyšší míra komprese než s nezávislými výskyty hodnot
- v kompresi textu Markovův model k-tého řádu = model konečného kontextu (finite context model) – kontext = stav modelu

Markovův model

Příklad

$x_1 x_2 \dots x_{10} = aababbabaa$

stavy modelu 1. řádu = posloupnosti (bezprostředně) předchozích symbolů délky 1 pro všechny symboly: a, b
$P(a) = \frac{6}{10}$, $P(b) = \frac{4}{10}$,
$P(a|a) = \frac{2}{5}$, $P(b|a) = \frac{3}{5}$, $P(a|b) = \frac{3}{4}$, $P(b|b) = \frac{1}{4}$

stavy modelu 2. řádu = posloupnosti (bezprostředně) předchozích symbolů délky 2 pro všechny symboly: aa, ab, ba, bb
$P(aa) = \frac{2}{9}$, $P(ab) = \frac{3}{9}$, $P(ba) = \frac{3}{9}$, $P(bb) = \frac{1}{9}$,
$P(a|aa) \to 0$, $P(b|aa) \to 1$, $P(a|ab) = \frac{2}{3}$, $P(b|ab) = \frac{1}{3}$, $P(a|ba) = \frac{1}{3}$, $P(b|ba) = \frac{2}{3}$,
$P(a|bb) \to 1$, $P(b|bb) \to 0$

Modely dat

Typy

- statický – neměnný pro různá originální data a během kódování, známý algoritmu dekomprese
- semi-adaptivní – vytvořený pro originální data (1. průchod daty při kompresi), během kódování neměnný (2. průchod) a předaný algoritmu dekomprese (např. s komprimovanými daty)
- adaptivní – dynamicky vytvářený/modifikovaný podle doposud zakódovaných originálních a dekomprimovaných dat

Teorie informace

Klasická Shannonova

- rámec pro bezeztrátové kompresní metody, vychází z pravděpodobnostního modelu dat
- Claude E. Shannon: A Mathematical Theory of Communication. *Bell System Technical Journal* **27**, pp. 379–423, 623–656, 1948.
- „míra průměrné informace (asociované s) experimentu(-em)“ – požadavky, pro nezávislé jevy $A_i, i = 1, \ldots, m, \bigcup A_i = \Omega$:
  1. spojitá funkce $H(p_i), p_i = P(A_i)$
  2. monotónně rostoucí vzhledem k počtu m stejně pravděpodobných jevů A_i ($p_i = \frac{1}{m}$)
  3. stejná při rozdělení experimentu na k podexperimentů (s disjunktními podmnožinami množiny jevů A_i), výsledek experimentu = podmnožina s jevem, výsledek podexperimentu = jev v podmnožině:
     $H(p_i) = H(q_1, q_2, \ldots, q_k) + q_1 H(\frac{p_1}{q_1}, \frac{p_2}{q_1}, \ldots, \frac{p_{j_1}}{q_1}) + q_2 H(\frac{p_{j_1+1}}{q_2}, \frac{p_{j_1+2}}{q_2}, \ldots, \frac{p_{j_2}}{q_2}) + \ldots +$
     $q_k H(\frac{p_{j_{k-1}+1}}{q_k}, \frac{p_{j_{k-1}+2}}{q_k}, \ldots, \frac{p_{j_k}}{q_k}), q_1 = \sum_{i=1}^{j_1} p_i, q_2 = \sum_{i=j_1+1}^{j_2} p_i, \ldots, q_k = \sum_{i=j_{k-1}+1}^{j_k=m} p_i$
- jediné možné řešení požadavků (Shannon): $H(p_i) = -K \sum_i p_i \log p_i$, K kladná konstanta

Teorie informace

Klasická Shannonova

- informace (self-information) (asociovaná s výskytem) jevu A: $i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)$ – $\log(1) = 0$ a roste s klesající $P(A) \neq 0$, pro nezávislé A, B $i(AB) = i(A) + i(B)$
- jednotka i: bit (shannon) pro $b = 2$, nat pro $b = e$, hartley pro $b = 10$
- entropie (asociovaná s) experimentu(-em): průměr $H(A_i) = \sum_i P(A_i) i(A_i) = -\sum_i P(A_i) \log P(A_i)$ informací nezávislých jevů $A_i, \bigcup A_i = \Omega$ (jako náhodných proměnných), $0 \log 0 := 0$
- Shannon: experiment = zdroj Z posloupnosti $X_1, X_2, \ldots, X_n$ symbolů z množiny $\{a_1, a_2, \ldots, a_m\}$ jako náhodných proměnných $X_j(a_i) = i$, pak entropie zdroje = průměrný počet binárních symbolů (bitů) potřebných pro zakódování každého symbolu posloupnosti = $H(Z) = \lim_{n\to\infty} \frac{1}{n} G_n, G_n = -\sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \ldots \sum_{i_n=1}^{m} P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n) \log P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n)$ – limit pro bezeztrátovou kompresi

Klasická Shannonova

- jestliže je výskyt každého symbolu X_j (jako náhodné proměnné) nezávislý a stejně pravděpodobnostně rozložený, pak $X_j = X$, $G_n = -n \sum_{i=1}^{m} P(X = i) \log P(X = i)$ a $^1H(Z) = -\sum_{i=1}^{m} P(a_i) \log P(a_i) =$ entropie 1. řádu
- podmíněná entropie (pro náhodné proměnné) X_1 v závislosti na X_2: průměr $H(Z) = H(X_1|X_2) = \sum_{i_2=1}^{m} P(a_{i_2}) H(X_1|X_2 = i_2) =$ $-\sum_{i_2=1}^{m} P(a_{i_2}) \sum_{i_1=1}^{m} P(a_{i_1}|a_{i_2}) \log P(a_{i_1}|a_{i_2})$ podmíněných entropií X_1 v závislosti na $X_2 = i_2$
- entropie Markovova modelu 1. řádu se stavy $S = \{s_j\}$: $H(X|S)$
- entropie (obecně) nezjistitelná $\Rightarrow$ odhad závislý na modelu struktury dat!

Klasická Shannonova

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

- $\{a_1, a_2, \ldots, a_7\} = \{2, 4, 6, 7, 10, 11, 14\}$
  $P(a_i) = p_i \approx f(a_i) = f_i$: $f_1 = f_5 = f_6 = \frac{2}{12}, f_2 = f_3 = f_7 = \frac{1}{12}, f_4 = \frac{3}{12}$
  $^1H(a_i) = -\sum_{i=1}^{7} p_i \log_2 p_i \doteq 2.689$ bitů/číslo

Klasická Shannonova

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

- $\{a_1, a_2, \ldots, a_7\} = \{2, 4, 6, 7, 10, 11, 14\}$
  $P(a_i) = p_i \approx f(a_i) = f_i$: $f_1 = f_5 = f_6 = \frac{2}{12}, f_2 = f_3 = f_7 = \frac{1}{12}, f_4 = \frac{3}{12}$
  ${}^1H(a_i) = -\sum_{i=1}^{7} p_i \log_2 p_i \doteq 2.689$ bitů/číslo
- Sousední čísla nejsou nezávislá $\rightarrow$ odstranění závislosti (korelace):
  $d_2, d_3, \ldots, d_{12} = 0, 2, 2, 1, 0, 0, 3, 0, 1, 0, 3$, $\{a_1, a_2, a_3, a_4\} = \{0, 1, 2, 3\}$
  $f_1 = \frac{5}{11}, f_2 = f_3 = f_4 = \frac{2}{11}$
  ${}^1H(a_i) = -\sum_{i=1}^{4} p_i \log_2 p_i \doteq 1.859$ bitů/číslo

Klasická Shannonova

Příklad

$x_1, x_2, \ldots, x_{12} = 2, 2, 4, 6, 7, 7, 7, 10, 10, 11, 11, 14$

- $\{a_1, a_2, \ldots, a_7\} = \{2, 4, 6, 7, 10, 11, 14\}$
  $P(a_i) = p_i \approx f(a_i) = f_i$: $f_1 = f_5 = f_6 = \frac{2}{12}, f_2 = f_3 = f_7 = \frac{1}{12}, f_4 = \frac{3}{12}$
  ${}^1H(a_i) = -\sum_{i=1}^{7} p_i \log_2 p_i \doteq 2.689$ bitů/číslo
- Sousední čísla nejsou nezávislá $\rightarrow$ odstranění závislosti (korelace):
  $d_2, d_3, \ldots, d_{12} = 0, 2, 2, 1, 0, 0, 3, 0, 1, 0, 3$, $\{a_1, a_2, a_3, a_4\} = \{0, 1, 2, 3\}$
  $f_1 = \frac{5}{11}, f_2 = f_3 = f_4 = \frac{2}{11}$
  ${}^1H(a_i) = -\sum_{i=1}^{4} p_i \log_2 p_i \doteq 1.859$ bitů/číslo
- Všechna čísla jsou mezi sebou závislá $\rightarrow$ odstranění závislosti (korelace):
  $d_1, d_2, \ldots, d_{12} = 0, -1, 0, 1, 1, 0, -1, 1, 0, 0, -1, 1$, $\{a_1, a_2, a_3\} = \{0, -1, 1\}$
  $f_1 = \frac{5}{12}, f_2 = \frac{3}{12}, f_3 = \frac{4}{12}$
  ${}^1H(a_i) = -\sum_{i=1}^{3} p_i \log_2 p_i \doteq 1.555$ bitů/číslo

Klasická Shannonova

Příklad

$x_1 x_2 \dots x_{10} = aababbabaa$

výskyt a a b nezávislý: $P(a) = \frac{6}{10}$, $P(b) = \frac{4}{10}$
$H = -P(a)\log_2 P(a) - P(b)\log_2 P(b) \doteq 0.971\,\text{b/symbol}$

Klasická Shannonova

Příklad

$x_1 x_2 \dots x_{10} = aababbabaa$

výskyt a a b nezávislý: $P(a) = \frac{6}{10}$, $P(b) = \frac{4}{10}$
$H = -P(a)\log_2 P(a) - P(b)\log_2 P(b) \doteq 0.971$ b/symbol

Markovův model 1. řádu: $P(a) = \frac{5}{9}$, $P(b) = \frac{4}{9}$,
$P(a|a) = \frac{2}{5}$, $P(b|a) = \frac{3}{5}$, $P(a|b) = \frac{3}{4}$, $P(b|b) = \frac{1}{4}$
$H = P(a)H(X|a) + P(b)H(X|b) = P(a)(-P(a|a)\log_2 P(a|a) - P(b|a)\log_2 P(b|a)) +$
$P(b)(-P(a|b)\log_2 P(a|b) - P(b|b)\log_2 P(b|b)) \doteq 0.9$ b/symbol

Klasická Shannonova

Příklad

$x_1x_2 \dots x_{10} = aababbabaa$

výskyt a a b nezávislý: $P(a) = \frac{6}{10}$, $P(b) = \frac{4}{10}$
$H = -P(a)\log_2 P(a) - P(b)\log_2 P(b) \doteq 0.971$ b/symbol

Markovův model 1. řádu: $P(a) = \frac{5}{9}$, $P(b) = \frac{4}{9}$,
$P(a|a) = \frac{2}{5}$, $P(b|a) = \frac{3}{5}$, $P(a|b) = \frac{3}{4}$, $P(b|b) = \frac{1}{4}$
$H = P(a)H(X|a) + P(b)H(X|b) = P(a)(-P(a|a)\log_2 P(a|a) - P(b|a)\log_2 P(b|a)) +$
$P(b)(-P(a|b)\log_2 P(a|b) - P(b|b)\log_2 P(b|b)) \doteq 0.9$ b/symbol

Markovův model 2. řádu: $P(aa) = \frac{1}{8}$, $P(ab) = \frac{3}{8}$, $P(ba) = \frac{3}{8}$, $P(bb) = \frac{1}{8}$,
$P(a|aa) \to 0$, $P(b|aa) \to 1$, $P(a|ab) = \frac{2}{3}$, $P(b|ab) = \frac{1}{3}$, $P(a|ba) = \frac{1}{3}$, $P(b|ba) = \frac{2}{3}$,
$P(a|bb) \to 1$, $P(b|bb) \to 0$
$H = \sum_{x_1x_2=aa}^{bb} P(x_1x_2)H(X|x_1x_2) =$
$\sum_{x_1x_2=aa}^{bb} P(x_1x_2) - \sum_{y=a,b} P(y|x_1x_2)\log_2 P(y|x_1x_2) \doteq 0.689$ b/symbol

Teorie informace

Klasická Shannonova

Uvažováním závislosti mezi symboly dat (posloupnosti) v modelu struktury dat nesnižujeme „entropii dat“. Entropie je vlastnost (hypotetického) zdroje dat a je stejná pro všechna data ze zdroje. Snižujeme odhad této entropie uvažováním delších n-tic symbolů dat a závilostí mezi nimi v modelu (až do $n \to \infty$)!

Algoritmická

- Kolmogorov/descriptive complexity / algoritmická entropie dat (Andrey N. Kolmogorov): délka nejmenšího/nejkratšího počítačového programu (včetně vstupu, v jakémkoliv programovacím jazyce), jehož jsou data výstupem – způsob modelování struktury dat
- není znám žádný systematický způsob výpočtu nebo libovolně blízkého odhadu
- Minimum Description Length (MDL) princip (J. Risannen): $MDL(x) = min_j(D_{M_j} + R_{M_j}(x))$, D_{M_j} délka popisu možného modelu M_j struktury dat x, $R_{M_j}(x)$ délka reprezentace x podle modelu M_j
- např. M_j polynomy j-tého řádu: pro vyšší j kratší $R_{M_j}(x)$ (přesnější model), ale delší D_{M_j} (složitější model), a naopak $\Rightarrow$ kompromis

Kódování

- abeceda $A = \{a_1, a_2, \ldots, a_n\}$, a_i = symboly

= (kód) ze zdrojové abecedy A do kódové abecedy B: injektivní $C : A \mapsto B^+$, B^+ = množina konečných neprázdných posloupností (= slov) symbolů z B – často $B = \{\mathbf{0}, \mathbf{I}\} \rightarrow$ binární kódování (kód)

- $C(a_i) \in B^+$ ... kódové slovo (kód) pro symbol a_i, $C(A) = \{C(a_i), a_i \in A\} \subseteq B^+$ ... kód (pro zdrojovou abecedu A), $l(a_i)$ ... délka $C(a_i)$, pro $B = \{\mathbf{0}, \mathbf{I}\}$ v bitech
- dekódování: $D : C(A) \mapsto A$
- např. $\{\mathbf{0}, \mathbf{I}, \mathbf{00}, \mathbf{II}\}$, ne $\{\mathbf{0}, \mathbf{0}, \mathbf{I}, \mathbf{II}\}$
- blokový kód (kód pevné délky, fixed-length code) = všechna kódová slova (pro všechny symboly) mají stejnou délku, např. ASCII

Kódování

Jednoznačně dekódovatelný kód

= každá (neprázdná) posloupnost symbolů z kódové abecedy je zřetězením nejvýše jedné posloupnosti kódových slov

= $C^+ : A^+ \mapsto B^+, C^+(a_{i_1}a_{i_2}\dots a_{i_j}) = C(a_{i_1})C(a_{i_2})\dots C(a_{i_j})$ injektivní

- dekódování: $D^+ : C^+(A^+) \mapsto A^+$
- např. každý blokový, $\{\mathbf{0}, \mathbf{0I}, \mathbf{0II}, \mathbf{III}\}$, ne $\{\mathbf{0}, \mathbf{0I}, \mathbf{I0}, \mathbf{II}\}$
- test: $S \leftarrow C(A)$ a opakuj $S \leftarrow S \cup \{s \in B^+; ps \in S \wedge p \in S\}$ dokud některé $s \in C(A)$ nebo S zůstane stejná, při $s \in C(A)$ kód $C(A)$ není jednoznačně dekódovatelný

Prefixový (prefix, instantaneous) kód

= žádné kódové slovo není prefixem jiného kódového slova

- např. každý blokový, $\{\mathbf{0}, \mathbf{I0}, \mathbf{II0}, \mathbf{III}\}$
- jednoznačně dekódovatelný

Kódování

Věta (Kraftova)

Prefixový kód s k kódovými slovy délek $l_1, l_2, \ldots, l_k$ nad kódovou abecedou velikosti m existuje právě když

$$\sum_{i=1}^{k} m^{-l_i} \leq 1 \quad \ldots \quad \textit{Kraftova nerovnost.}$$

Kódování

Věta (Kraftova)

Prefixový kód s k kódovými slovy délek $l_1, l_2, \ldots, l_k$ nad kódovou abecedou velikosti m existuje právě když

$$\sum_{i=1}^{k} m^{-l_i} \leq 1 \quad \ldots \quad \textit{Kraftova nerovnost.}$$

□

Věta (McMillanova)

Jednoznačně dekódovatelný kód s k kódovými slovy délek $l_1, l_2, \ldots, l_k$ nad kódovou abecedou velikosti m existuje právě když

$$\sum_{i=1}^{k} m^{-l_i} \leq 1 \quad (\ldots \quad \textit{Kraft-McMillanova nerovnost}).$$

□

Kódování

Optimální kód

- pro pravděpodobnostní model dat (výskytu symbolů), prefixový kód
- průměrná délka kódu (na symbol, code rate): průměr $\bar{l}(C(A)) = \sum_{i=1}^{n} P(a_i)l(a_i)$ délek $l(a_i)$ pro všechny $a_i \in A$, $P(a_i) \neq 0$ = pravděpodobnost výskytu symbolu a_i

$=$ s minimální $\bar{l}(C(A))$ (v rámci třídy kódů, např. prefixové)

Kódování

Optimální kód

- pro pravděpodobnostní model dat (výskytu symbolů), prefixový kód
- průměrná délka kódu (na symbol, code rate): průměr $\bar{l}(C(A)) = \sum_{i=1}^{n} P(a_i) l(a_i)$ délek $l(a_i)$ pro všechny $a_i \in A$, $P(a_i) \neq 0$ = pravděpodobnost výskytu symbolu a_i

$=$ s minimální $\bar{l}(C(A))$ (v rámci třídy kódů, např. prefixové)

Věta (Shannon noisless coding theorem)

Pro optimální jednoznačně dekódovatelný kód ze zdrojové abecedy A do kódové abecedy B platí

$$\frac{H(A)}{\log_b m} \leq \bar{l}(C(A)) < \frac{H(A)}{\log_b m} + 1$$

kde $H(A)$ je entropie zdroje symbolů z A, b je stejné jako v H a m je velikost B.
$\bar{l}(C(A)) = \frac{H(A)}{\log_b m}$ právě když $P(a_i) = m^{-l(a_i)}$ pro všechny $a_i \in A$. □

- redundance kódu: $\bar{l}(C(A)) - \frac{H(A)}{\log_b m}$, také v % z $\frac{H(A)}{\log_b m}$, $\bar{l}(C(A)) = \frac{H(A)}{\log_b m}$ ... absolutně optimální kód

Optimální kód

- změnou zdrojové abecedy na k-tice (nezávislých) symbolů z původní abecedy A (rozšíření zdrojové abecedy, source extension) se lze $\bar{l}(C(A)) = \frac{H(A)}{\log_b m}$ libovolně přiblížit (až do $k \to \infty$):

$$\frac{H(A^k)}{\log_b m} \leq \bar{l}(C(A^k)) < \frac{H(A^k)}{\log_b m} + 1$$
$$\frac{kH(A)}{\log_b m} \leq k\bar{l}(C(A)) < \frac{kH(A)}{\log_b m} + 1$$
$$\frac{H(A)}{\log_b m} \leq \bar{l}(C(A)) < \frac{H(A)}{\log_b m} + \frac{1}{k}$$

Optimální kód

- změnou zdrojové abecedy na k-tice (nezávislých) symbolů z původní abecedy A (rozšíření zdrojové abecedy, source extension) se lze $\bar{l}(C(A)) = \frac{H(A)}{\log_b m}$ libovolně přiblížit (až do $k \to \infty$):

$$\frac{H(A^k)}{\log_b m} \leq \bar{l}(C(A^k)) < \frac{H(A^k)}{\log_b m} + 1$$
$$\frac{kH(A)}{\log_b m} \leq k\bar{l}(C(A)) < \frac{kH(A)}{\log_b m} + 1$$
$$\frac{H(A)}{\log_b m} \leq \bar{l}(C(A)) < \frac{H(A)}{\log_b m} + \frac{1}{k}$$

- abeceda A^k ale může mít velikost až n^k (n je velikost A)!

Optimální kód

Příklad

$A = \{a_1, a_2, a_3\}$
$P(a_i) = 0.8, P(a_2) = 0.02, P(A_3) = 0.18$
$H(A) = -\sum_{i=1}^{3} P(a_i) \log_2 P(a_i) \doteq 0.816$ bitů/symbol
$C(A) = \{\langle a_1, \mathbf{0}\rangle, \langle a_2, \mathbf{11}\rangle, \langle a_3, \mathbf{10}\rangle\}$
$\bar{l}(C(A)) = \sum_{i=1}^{3} P(a_i) l(a_i) = 1.2$ b/symbol
$\bar{l}(C(A)) - \frac{H(A)}{\log_2 2} \doteq 0.384$ b/symbol $\doteq 47\,\%$

Optimální kód

Příklad

$A = \{a_1, a_2, a_3\}$
$P(a_i) = 0.8, P(a_2) = 0.02, P(A_3) = 0.18$
$H(A) = -\sum_{i=1}^{3} P(a_i) \log_2 P(a_i) \doteq 0.816$ bitů/symbol
$C(A) = \{\langle a_1, \mathbf{0}\rangle, \langle a_2, \mathbf{II}\rangle, \langle a_3, \mathbf{I0}\rangle\}$
$\bar{l}(C(A)) = \sum_{i=1}^{3} P(a_i) l(a_i) = 1.2$ b/symbol
$\bar{l}(C(A)) - \frac{H(A)}{\log_2 2} \doteq 0.384$ b/symbol $\doteq 47\,\%$

$A^2 = \{a_1a_1, a_1a_2, a_1a_3, a_2a_1, a_2a_2, a_2a_3, a_3a_1, a_3a_2, a_3a_3\}$
$P(a_1a_1) = 0.64, P(a_1a_2) = P(a_2a_1) = 0.016, P(a_1a_3) = P(a_3a_1) = 0.144, P(a_2a_2) = 0.0004, P(a_2a_3) = P(a_3a_2) = 0.0036, P(a_3a_3) = 0.0324$
$C(A^2) = \{\langle a_1a_1, \mathbf{0}\rangle, \langle a_1a_2, \mathbf{I0I0I}\rangle, \langle a_1a_3, \mathbf{II}\rangle, \langle a_2a_1, \mathbf{I0I000}\rangle, \langle a_2a_2, \mathbf{I0I00I0I}\rangle, \langle a_2a_3, \mathbf{I0I00II}\rangle, \langle a_3a_1, \mathbf{I00}\rangle, \langle a_3a_2, \mathbf{I0I00I00}\rangle, \langle a_3a_3, \mathbf{I0II}\rangle\}$
$\bar{l}(C(A)) = \frac{\bar{l}(C(A^2))}{2} = \frac{\sum_{i=1,j=1}^{3,3} P(a_ia_j) l(a_ia_j)}{2} \doteq \frac{1.723}{2} \doteq 0.862$ b/symbol
$\bar{l}(C(A)) - \frac{H(A)}{\log_2 2} \doteq 0.046$ b/symbol $\doteq 5.6\,\%$

Optimální kód

Věta

Pro optimální prefixový kód ze zdrojové abecedy A do kódové abecedy B platí

1. *Symboly z A s větší pravděpodobností výskytu mají kratší kódová slova.*
2. *$m' \in \{2, 3, \ldots, m\}, m' \equiv n \pmod{(m-1)}$ symbolů z A s nejmenší pravděpodobností výskytu, kde $n \geq 2$ je velikost A a $m \geq 2$ je velikost B, má stejně (maximálně) dlouhá kódová slova a ta se liší pouze v jednom symbolu.* □

Proč m' a ne m? Odpověď u Huffmanova kódování (viz dále).

Základní techniky a kódování čísel

Run-length encoding (RLE)

= kódování posloupností stejných zdrojových symbolů (runs) kódy příznaku kódování opakování, délky posloupnosti a jednoho symbolu místo samotných symbolů

- podle délky kódů příznaku a délky až pro posloupnosti delší než určitý počet k symbolů, např. 3
- kód příznaku může být zaměnitelný s kódem symbolu $\rightarrow$ kódování s kódem délky zmenšené o k za kódy určitého počtu k symbolů
- aplikace: text, obraz (BMP)

Diferenční kódování

= kódování (malého) rozdílu symbolu/čísla od předchozího (nebo predikce z několika předchozích) kódy příznaku kódování rozdílu a rozdílu místo samotného symbolu/čísla, s výjimkou prvního

Run-length encoding (RLE)

Input: číslo k
$r \leftarrow 0$;
while načti ze vstupu symbol a **do**
    if $r = 0$ **then**
        $x \leftarrow a$;
        $r \leftarrow 1$;
    else
        if $a = x$ **then**
            $r \leftarrow r + 1$;
        else
            if $r \leq k$ **then**
                zapiš na výstup r kódů symbolu x;
            else
                zapiš na výstup kódy příznaku, čísla r a symbolu x / k kódů symbolu x a kód čísla $r - k$;
            $x \leftarrow a$;
            $r \leftarrow 1$;

PRIKLAD: $k = 3$, vstup $bbbbaaaarrrbbaaaaara$, kod priznaku x, kod opakovani cislo, kod symbolu symbol, obe varianty

Move-to-front (MTF) kódování

= kódování často se opakujících symbolů malými čísly (speciálně posloupností stejných symbolů posloupností čísel 0)

- lokálně adaptivní = adaptace podle lokálních četností výskytu symbolů

Uses: zdrojová abeceda $A = \{a_1, \ldots, a_n\}$, (volitelně) pravděpodobnosti $\{p_1, \ldots, p_n\}$ výskytu a_i
(volitelně) setřiď a_i a p_i tak, že $p_i \geq p_j$ pro $i < j$;
while načti ze vstupu symbol $a \in A$ **do**
    zapiš na výstup číslo $i - 1$, kde $a_i = a$;
    if $i > 1$ **then**
        $x \leftarrow a_i$;
        $a_j \leftarrow a_{j-1}$ pro $j = 2, 3, \ldots, i$;
        $a_1 \leftarrow x$;

PRIKLAD: $A = \{a, b, r\}$, $p(a) = \frac{10}{20}, p(b) = \frac{6}{20}, p(r) = \frac{4}{20}$, vstup $bbbbaaaarrrbbaaaaara$, se setrizenim i bez

Kódování čísel

- přirozených čísel – celá lze bijektivně zobrazit na přirozená (např. $-2i$ pro $i < 0$ a $2i + 1$ pro $i \geq 0$)
- předpoklad nižší pravděpodobnosti výskytu u větších čísel
- binární kódy s proměnnou délkou (variable-length codes, fixed-to-variable codes) = proměnná délka kódových slov pro zdrojová slova pevné délky (symboly nebo jejich k-tice), nízká průměrná délka kódu vs. náročnější manipulace s kódem (v porovnání s blokovým kódem, s využitím bufferu)

Unární kód

- kódování přirozených čísel
- = pro $i \geq 0$: zřetězení i **I** a **0** (nebo opačně), např. **IIIII0** pro 5
- prefixový, délka $i + 1$, optimální při $P(i) = \frac{1}{2^i}$

Další

- start-step-stop (obecné unární) kódy, start/stop kód, Levensteinův kód, Stoutovy kódy, Yamamotovy kódy, taboo kódy, Goldbachovy kódy, aditivní kódy aj.

Eliasovy kódy

- kódování přirozených čísel, P. Elias
- **Alpha** $= \alpha(i)$ pro $i \geq 0$: unární kód i, s $\mathbf{I}$ na konci
- **Beta** $= \beta(i)$ pro $i \geq 0$: reprezentace i ve dvojkové soustavě (= binární reprezentace) – neprefixový
- další – pro $i \geq 1$: zřetězení kódu $l(\beta(i)) = \lfloor \log_2 i \rfloor + 1$ a $\beta(i)$, prefixové
- pro každé $i \geq 1$: $i = 2^{l(\beta(i))-1} + k$, $0 \leq k < 2^{l(\beta(i))-1}$

Gamma

= $\gamma(i)$ pro $i \geq 1$: zřetězení $l(\beta(i)) - 1$ $\mathbf{0}$ a $\beta(i)$ nebo $\alpha(l(\beta(i)) - 1)$ a $\beta(k)$, např. $\mathbf{00I0I}$ pro 5

- délka $2\lfloor \log_2 i \rfloor + 1$, optimální při $P(i) = \frac{1}{2i^2}$

Delta

= $\delta(i)$ pro $i \geq 1$: zřetězení $l(\beta(l(\beta(i)))) - 1$ $\mathbf{0}$, $\beta(l(\beta(i)))$ a $\beta(i)$ bez první $\mathbf{I}$ nebo $\gamma(l(\beta(i)))$ a $\beta(k)$, např. $\mathbf{0II0I}$ pro 5

- délka $2\lfloor \log_2 \log_2 2i \rfloor + \lfloor \log_2 i \rfloor + 1$, optimální při $P(i) = \frac{1}{2i(\log_2 2i)^2}$

Eliasovy kódy

Omega (rekurzivní)

= $\omega(i)$ pro $i = 1$: $\mathbf{0}$, a pro $i \geq 2$: zřetězení odzadu $\mathbf{0}$ a počínaje $k := i$ pokud $k \geq 2$ rekurzivně $\beta(k)$, $k := l(\beta(k)) - 1$, např. $\mathbf{I0I0I0}$ pro 5

- dekódování: $i := 1$ a opakovaně jestliže je další bit $\mathbf{I}$ tak s dalšími i bity tvoří kód $\beta(i)$
- délka $\sum_{j=1}^{k}(\lfloor \log_2 i \rfloor^j + 1) + 1$, $\lfloor \log_2 i \rfloor^k = 1$

Fibonacciho kódy

- kódování přirozených čísel, L. Pisano (Fibonacci)
- Fibonacciho reprezentace $a_1 a_2 \ldots$ přirozeného čísla $i \geq 1$: $i = \sum_{j=1} a_j F_j$, $a_j \in \{0, 1\}$, F_j j-té Fibonacciho číslo ($F_1 = 1, F_2 = 2, F_j = F_{j-1} + F_{j-2}$) – neobsahuje sousední 1

$=$ pro $i \geq 1$: zřetězení Fibonacciho reprezentace i (jako bitů) a $\mathbf{I}$, např. $\mathbf{000II}$ pro 5 – končí $\mathbf{II}$

- délka $\leq \lfloor \log_\phi \sqrt{5} n \rfloor + 1$, $\phi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ tzv. zlatý řez
- prefixové, robustnější než jiné kódy čísel
- další (zobecněné) založené na k-krokových (zobecněných) Fibonacciho číslech

Golombovy kódy

- kódování přirozených čísel, S. W. Golomb
- parametr přirozené číslo $j > 0$

= pro $i \geq 0$ zřetězení dvou kódů:

1. unární kód $q = \lfloor \frac{i}{j} \rfloor$ (= celé části $\frac{i}{j}$)
2. $\lfloor \log_2 j \rfloor$-bitová binární reprezentace $r = i - qj$ (= zbytek po celočíselném dělení $\frac{i}{j}$) pro $r = 0, 1, \ldots, 2^{\lceil \log_2 j \rceil} - j - 1$ a $\lceil \log_2 j \rceil$-bitová binární reprezentace $r + 2^{\lceil \log_2 j \rceil} - j$ pro $r = 2^{\lceil \log_2 j \rceil} - j, \ldots, j - 1$

Příklad

$j = 5$
$\lfloor \log_2 5 \rfloor = 2$-bitové binární reprezentace $r = 0, 1, 2$ a $\lceil \log_2 5 \rceil = 3$-bitové binární reprezentace $r + 3$ pro $r = 3, 4$
$0 \mapsto \mathbf{000}$, $1 \mapsto \mathbf{00I}$, $2 \mapsto \mathbf{0I0}$, $3 \mapsto \mathbf{0II0}$, $4 \mapsto \mathbf{0III}$, $5 \mapsto \mathbf{I000}$, $6 \mapsto \mathbf{I00I}$, ...

Golombovy kódy

- délka pro malé j z malé rychle narůstá, pro velké j z delší narůstá pomalu
- prefixové, pro $j = \lceil -\frac{1}{\log_2 p} \rceil$ (přesněji $j = \lceil -\frac{\log_2(1+p)}{\log_2 p} \rceil$) optimální při $P(i) = p^{i-1}(1-p)$ – geometrické rozdělení pravděpodobnosti, např. posloupnost (run z RLE) $i-1$ výskytů symbolu s vysokou pravděpodobností výskytu p ukončená jedením výskytem jiného symbolu s nízkou pravděpodobností $1-p$ (např. prohra a výhra) $\rightarrow$ (adaptivní) Golomb RLE
- použití např. v bezeztrátové kompresi obrazu (JPEG-LS)

Riceovy kódy

$\sim$ Golomb-Riceovy kódy, R. F. Rice (Rice machine)

$=$ Golombovy kódy pro $j = 2^k$ pro nějaké (celé nezáporné) k

- jednodušší kódování (a dekódování) pro $i \geq 0$: zřetězení unárního kódu pro zbývajících $q = \lfloor \frac{i}{j} \rfloor$ bitů a k nejméně významných bitů binární reprezentace i
- délka $\lfloor \frac{i}{j} \rfloor + k + 1$, optimální při $P(i) = \frac{1}{2^{\frac{i}{j}+k+1}}$
- použití např. v bezeztrátové kompresi audia (MPEG-4 ALS, FLAC)

Statistické metody

Tunstallův kód

= zdrojová slova proměnné délky kódována na kódová slova pevné délky $k \geq \lceil \log_m n \rceil$ kódových symbolů = blokový kód, $\sim$ variable-to-fixed code (n je velikost zdrojové abecedy, m je velikost kódové abecedy)

- chyby v kódových slovech se při dekódování nešíří – robustnost
- požadavky:
  1. každou (neprázdnou) posloupnost zdrojových symbolů musí být možné vyjádřit jako (případně prefix) zřetězení právě jedné posloupnosti zdrojových slov kódovaných na kódové slovo – jednoznačná kódovatelnost
  2. průměrná délka zdrojových slov kódovaných na kódové slovo je maximální = optimální kód $\rightarrow$ delší zdrojová slova mají větší pravděpodobnost výskytu
  3. je použito maximum kódových slov, ideálně všech m^k – optimalita
- prefixový – pro zdrojová slova kódovaná na kódové slovo, jednoznačná kódovatelnost
- průměrná délka kódu: $\frac{k}{\sum_{i=1}^{t} P(w_i) l(w_i)}$, t počet zdrojových slov w_i délky $l(w_i)$ s pravděpodobností výskytu $P(w_i)$
- pouze statický a semi-adaptivní model

Tunstallův kód

- B. P. Tunstall

Input : číslo k
Uses : zdrojová abeceda A, $n = |A|$, pravděpodobnosti $P(A^+)$ výskytu slova z A, velikost m kódové abecedy
Output: $C(U)$
$T \leftarrow A$;
$i \leftarrow 0$;
while $n + (i+1)(n-1) \leq m^k$ **do**
    $x \leftarrow w \in T, P(w) \geq P(w'), w' \in T$;
    $T \leftarrow (T \setminus \{x\}) \cup \{xy \,|\, y \in A\}$;
    $i \leftarrow i + 1$;
$C(T) \leftarrow \{$kódová slova délky $k\}$ libovolně;

PRIKLAD: $k = 4$, $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$, $p(r) = \frac{5}{20}$, $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$, $m = 2$, vstup $barbaraabarboraubaru$, redundance

Shannon-Fanovo kódování

- C. E. Shannon, R. M. Fano
- první pokus o optimální binární prefixový kód – využití distribuční funkce/kumulované pravděpodobnosti (cumulative distribution function) zdroje

Input : čísla a, b
Uses : zdrojová abeceda $A = \{a_1, \ldots, a_n\}, n \geq 2$, pravděpodobnosti $\{p_1, \ldots, p_n\}$, $p_i \geq p_j$ pro $i < j$, výskytu a_i
Output : kód $C(A)$
if $a + 1 = b$ **then**
    $C(a_a) \leftarrow \mathbf{0}$;
    $C(a_b) \leftarrow \mathbf{I}$;
najdi j takové, že $|\sum_{i=a}^{j} p_i - \sum_{i=j+1}^{b} p_i|$ je minimální;
$C(A) \leftarrow$ zavolej se rekurzivně s a, j a s $j + 1, b$;
$C(a_i) \leftarrow \mathbf{0}C(a_i)$ pro $i = a, \ldots, j$;
$C(a_i) \leftarrow \mathbf{I}C(a_i)$ pro $i = j + 1, \ldots, b$;
Run with: $1, n$

PRIKLAD: $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$, $p(r) = \frac{5}{20}$, $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$, vstup $barbaraabarboraubaru$, redundance

- optimální při $|\sum_k p_k = \sum_{i=a}^{j} p_i - \sum_{i=j+1}^{b} p_i = \sum_l p_l|$, $k, l \in \{a, \ldots, b\}, \{k\} \cap \{l\} = \emptyset$, minimální

Huffmanovo kódování

- David A. Huffman: A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pp. 1098–1102, 1952.
- optimální prefixové, vyplývá z vlastností optimálního prefixového kódu (viz Věta dříve) a následujícího Lemma

Huffmanovo kódování

- David A. Huffman: A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pp. 1098–1102, 1952.
- optimální prefixové, vyplývá z vlastností optimálního prefixového kódu (viz Věta dříve) a následujícího Lemma

Lemma

Nechť $C' : A' \mapsto B^+$, *kde* $A' = \{a'_1, \ldots, a'_{n'}\}, n' \geq m \geq 2, a'_i = a_i \in A, i < n'$, *pravděpodobnosti výskytu symbolů* a'_i *jsou* $p'_i = p_i, i < n', p'_{n'} = \sum_{j=n-m'+1}^{n} p_j$, $A = \{a_1, \ldots, a_n\}$, $n = n' + m' - 1$, *s pravděpodobnostmi výskytu* p_i *symbolů* a_i, *kde* $p_{n-m'+1}, \ldots, p_n$ *jsou nejmenší,* $m' \in \{2, 3, \ldots, m\}, m' \equiv n \pmod{(m-1)}$, $B = \{b_1, \ldots, b_m\}$, *je optimální prefixový kód ze zdrojové abecedy* A' *do kódové abecedy* B.
Pak kód $C : A \mapsto B^+$, $C(a_i) = C'(a'_i), i < n', C(a_{n-m'+j}) = C'(a'_{n'})b_j, j \leq m'$, *ze zdrojové abecedy* A *do kódové abecedy* B *je také optimální.* □

Huffmanovo kódování

Statický a semi-adaptivní model

Input : abeceda $A' = \{a'_1, \ldots, a'_{n'}\}$, pravděpodobnosti $\{p'_1, \ldots, p'_{n'}\}$ výskytu a'_i
Uses : zdrojová abeceda $A = \{a_1, \ldots, a_n\}$, pravděpodobnosti $\{p_1, \ldots, p_n\}$ výskytu a_i, kódová abeceda $B = \{b_1, \ldots, b_m\}$
Output : kód $C(A')$

setřiď a'_i a p'_i tak, že $p'_i \geq p'_j$ pro $i < j$;
if $n' \leq m$ **then**
    $C(a'_i) \leftarrow b_i$ pro $i = 1, \ldots, n'$;
else
    if $n' = n$ **then**
        $m' \leftarrow (n-2) \mod (m-1) + 2$;
    else
        $m' = m$
    $C(A') \leftarrow$ zavolej se rekurzivně s $A' \setminus \{a'_{n'-m'+2}, \ldots, a_{n'}\}, \{p'_1, \ldots, p'_{n'-m'}, \sum_{i=n'-m'+1}^{n'} p'_i\}$;
    $C(a'_{n'-m'+i}) \leftarrow C(a'_{n'-m'+1})b_i$ pro $i = 1, \ldots, m'$;
Run with: $A = \{a_1, \ldots, a_n\}, \{p_1, \ldots, p_n\}$

PRIKLAD: $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$, $p(r) = \frac{5}{20}$, $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$, $m = 2$, vstup $barbaraabarboraubaru$, $m = 3$, vstup ??, redundance

Huffmanovo kódování

Proč m' a ne m? Protože při tomto počtu nejdelších kódových slov bude u všech kratších kódových slov a prefixů využito všech m symbolů kódové abecedy a kód má být optimální prefixový.

Huffmanův strom $T_H(A) = \langle V_H(A), E_H(V_H(A)) \rangle$ = reprezentace Huffmanova kódu $C(A)$ formou m-árního stromu:

- listové uzly $v_l(a_i) \in V_H(A)$ pro symboly $a_i \in A, i = 1, \ldots, n$
- vnitřní uzly $v(a'_{n'}) \in V_H(A)$ pro všechny $a'_{n'}$ (ve všech rekurzivních voláních) + kořenový uzel $v_r \in V_H(A)$
- hrany $\langle v(a'_{n'}), v(a'_{n'-m'+i}) \rangle_{b_i} \in E_H(V_H(A))$ a $\langle v_r, v(a'_i) \rangle_{b_i} \in E_H(V_H(A))$ označené $b_i \in B$ z uzlu pro $a'_{n'}$ následujícího rekurzivního volání do uzlů pro $a'_{n'-m'+i}, i = 1, \ldots, m'$ při $n' > m$ a z kořenového uzlu do uzlů pro $a'_i, i = 1, \ldots, n'$ při $n' \leq m$
- $C(a) =$ zřetězení $b \in B$ označujících hrany na cestě stromem z v_r do $v_l(a)$

PRIKLAD

Huffmanovo kódování

- minimální rozdíly v délkách kódových slov (pro jejich minimální bufferování při pevné rychlosti přenosu/ukládání): třídění a'_i a p'_i tak, že původní $a'_{n'}$ bude po zatřídění $a'_i, i < j$ pro všechna j, pro která $p'_j = p'_i$
- $m = 2$ a $p_i > \sum_{j=i+2}^{n} p_j$, $p_i \geq p_j$ pro $i < j$ (speciálně $p_i = 2^{-i}, i = 1, \ldots, n-1$ a $p_n = 2^{-(n-1)}$) $\rightarrow$ unární číselný kód $i - 1$ pro $a_i, i = 1, \ldots, n-1$ a n **I** pro a_n
- $m = 2$ a $p_1 < p_{n-1} + p_n$, $p_i \geq p_j$ pro $i < j$ (speciálně $p_i = \frac{1}{n}$) $\rightarrow$ blokový kód délky $k = \lfloor \log_2 n \rfloor$ pro $a_1, \ldots, a_{2^k}$ a délky $k + 1$ pro $a_{2^k+1}, \ldots, a_n$

PRIKLADY

Huffmanovo kódování

Adaptivní model – binární kód

- triviálně: znovuvytváření kódu pro každý další symbol na vstupu – výpočetně náročné
- Faller, Gallager, Knuth, Vitter
- vlastnost Huffmanova stromu (tzv. sibling property): $p'_{n'} \leq \ldots \leq p'_{n'-m'+1} \leq p'_{n'} \leq \ldots \leq p'_{n'-m'+1}$ následujícího rekurzivního volání – musí stále platit $\rightarrow$ v následujících algoritmech zajištěno pomocí (aktuálního) počtu $n(x)$ výskytů symbolu x a čísla $i(v(x)), v(x) \in V_H(A)$
- speciální (escape) symbol e značící neexistující/první výskyt symbolu

$T_H(A) \leftarrow \langle \{v_l(e)\}, \emptyset \rangle$, $C(e) \leftarrow$ prázdný řetězec;
$n(e) \leftarrow 0$;
$i(v_l(e)) \leftarrow 1$;
while načti ze vstupu symbol $a \in A$ **do**
    if $v_l(a) \notin V_H(A)$ **then**
        zapiš na výstup $C(e)$ a kód a;
    else
        zapiš na výstup $C(a)$;
    zavolej následující algoritmus;

Huffmanovo kódování

Adaptivní model – binární kód

if $v_l(a) \notin V_H(A)$ **then**
$V_H(A) \leftarrow V_H(A) \cup \{v_l(a), v(x)\}$;
$n(a) \leftarrow n(x) \leftarrow 1$;
$i(v(x)) \leftarrow i(v_l(e))$; $i(v_l(a)) \leftarrow i(v(x)) + 1$; $i(v_l(e)) \leftarrow i(v_l(a)) + 1$;
$E_H(V_H(A)) \leftarrow (E_H(V_H(A)) \setminus \{\langle u, v_l(e)\rangle_b\}) \cup \{\langle v(x), v_l(e)\rangle_{\mathbf{1}}, \langle v(x), v_l(a)\rangle_{\mathbf{0}}, \langle u, v(x)\rangle_b\}$;
else
$v(x) \leftarrow v_l(a)$;
while $v(x) \neq v_r$ **do**
if $\langle v(x), v_l(e)\rangle \notin E_H(V_H(A))$ **then**
najdi $v(y)$ takové, že $i(v(y)) < i(v(z)), \forall v(z) \in V_H(A), n(y) = n(z)$;
if $v(y) \neq v(x) \wedge \langle v(y), v(x)\rangle \notin E_H(V_H(A))$ **then**
$v \leftarrow v(x)$; $v(x) \leftarrow v(y)$; $v(y) \leftarrow v$;
$i \leftarrow i(v(x))$; $i(v(x)) \leftarrow i(v(y))$; $i(v(y)) \leftarrow i$;
$n(x) \leftarrow n(x) + 1$;
$v(x) \leftarrow u, \langle u, v(x)\rangle \in E_H(V_H(A))$;
$n(x) \leftarrow n(x) + 1$;

PRIKLAD

Huffmanovo kódování

Adaptivní model – binární kód

$T_H(A) \leftarrow \langle \{v_l(e)\}, \emptyset \rangle$;
$n(e) \leftarrow 0$;
$i(v_l(e)) \leftarrow 1$;
while není konec vstupu **do**
    $v \leftarrow v_r$;
    while $v \neq v_l(a)$ **do**
        načti ze vstupu symbol $b \in B$;
        $v \leftarrow u, \langle v, u \rangle_b \in E_H(V_H(A))$;
    if $a = e$ **then**
        načti ze vstupu kód symbolu $a \in A$;
        dekóduj kód a a zapiš na výstup a;
    else
        zapiš na výstup symbol $a \in A$;
    zavolej předchozí algoritmus;

PRIKLAD

Huffmanovo kódování

Aplikace

- často v návaznosti na jiné metody, např. na diferenční kódování (obraz, zvuk)

Aritmetické kódování

- průměrná délka optimálního prefixového kódu, např. Huffmanova, je minimálně rovna entropii zdroje a nejvýše o 1 větší než entropie (Shannon noisless coding theorem, viz Věta dříve) – platí těsnější, nejvýše o nejvyšší pravděpodobnost $p_{max} \geq 0.5$ zdrojových symbolů nebo o $p_{max} + 0.086, p_{max} < 0.5$
- změna zdrojové abecedy na k-tice (nezávislých) symbolů z původní abecedy A pro přiblížení se entropii ale zvyšuje velikost abecedy, a tím i Huffmanova stromu, na $|A|^k$, např. pro $p_1 = 0.95, p_2 = 0.03, p_3 = 0.02$ je entropie přibližně 0.335 b/symbol, průměrná délka Huffmanova kódu 1.05 b/symbol, kódu pro 9 dvojic symbolů přibližně 0.611 b/symbol a kódu pro $?$?tic symbolů přibližně $?$ b/symbol!

$\Rightarrow$ výhodnější kódovat zdrojová slova než samostatné symboly – ale nevytvářet kód pro všechna slova dané délky, např. Huffmanův!

$\rightarrow$ kód pouze pro zdrojová slova na vstupu

- vhodné pro malé zdrojové abecedy, např. binární, s velkými rozdíly v pravděpodobnostech symbolů

$=$ kódování zdrojových slov do čísel z podintervalů $[0, 1)$ kódovaných do binárního kódu

Aritmetické kódování

- Pasco, Rissanen, 1976, Rissanen, Langdon, 1979
- využití distribuční funkce/kumulované pravděpodobnosti (cumulative distribution function) $F_X(i) = \sum_{k=1}^{i} P(X = k), P(X = k) = P(a_k) = p_k$ zdroje (nezávisle se stejným pravděpodobnostním rozložením se vyskytujících) symbolů z abecedy $A = \{a_1, a_2, \ldots, a_n\}$ jako náhodných proměnných $X(a_i) = i$, $F_X(0) = 0$

$l_p \leftarrow 0$; $u_p \leftarrow 1$;
while načti ze vstupu symbol $a_i \in A$ **do**
    $l \leftarrow l_p + (u_p - l_p)F_X(i-1)$;
    $u \leftarrow l_p + (u_p - l_p)F_X(i)$;
    $l_p \leftarrow l$; $u_p \leftarrow u$;
    `// přeškálování` $[l, u)$
zapiš na výstup C = binární reprezentace jakéhokoliv čísla z $[l, u)$ s minimem bitů;
PRIKLAD

Aritmetické kódování

- $C(A^+)$ je prefixový binární kód ze zdrojových slov nad abecedou A, průměrná délka pro slova délky k je $H(A^k) \leq \bar{l}(C(A^k)) < H(A^k) + 1 \Rightarrow$ průměrná délka na symbol z A je $H(A) \leq \bar{l} < H(A) + \frac{1}{k}$
- pro dekódování je nutné znát délku L kódovaného zdrojového slova $\rightarrow$ uložit spolu s komprimovanými daty nebo speciální zdrojový symbol značící konec vstupu

$l_p \leftarrow 0$; $u_p \leftarrow 1$;
$j \leftarrow 0$;
načti ze vstupu binární reprezentaci čísla $x \in [0, 1)$;
while $j < L$ **do**
    najdi $i \in \{1, \ldots, n\}$ takové, že $F_X(i-1) \leq \frac{x - l_p}{u_p - l_p} < F_X(i)$;
    zapiš na výstup symbol $a_i \in A$;
    $j \leftarrow j + 1$;
    if $j < L$ **then**
        $l \leftarrow l_p + (u_p - l_p) F_X(i-1)$;
        $u \leftarrow l_p + (u_p - l_p) F_X(i)$;
        $l_p \leftarrow l$; $u_p \leftarrow u$;
        `// přeškálování` $[l, u)$

PRIKLAD

Aritmetické kódování

- u kódování i dekódování žádoucí průběžný výstup během čtení vstupu, ne až po načtení celého vstupu $\rightarrow$ kód čísla z $[l, u)$ průběžně
- $[l, u)$ se s délkou zdrojového slova zmenšuje, ale uložení necelých čísel je v praxi s omezenou přesností $\rightarrow$ omezení délky slova nebo přeškálování $[l, u)$:
  1. $u < 0.5$: $x \leftarrow 2x, x \in \{l, u\}$
  2. $l \geq 0.5$: $x \leftarrow 2(x - 0.5), x \in \{l, u\}$
  3. $l \geq 0.25 \wedge u < 0.75$: $x \leftarrow 2(x - 0.25), x \in \{l, u\}$
  - případy $3\overbrace{\ldots}^{c\text{-krát}}31 = 12\overbrace{\ldots}^{c\text{-krát}}2$ a $3\overbrace{\ldots}^{c\text{-krát}}32 = 21\overbrace{\ldots}^{c\text{-krát}}1$

Aritmetické kódování

$c \leftarrow 0$;
// while ...
while $u < 0.5 \vee l \geq 0.5 \vee (l \geq 0.25 \wedge u < 0.75)$ **do**
    if $l \geq 0.25 \wedge u < 0.75$ **then**
        // případ 3
        $c \leftarrow c + 1$;
        $d \leftarrow 0.25$;
    else
        if $u < 0.5$ **then**
            // případ 1
            $b \leftarrow \mathbf{0}$;
            $d \leftarrow 0$;
        else
            // případ 2
            $b \leftarrow \mathbf{I}$;
            $d \leftarrow 0.5$;
        zapiš na výstup b;
        while $c > 0$ **do**
            zapiš na výstup inverzi b;
            $c \leftarrow c - 1$;
    $l \leftarrow 2(l - d)$;
    $u \leftarrow 2(u - d)$;

zapiš na výstup $C = \mathbf{I}$;

DOKLAD

Aritmetické kódování

načti ze vstupu $\lceil -\log_2 p_{min} \rceil$ bitů binární reprezentace čísla $x \in [0, 1)$, p_{min} nejnižší pravděpodobnost zdrojových symbolů;

```
// while ...
```

while $u < 0.5 \vee l \geq 0.5 \vee (l \geq 0.25 \wedge u < 0.75)$ **do**

    if $l \geq 0.25 \wedge u < 0.75$ **then**

        $d \leftarrow 0.25$;

    else

        if $u < 0.5$ **then**

            $d \leftarrow 0$;

        else

            $d \leftarrow 0.5$;

    $l \leftarrow 2(l - d)$;

    $u \leftarrow 2(u - d)$;

    načti ze vstupu další bit b anebo $b \leftarrow \mathbf{0}$;

    $x \leftarrow 2(x - d) + b\frac{1}{2^{\lceil -\log_2 p_{min} \rceil}}$;

PRIKLAD

Aritmetické kódování

Celočíselná implementace

- zobrazení $[0,1)$ na $[0,M-1)$ ($0.25 \mapsto \frac{M}{4}$, $0.5 \mapsto \frac{M}{2}$, $0.75 \mapsto \frac{3M}{4}$), $M \geq 4\frac{1}{p_{min}}$, M typicky 2^8, 2^{16}, 2^{32} nebo 2^{64} podle datového typu pro čísla z $[0,M-1)$
- odhad $F_X(i)$ s frekvencemi/četnostmi $f(a_k) = \frac{n(a_k)}{\sum_{j=1}^{|A|} n(a_j)}$ výskytu symbolu $a_k \in A$ místo pravděpodobností $P(a_k)$
- $l \leftarrow l_p + \lfloor (u_p - l_p + 1)F_X(i-1) \rfloor$, $u \leftarrow l_p + \lfloor (u_p - l_p + 1)F_X(i) \rfloor - 1$, $\frac{x - l_p + 1}{u_p - l_p + 1}$, $u \leftarrow 2(u-d)+1$, $x \leftarrow 2(x-d)+b$ – kvůli celočíselné aritmetice
- načti ze vstupu $\lceil \log_2 M \rceil$ bitů binární reprezentace čísla $x \in [0, M-1)$
- při $M = 2^k$ pro nějaké $k \geq 2$:
  - $u < \frac{M}{2} \rightarrow$ nejvýznamnější bit l i u je $\mathbf{0}$
  - $l \geq \frac{M}{2} \rightarrow$ nejvýznamnější bit l i u je $\mathbf{I}$
  - $l \geq \frac{M}{4} \wedge u < \frac{3M}{4} \rightarrow$ druhý nejvýznamnější bit l je $\mathbf{I}$ a u je $\mathbf{0}$
  - $2x$ a $2(x - \frac{M}{2}) \rightarrow$ bitový posun x doleva o 1 bit, $2(x - \frac{M}{4}) \rightarrow$ navíc inverze (nového) nejvýznamnějšího bitu

PRIKLAD

Aritmetické kódování

Adaptivní model

- průběžný odhad $F_X(i)$ s frekvencemi/četnostmi $f(a_k) = \frac{n(a_k)}{\sum_{j=1}^{|A|} n(a_j)}$ výskytu symbolu $a_k \in A$ místo pravděpodobností $P(a_k)$
  - inicializace na známý odhad nebo počet $n(a) = 1$ výskytu každého symbolu $a \in A$
  - inkrementace $n(a)$ po (de)kódování symbolu a
- uložení p_{min} spolu s komprimovanými daty, u celočíselné implementace nesmí p_{min} klesnout pod $4\frac{1}{M} \rightarrow$ v praxi $n(a_j) \leftarrow \frac{n(a_j)}{2}$ pro $n(a_j) > 1$ při $\sum_{j=1}^{|A|} n(a_j) = \frac{M}{4}$

Aplikace

- ve standardech komprese multimediálních dat (obraz, video, zvuk)

Aritmetické kódování – QM kódování

= modifikované adaptivní binární aritmetické kódování (Q kódování, skew kódování), tj. pro binární zdrojovou abecedu s adaptivním modelem

- $A = u - l$ místo u: $l \leftarrow l_p + A_p F_X(i-1)$ a $A \leftarrow A_p p_i$
- místo 2 symbolů A, $a_1 = \mathbf{0}$ a $a_2 = \mathbf{I}$, $a_1 =$ více (MPS) a $a_2 =$ méně (LPS) pravděpodobný symbol s průběžně odhadovanými frekvencemi/četnostmi $1-q$ a q výskytu
  - $l \leftarrow l_p$ a $A \leftarrow A_p(1-q)$ pro MPS
  - $l \leftarrow l_p + A_p(1-q)$ a $A \leftarrow A_p q$ pro LPS
  - $\frac{x - l_p}{A_p} < 1 - q$ pro MPS a $\geq 1 - q$ pro LPS
- potlačení násobení (i pro nebinární zdrojové abecedy) – udržování hodnoty A v $[0.75, 1.5)$ a zanedbání násobení A_p
  - $l \leftarrow l_p$ a $A \leftarrow A_p - q$ pro MPS
  - $l \leftarrow l_p + A_p - q$ a $A \leftarrow q$ pro LPS
  - $x - l_p < 1 - q$ pro MPS a $\geq 1 - q$ pro LPS
  - přeškálování l, A: $A < 0.75$: $A \leftarrow 2A$, $l \leftarrow 2l$ pro $l < 0.5$ a $l \leftarrow 2(l - 0.5)$ pro $l \geq 0.5$

Aritmetické kódování – QM kódování

while $A < 0.75$ **do**
    if $l < 0.5$ **then**
        $b \leftarrow \mathbf{0}$;
        $d \leftarrow 0$;
    else
        $b \leftarrow \mathbf{I}$;
        $d \leftarrow 0.5$;
    zapiš na výstup b;
    $l \leftarrow 2(l - d)$;
    $A \leftarrow 2A$;

zapiš na výstup $C = \mathbf{I}$;

načti ze vstupu $\lceil -\log_2 q \rceil$ bitů binární reprezentace čísla $x \in [0, 1)$;
`// while ...`
while $A < 0.75$ **do**
    if $l < 0.5$ **then**
        $d \leftarrow 0$;
    else
        $d \leftarrow 0.5$;
    $l \leftarrow 2(l - d)$;
    $A \leftarrow 2A$;
    načti ze vstupu další bit b anebo $b \leftarrow \mathbf{0}$;
    $x \leftarrow 2(x - d) + b\frac{1}{2^{\lceil -\log_2 q \rceil}}$;

Aritmetické kódování – QM kódování

- inicializace $q = 0.5$ a aktualizace q podle tabulky hodnot při přeškálování, ne po (de)kódování každého symbolu
- při častějším výskytu LPS než MPS, $q > A - q$, prohození symbolů včetně aktuálních hodnot l a A, při přeškálování
- celočíselná implementace: zobrazení $[0, 1.5)$ na $[0, M-1)$ ($0.25 \mapsto \frac{M}{6}$, $0.5 \mapsto \frac{M}{3}$, $0.75 \mapsto \frac{M}{2}, 1 \mapsto \frac{2M}{3}$), $M \geq \frac{4}{3}\frac{1}{q}$, i pro hodnoty q
- použití ve standardu JPEG (JBIG) komprese obrazu

Kontextové (context-based) metody

Kontextové (context-based) metody

- výskyt symbolu na vstupu není nezávislý na výskytu ostatních symbolů

$=$ pro modelování symbolu na vstupu využití kontextu délky k = posloupnost (bezprostředně) předchozích k symbolů symbolu na vstupu $\rightarrow$ Markovův model k-tého řádu

$\rightarrow$ větší rozdíly v (podmíněných) pravděpodobnostech výskytu symbolů v kontextu jiných symbolů $\rightarrow$ lepší predikce symbolu (např. na konci slova) $\rightarrow$ vyšší míra komprese

Příklad

vstup $barbaraabarboraubaru$, $A = \{a, b, r, u, o\}$
$f(a) = \frac{7}{20}$, $f(b) = \frac{5}{20}$, $f(r) = \frac{5}{20}$, $f(u) = \frac{2}{20}$, $f(o) = \frac{1}{20}$,
$f(a|a) = \frac{1}{7}$, $f(b|a) = \frac{1}{7}$, $f(r|a) = \frac{4}{7}$, $f(u|a) = \frac{1}{7}$, $f(a|b) = \frac{4}{5}$, $f(o|b) = \frac{1}{5}$, $f(a|r) = \frac{2}{5}$,
$f(b|r) = \frac{2}{5}$, $f(u|r) = \frac{1}{5}$, $f(b|u) = 1$, $f(r|o) = 1$, ostatní $f = 0$
$f(b|aa) = 1$, $f(a|ab) = 1$, $f(a|ar) = \frac{1}{4}$, $f(b|ar) = \frac{2}{4}$, $f(u|ar) = \frac{1}{4}$, $f(b|au) = 1$,
$f(r|ba) = 1$, $f(r|bo) = 1$, $f(a|ra) = \frac{1}{2}$, $f(u|ra) = \frac{1}{2}$, $f(a|rb) = \frac{1}{2}$, $f(o|rb) = \frac{1}{2}$,
$f(a|ub) = 1$, $f(a|or) = 1$, ostatní $f = 0$

PPM (Prediction with partial match)

- vyšší pravděpodobnost výskytu symbolu v delším kontextu, ale počet kontextů délky k je $|A|^k$ (A abeceda symbolů)

$\rightarrow$ nezjišťovat pravděpodobnosti výskytu symbolu pro všechny kontexty dané délky, ale pouze pro kontexty na vstupu

- speciální (escape) symbol e abecedy značící neexistující/první výskyt symbolu na vstupu v aktuálním kontextu

$=$ adaptivní modelování pravděpodobností výskytu symbolů ve zkracujícím se kontextu jako posloupnosti bezprostředně předchozích symbolů

- Cleary, Witten, 1984
- průběžné odhady (podmíněných) pravděpodobností $P(a_i|c^k)$ a distribučních funkcí/ kumulovaných pravděpodobností $F_X(i|c^k)$ výskytu symbolů z abecedy $A|c^k \subseteq A \cup \{e\} = \{a_1, a_2, \ldots, a_n, e\}$ jako náhodných proměnných $X(a_i|c^k) = i|c^k$ s frekvencemi/ četnostmi $f(a_i|c^k) = \frac{n(a_i|c^k)}{\sum_{j=1}^{|A|c^k|} n(a_j|c^k)}$ výskytu symbolu $a_i \in A|c^k$ v kontextu c_k délky k
  - výskyt v kontextu c^0 = výskyt (bez kontextu)
  - výskyt v kontextu c^{-1} = neexistující/první výskyt, $A|c^{-1} = A$, $n(a_i|c^{-1}) = 1$ pro všechny $a_i \in A$

PPM (Prediction with partial match)

while načti ze vstupu symbol $a \in A$ **do**
  $k \leftarrow K$;
  while a se nevyskytl v kontextu $c^k \wedge k \geq 0$ **do**
    kóduj speciální symbol e v kontextu c^k;
    $k \leftarrow k - 1$;
  zapiš na výstup kód symbolu a v kontextu c^k;
  `// algoritmus datové reprezentace` $n(a|c^k)$ `nebo`
  $n(a|c^k) \leftarrow n(a|c^k) + 1$ pro všechny $c_k, K \geq k \geq 0$;

- $f(e|c^k)$? ze začátku (relativně) velká, s počtem zpracovaných symbolů klesající
  - metoda A (PPMA): $n(e|c^k) = 1$
  - metoda B (PPMB): $n(e|c^k) = |A|c^k \setminus \{e\}|$ a $n(a|c^k) \leftarrow n(a|c^k) - 1$ pro $a \in A|c^k \setminus \{e\}$ (při $n(a|c^k) = 0$ vyřazení a z $A|c^k \setminus \{e\}$) – větší šance nového symbolu v kontextu s více symboly v kontextu
  - metoda C (PPMC, Moffat): $n(e|c^k) = |A|c^k \setminus \{e\}|$ – v průměru nejvyšší míra komprese
  - PPMP: odhad $\frac{d_1}{d} - \frac{d_2}{d^2} + \frac{d_3}{d^3} - \ldots$, d_i symbolů vyskytujících se na vstupu i-krát, na základě Poissonova rozdělení pravděpodobnosti výskytu d symbolů na vstupu
  - PPMX: přibližná verze PPMP s odhadem $\frac{d_1}{d}$, PPMC při $d_1 = 0$ nebo $d_1 = d$

PRIKLAD

PPM (Prediction with partial match)

$k \leftarrow K$;
while načti ze vstupu a dekóduj kód symbolu $a \in A|c^k$ v kontextu c^k **do**
    if $a = e$ **then**
        $k \leftarrow k - 1$;
    else
        zapiš na výstup symbol $a \in A$;
        `// algoritmus datové reprezentace` $n(a|c^k)$ `nebo`
        $n(a|c^k) \leftarrow n(a|c^k) + 1$ pro všechny $c_k, K \geq k \geq 0$;
        $k \leftarrow K$;

PRIKLAD

- K? co nejvyšší? – vyšší pravděpodobnost výskytu symbolu v delším kontextu
  - vyšší pravděpodobnost kódování speciálního symbolu e, delší kontexty bývají neaktuální – míra komprese nejprve s K prudce roste, ale pak mírně klesá $\rightarrow$ v praxi 5 nebo 6 pro textová data
  - PPM*: v delších kontextech bývá výskyt jen jednoho symbolu = deterministický kontext $\rightarrow$ délka nejkratšího, jinak délka nejdelšího nedeterministického, varianta PPMZ (Bloom)
- kódování symbolů (adaptivním) aritmetickým kódováním s dynamickým pravděpodobnostním modelem podmíněným aktuálním kontextem c^k

PPM (Prediction with partial match)

Exclusion principle

- menší abeceda znamená kratší kódy symbolů

= když se symbol a nevyskytl v kontextu c^k délky $K \geq k \geq 0$, symboly vyskytující se v c^k, tzn. z $A|c^k \setminus \{e\}$, lze vyřadit ze všech abeced $A|c^l$ kontextů c^l délky $-1 \leq l < k$ – pouze pro kódování a, aktuální kontext se mění!

PRIKLAD

Datová reprezentace $n(a|c^k), a \in A$ pro všechny kontexty $c^k, K \geq k \geq 0 = n$-ární strom $T = \langle V, E(V) \rangle$ (n velikost A)

- listové uzly $v_l(a|c^{k_{max}}) \in V$ pro symboly $a \in A$ pro kontext $c^{k_{max}}, k_{max} = \max_{c^k} k$
- vnitřní uzly $v(a_{-i}|c^k) \in V$ pro symboly $a_{-i}, i = -k, \ldots, -1$ kontextu $c^k = a_{-k}c^{k-1}$, $c^1 = a_{-1}$ + kořenový uzel $v_r \in V$
- hrany $\langle v(a_{-1}|c^{k_{max}}), v_l(a|c^{k_{max}}) \rangle \in E(V)$ nebo $\langle v_r, v_l(a|c^0) \rangle \in E(V)$, a $\langle v(a_{-i}|c^k), v(a_{-i+1}|c^k) \rangle \in E(V)$ a $\langle v_r, v(a_{-k}|c^k) \rangle \in E(V)$

= trie = v uzlech část prvku (symbol a_{-i} kontextu c^k), ne celý prvek (kontext c^k)

- $s(v(a|c^k)) = v(a|c^{k-1}), k \geq 1$ a $s(v(a|c^0)) = v_r$ pro tentýž symbol a

PPM (Prediction with partial match)

$T \leftarrow \langle \{v_r\}, \emptyset \rangle$;
$v_p \leftarrow v_r$;
// while ...
while $k \geq 0$ **do**
    if $v_p \neq v_r$ **then**
        $v_p \leftarrow s(v_p)$;
    if $\langle v_p, v_l(a|c^k) \rangle \notin E(V)$ **then**
        $V \leftarrow V \cup \{v_l(a|c^k)\}$;
        $n(a|c^k) \leftarrow 1$;
        $E(V) \leftarrow E(V) \cup \{\langle v_p, v_l(a|c^k) \rangle\}$;
    else
        $n(a|c^k) \leftarrow n(a|c^k) + 1$;
    if $k = k_{max}$ **then**
        $v_p \leftarrow v_l(a|c^k)$;
    $k \leftarrow k - 1$;

PRIKLAD

PPM (Prediction with partial match)

- pro každý symbol $a \in A$ na vstupu přidáno 0 až $K + 1$ uzlů $\rightarrow$ při velkém smazání a konstrukce nového stromu z posledních několika symbolů (v praxi 2048)

PAQ

- kontext nemusí být posloupnost bezprostředně předchozích symbolů na vstupu, jako u PPM

$\rightarrow$ kombinace modelů symbolu na vstupu využívajících různé kontexty = context mixing

$=$ adaptivní modelování pravděpodobností výskytu binárních symbolů (bitů, z binární zdrojové abecedy) kombinací modelů s různými kontexty

- Matt Mahoney, 2002 a další (Osnach, Ratushnyak – PAQAR, Skibiński – PAsQDa, Taylor – RK) – několik verzí (8 hlavních) a mnoho variant (pro různé typy dat), free software, postupná vylepšení místo zcela nových metod z 80. až 90. let
- kontexty např.:
  - posloupnost bezprostředně předchozích 0 až 63 bitů (po osmi) na vstupu, jako u PPM
  - bezprostředně předchozí 0 a 1 slovo na vstupu – např. znak textu
  - nejvýznamnější bity bezprostředně předchozích slov – pro multimediální data (zvuk, obraz)
  - sousední slova vícedimenzionálních dat (stejná ve stejné vzdálenosti) – např. obrazové body
  - vybraná předchozí slova („řídký kontext“) – pro různé binární soubory, např. spustitelné, již (ztrátově) komprimované aj.
- nový model z původního a 10-bitového kontextu na základě aktualizované tabulky = SSE (secondary symbol estimation) – od verze 2

PAQ

- i-tý model bitu na vstupu:
  - počty $n_i(a), a = \{\mathbf{0}, \mathbf{I}\}$ bitů v kontextu – verze 1 až 6, $n_i = n_i(\mathbf{0}) + n_i(\mathbf{I})$, $n_i(a) \leftarrow n_i(a) + 1$ a $n_i(\mathbf{I} - a) \leftarrow \lfloor 1 + \frac{n_i(\mathbf{I}-a)}{2} \rfloor$ jestliže $n_i(\mathbf{I} - a) > 2$, a hodnota modelovaného bitu
  - průběžné odhady (podmíněných) pravděpodobností $P_i(a|c), a = \{\mathbf{0}, \mathbf{I}\}$ – od verze 7: $P_i(a|c) = \frac{n_i(a)}{n_i}$
- průběžné odhady (podmíněných) pravděpodobností $P(a|c), a = \{\mathbf{0}, \mathbf{I}\}$ kombinací modelů:
  - $P(a|c) = \frac{s(a)}{s}, s = s(\mathbf{0}) + s(\mathbf{I}), s(a) = \epsilon + \sum_i w_i n_i(a)$ – verze 1 až 6, ϵ (experimentálně zjištěný) parametr pro nenulové $s(a)$, w_i váha i-tého modelu závisející na délce kontextu c – pevné (verze 1 až 3) nebo upravované pro minimalizaci chyby modelu a preferující přesnější modely (verze 4 až 6): $w_i \leftarrow \max[0, w_i + \frac{s n_i(\mathbf{I}) - s(\mathbf{I}) n_i}{s(\mathbf{0}) s(\mathbf{I})}(a - P(\mathbf{I}|c))]$
  - neuronovou sítí – od verze 7: $P(\mathbf{I}|c) = \frac{1}{1+e^{-\sum_i w_i x_i}}, x_i = \ln(\frac{P_i(\mathbf{I}|c)}{1-P_i(\mathbf{I}|c)})$, $w_i \leftarrow w_i + \mu x_i (a - P(\mathbf{I}))$, μ malá konstanta (míra adaptace)
- kódování bitů adaptivním binárním aritmetickým kódováním (QM kódováním??) s dynamickým pravděpodobnostním modelem $\{P(\mathbf{0}|c), P(\mathbf{I}, c)\}$ – podobně jako v PPM

Blokové třídění

$\sim$ Burrows-Wheelerova transformace (BWT)

- Michael Burrows, David J. Wheeler, 1994 (transformace Wheeler, 1983)
- nepoužívá (podmíněné) pravděpodobnosti výskytu symbolů v kontextu jiných symbolů ani posloupností symbolů
- blok b^k délky k: $c^{k-1}a$, a symbol na vstupu, kontext $c^{k-1} = a_{-k+1}a_{-k+2}\ldots a_{-1} =$ posloupnost bezprostředně předchozích $k-1$ symbolů symbolu a

$=$ permutace bloku b^k na blok obsahující posloupnosti stejných symbolů (viz dále)

Blokové třídění

$\sim$ Burrows-Wheelerova transformace (BWT)

- Michael Burrows, David J. Wheeler, 1994 (transformace Wheeler, 1983)
- nepoužívá (podmíněné) pravděpodobnosti výskytu symbolů v kontextu jiných symbolů ani posloupností symbolů
- blok b^k délky k: $c^{k-1}a$, a symbol na vstupu, kontext $c^{k-1} = a_{-k+1}a_{-k+2}\ldots a_{-1} =$ posloupnost bezprostředně předchozích $k-1$ symbolů symbolu a

$=$ permutace bloku b^k na blok obsahující posloupnosti stejných symbolů (viz dále) $\rightarrow$ move-to-front (MTF) kódování permutovaného bloku

Blokové třídění

while načti ze vstupu nejvýše K symbolů jako blok $b^k = b_1^k \in A^+$ **do**
    zapiš na výstup číslo k;
    $i \leftarrow 2$;
    while $i \leq k$ **do**
        $b_i^k \leftarrow$ rotace b_{i-1}^k o 1 symbol doleva;
        $i \leftarrow i + 1$;
    setřiď $b_1^k, \ldots, b_k^k$ lexikograficky;
    $i \leftarrow 1$;
    while $i \leq k$ **do**
        zapiš na výstup poslední symbol b_i^k;
        $i \leftarrow i + 1$;
    zapiš na výstup číslo i, kde $b^k = b_i^k$;

PRIKLAD

Blokové třídění

V lexikograficky setříděných blocích $b_1^k, \ldots, b_k^k$ z předchozího algoritmu:

Kódování

- posloupnost prvních symbolů bloků $b_i^k, i = 1, \ldots, k$ obsahuje posloupnosti stejných symbolů (kvůli setřídění b_i^k) $\Rightarrow$ posloupnost posledních symbolů bloků b_i^k obsahuje posloupnosti stejných symbolů, jestliže je v původním bloku b^k před prvním symbolem bloku b_i^k (lokálně) opakovaně poslední symbol bloku b_i^k (kvůli rotaci o 1 symbol doleva) = častý kontext

Dekódování

- posloupnost prvních symbolů bloků b_i^k – $F_i \in A, i = 1, \ldots, k$ v následujícím algoritmu = lexikograficky setříděná posloupnost posledních symbolů bloků b_i^k – $L_i \in A$ = permutace φ posledních symbolů – $F_{\varphi(i)} = L_i \Leftrightarrow F_j = L_{\varphi^{-1}(j)}$
- v původním bloku b^k je za posledním symbolem L_j bloku $b_j^k, b_j^k \neq b^k$ první symbol F_j bloku $b_j^k \Rightarrow F_{\varphi^{-1}(j)}$ je za $L_{\varphi^{-1}(j)} = F_j$

Blokové třídění

while načti ze vstupu číslo k **do**
    načti ze vstupu k symbolů $L_1, \ldots, L_k \in A$;
    $F_i \leftarrow L_i$ pro $i = 1, \ldots, k$;
    setřiď $F_1, \ldots, F_k$ lexikograficky;
    $a \leftarrow F_1$;
    $i_F(a) \leftarrow 1$;
    $i \leftarrow 2$;
    while $i \leq k$ **do**
        if $F_i \neq a$ **then**
            $a \leftarrow F_i$;
            $i_F(a) \leftarrow i$;
        $i \leftarrow i + 1$;
    $i \leftarrow 1$;
    while $i \leq k$ **do**
        $\varphi^{-1}(i_F(L_i)) \leftarrow i$;
        $i_F(L_i) \leftarrow i_F(L_i) + 1$;
        $i \leftarrow i + 1$;
    načti ze vstupu číslo j;
    $i \leftarrow 1$;
    while $i \leq k$ **do**
        zapiš na výstup symbol F_j;
        $j \leftarrow \varphi^{-1}(j)$;
        $i \leftarrow i + 1$;

PRIKLAD

Blokové třídění

Implementace

- maximální délka K bloku až statisíce symbolů (bytů)
- kódování pouze s původním blokem b^k – b_i^k ukazatele na první symbol bloku b_i^k v b^k
- dekódování pouze s posledními symboly $L_i \in A$ bloků $b_i^k, i = 1, \ldots, k$
- move-to-front (MTF) kódování permutovaného bloku následované run-length kódováním (RLE) a statistickým kódováním (Huffmanovým nebo aritmetickým)

Slovníkové metody

Slovníkové metody

- výskyt symbolu na vstupu není nezávislý na výskytu ostatních symbolů – symboly se často vyskytují (nebo naopak nevyskytují) v opakujících se vzorech (patterns), např. slova nebo části vět v textu
- nepoužívají statistický model, např. (podmíněné) pravděpodobnosti výskytu vzorů

$=$ pro kompresi slov na vstupu využívání často se vyskytujících vzorů symbolů $=$ posloupnosti předchozích symbolů aktuálního symbolu na vstupu

$\rightarrow$ vyhledávání vzorů na vstupu a jejich ukládání do slovníku a kódování odkazu na vzor ve slovníku při/místo kódování slov na vstupu $\rightarrow$ vyšší míra komprese

- slovník = (implicitně) posloupnost předchozích symbolů aktuálního symbolu na vstupu nebo (explicitně) datová struktura vzorů symbolů
- často se vyskytujících vzorů (ve slovníku) by mělo být relativně málo, vzhledem ke všem vzorům $\rightarrow$ výběr nejčastěji se vyskytujících vzorů
- pro specifické aplikace se známými často se vyskytujícími vzory statický, příp. semi-adaptivní, model/slovník (neukládání vzorů), jinak adaptivní – počáteční prázdný nebo malý výchozí a při naplnění nepřidávání, vymazání celého nebo nejdéle nepoužitých vzorů (least recently used, LRU)
- jednoduchá a rychlá dekomprese – oproti statistickým metodám

Kódování n-gramů

- použití statistického statického, příp. semi-adaptivního, modelu
- slovník = všechny symboly vstupní abecedy A + nejčastěji se vyskytující 2- až n-tice symbolů (n-gramy) na vstupu do velikosti slovníku, seřazené podle pravděpodobnosti výskytu sestupně

= kódování slov na vstupu indexem stejného slova ve slovníku

$x \leftarrow$ prázdný řetězec;
while načti ze vstupu symbol $a \in A$ **do**
    if xa je ve slovníku **then**
        $x \leftarrow xa$;
    else
        zapiš na výstup index x ve slovníku;
        $x \leftarrow a$;
if $x \neq$ prázdný řetězec **then**
    zapiš na výstup index x ve slovníku;

PRIKLAD

- kódování indexů statickým kódem nebo statistickým kódováním (Huffmanovým nebo aritmetickým)

LZ77

- také LZ1, Abraham Lempel, Jakob Ziv, 1977 – základ, mnoho variant $\rightarrow$ rodina metod LZ77
- adaptivní slovník $=$ posloupnost bezprostředně předchozích K symbolů aktuálního symbolu na vstupu (kontext délky K) – search buffer (délky K)

$=$ kódování (i prázdného) slova a dalšího symbolu na vstupu kódy pozice stejného slova začínajícího v search bufferu (nebo pozice 0), jeho délky a symbolu

- aktuální posloupnost L symbolů na vstupu – look-ahead buffer (délky L)
- search $+$ look-ahead buffer $=$ posuvné okno (délky $K + L$)

LZ77

$x \leftarrow$ prázdný řetězec, $o \leftarrow 0$, $l \leftarrow 0$;
while načti ze vstupu symbol $a \in A$ **do**
    if xa začíná v search bufferu a $l < L$ **then**
        $o \leftarrow$ nejmenší vzdálenost (v počtu symbolů) prvního symbolu xa v search bufferu od konce bufferu;
        $l \leftarrow l + 1$;
        $x \leftarrow xa$;
    else
        zapiš na výstup kódy o, l a a;
        $x \leftarrow$ prázdný řetězec, $o \leftarrow 0$, $l \leftarrow 0$;
if $x \neq$ prázdný řetězec **then**
    zapiš na výstup kódy o a l;

PRIKLAD

- kódování vzdáleností, délek a symbolů statickým kódem nebo (adaptivním) statistickým kódováním (Huffmanovým = LZH nebo aritmetickým)

Varianty LZ77

- kódování celých trojic vzdálenost-délka-symbol místo jednotlivě
- proměnlivé délky K a L bufferů, delší K a L častější a delší nález v search bufferu, ale delší hledání a větší vzdálenosti a délky

LZR (Rodeh)

- délky K a L bufferů neomezené

$=$ kompresní metoda z algoritmu LZ76 pro měření „složitosti“ textu hledáním předchozích výskytů slov

Varianty LZ77

LZSS (Storer, Szymanski)

- (bitový) příznak pro kódování dvojic vzdálenost-délka anebo symbolu – ukládány skupiny příznaků pro skupiny kódů, např. 8
- kódování samostatných symbolů, pokud by kód dvojice byl stejně dlouhý nebo delší (nebo i mírně kratší, kvůli náročnějšímu kódování dvojice)
- délky K a L tak, aby dvojice byla kódována do pevného dvojnásobného počtu bitů než symbol, např. $K = 2^{11}$ a $L = 2^5$ pro $|A| = 2^8$
- varianta LZB: kódování vzdáleností do postupně se zvyšujícího počtu bitů podle aktuální velikosti search bufferu a délek Eliasovým Gamma kódem
- varianta SLH: kódování vzdáleností a symbolů semi-adaptivním Huffmanovým kódem

Varianty LZ77

Deflate (Philip W. Katz)

- nejúspěšnější varianta LZ77 a LZSS – dvojice délka-vzdálenost
- volitelně (příp. redukované) vyhledání i delšího vzoru po kódování aktuálního symbolu na vstupu
- komprese vstupních dat po blocích různé délky
- mód 1 = bez komprese – 5 B záhlaví navíc (3 b v 1 B kód módu 1, 2 B délka a 2 B jedničkový komplement délky)
- mód 2: kódování dvojic a symbolů podle statických modelů
  - symboly a délky čísly $0 - 285$: $0 - 255$ symboly (byty), 256 konec bloku, $257 - 285$ s dalšími $0 - 5$ bity (pro určení hodnoty) délky $3 - 258 = L$, kódovaných statickým Huffmanovým kódem ($7 - 9$ bitů/číslo)
  - vzdálenosti $1 - 2^{15} = K$ čísly $0 - 29$ s dalšími $0 - 13$ bity (pro určení hodnoty v rozsahu), kódovaných binární reprezentací čísla (5 bitů/číslo)

Obrázek: Kódování symbolů/délek a vzdáleností v módu 2

Varianty LZ77

Code	Extra bits	Lengths	Code	Extra bits	Lengths	Code	Extra bits	Lengths
257	0	3	267	1	15,16	277	4	67–82
258	0	4	268	1	17,18	278	4	83–98
259	0	5	269	2	19–22	279	4	99–114
260	0	6	270	2	23–26	280	4	115–130
261	0	7	271	2	27–30	281	5	131–162
262	0	8	272	2	31–34	282	5	163–194
263	0	9	273	3	35–42	283	5	195–226
264	0	10	274	3	43–50	284	5	227–257
265	1	11,12	275	3	51–58	285	0	258
266	1	13,14	276	3	59–66			

edoc	Bits	Prefix codes
0–143	8	00110000–10111111
144–255	9	110010000–111111111
256–279	7	0000000–0010111
280–287	8	11000000–11000111

Code	Extra bits	Distance	Code	Extra bits	Distance	Code	Extra bits	Distance
0	0	1	10	4	33–48	20	9	1025–1536
1	0	2	11	4	49–64	21	9	1537–2048
2	0	3	12	5	65–96	22	10	2049–3072
3	0	4	13	5	97–128	23	10	3073–4096
4	1	5,6	14	6	129–192	24	11	4097–6144
5	1	7,8	15	6	193–256	25	11	6145–8192
6	2	9–12	16	7	257–384	26	12	8193–12288
7	2	13–16	17	7	385–512	27	12	12289–16384
8	3	17–24	18	8	513–768	28	13	16385–24576
9	3	25–32	19	8	769–1024	29	13	24577–32768

Varianty LZ77

Deflate (Philip W. Katz)

- mód 3: kódování dvojic a symbolů semi-adaptivními Huffmanovými kódy (pro symboly/délky a vzdálenosti s čísly podle módu 2, max. 15 bitů) napříč bloky, kódovaných jako posloupnosti délek kódů RLE (s min. počtem 4 stejných symbolů za sebou) a (semi-adaptivním) Huffmanovým kódem uloženým jako modifikovaná posloupnost délek kódů (3 bity/délka)
  - (ekvivalentní) Huffmanův kód C' z posloupnosti $l(a_1), \ldots, l(a_n)$ délek kódových slov $C(a_i)$ Huffmanova kódu C: $C'(a_i) =$ binární reprezentace čísla $B_j + k$ délky $l(a_i) = j$, kde $B_j = 2(B_{j-1} + |\{a\}, l(a) = j - 1|), B_1 = 0$, $a_{i'_0}, \ldots, a_{i'_k} = a_i, l(a_{i'}) = j$
    PRIKLAD

Varianty LZ77

LZPP (Pylak)

- varianta LZSS – minimální délka 3 nalezeného slova
- většina vzdáleností a délek je malých $\Rightarrow$ velká entropie $\rightarrow$ modifikované adaptivní aritmetické kódování (range encoding):
  - po bytech, pravděpodobnosti výskytu bez kontextu a v kontextu délky 1 – s vyloučením symbolů za vyhledaným slovem v search bufferu (exclusion principle)
  - speciální (escape) symbol abecedy pro neexistující/první výskyt symbolu v search bufferu (místo inicializace počtu výskytu každého symbolu na 1) – pravděpodobnost pro každý symbol abecedy (z počtu kódování symbolu speciálním symbolem)
  - i pro příznak – klesající pravděpodobnost pro symbol
- další příznaky pro pouze vzdálenosti s nejčastější délkou 3 a kódování v kontextu délky 1

Varianty LZ77

LZMA (Igor Pavlov)

-

Další

- LZX
- LZP

LZ77

Implementace

- délky K a L bufferů až tisíce a desítky až stovky symbolů (bytů)
- pro posuvné okno kruhová fronta (LZSS, LZPP), pro search buffer např. suffix stromy (LZR), (vyvážené) binární vyhledávací stromy (s lexikografickým uspořádáním slov v uzlech, LZSS), hešovací tabulky (SLH, Deflate – volitelně tří symbolů, LZPP – CRC-32)
- ARJ, (PK)Arc, LHArc, LHA (LZSS + Huffman), (PK)Zip, gzip, zlib a(Deflate), RAR, ACE (LZSS + Huffman) aj.

Aplikace

- nárůst kvůli poplatkům z patentu na LZW
- v síťových protokolech HTTP, PPP (Deflate)
- v kompresi obrazu PNG (Deflate) a dokumentů PDF (Deflate)

LZ78

- LZ77 předpokládá, že opakující se vzory se vyskytují blízko sebe (do vzdálenosti délky K search bufferu), ale stejné slovo na vstupu může začínat před search bufferem
- také LZ2, Abraham Lempel, Jakob Ziv, 1978 – základ, mnoho variant $\rightarrow$ rodina metod LZ78
- adaptivní slovník $=$ slova na vstupu uložená ve slovníku (nebo prázdné slovo) zřetězená s dalším symbolem na vstupu za slovem, počáteční prázdný

$=$ kódování (i prázdného) slova a dalšího symbolu na vstupu kódy indexu stejného slova ve slovníku (nebo indexu 0) a symbolu, a uložení do slovníku zřetězení slova a symbolu

- vytváření stejného slovníku při kódování i dekódování
- ze slovníku se nemaže – na rozdíl od search bufferu LZ77, vymazání slovníku při jeho naplnění $=$ předpoklad LZ77 (že opakující se vzory se vyskytují blízko sebe – do vzdálenosti odpovídající velikosti slovníku)
- pomalá adaptace na vstup – do slovníku se přidávají slova jen o 1 symbol delší než slovo již ve slovníku

LZ78

$x \leftarrow$ prázdný řetězec, $i \leftarrow 0$;
while načti ze vstupu symbol $a \in A$ **do**
    if xa je ve slovníku **then**
        $i \leftarrow$ index xa ve slovníku (počínaje 1);
        $x \leftarrow xa$;
    else
        ulož xa jako další položku do slovníku;
        zapiš na výstup kódy i a a;
        $x \leftarrow$ prázdný řetězec, $i \leftarrow 0$;
if $x \neq$ prázdný řetězec **then**
    zapiš na výstup kód i;

PRIKLAD

- kódování indexů a symbolů statickým kódem nebo (adaptivním) statistickým kódováním (Huffmanovým nebo aritmetickým)

LZ78

while načti ze vstupu a dekóduj kód indexu i ve slovníku **do**
    $x \leftarrow$ prázdný řetězec;
    if $i \neq 0$ **then**
        zapiš na výstup slovo $x \in A^+$ na indexu i ve slovníku;
    if načti ze vstupu a dekóduj kód symbolu $a \in A$ **then**
        ulož xa jako další položku do slovníku;
        zapiš na výstup symbol $a \in A$;

PRIKLAD

LZ78

Datová reprezentace slovníku $=$ n-ární strom $T = \langle V, E(V) \rangle$ (n velikost A)

- velikost slovníku jednotky až desítky tisíc (2^{16}) symbolů
- trie jako u PPM: $a|c^k \approx$ slovo $a_{-k} \dots a_{-1} a$, pro $a \in A$ s prefixem $x \in A^+$ index slova xa ve slovníku
- kódování:
  - modifikace: pro uzly $v(xa), v(xab), v(xac), v(xad), \dots \in V$ pro symboly a s prefixem x a $b, c, d, \dots$ s prefixem xa hrany $\langle v(xab), v(xac) \rangle, \langle v(xac), v(xad) \rangle, \dots \in E(V)$ místo hran $\langle v(xa), v(xac) \rangle, \langle v(xa), v(xad) \rangle, \dots$
  - tabulka se sloupci index slova xa, symbol a, index slova xab a pro slova $xab, xac, \dots$ index slova $xac, xad, \dots$
- dekódování: tabulka se sloupci index slova xa, symbol a a pro slova xab index slova xa

PRIKLAD

Varianty LZ78

LZFG (Fiala, Greene)

- kombinace LZ77 (search buffer) a LZ78 (kódy dvojic)

= kódování (neprázdného) slova na vstupu kódy délky a pozice stejného slova začínajícího v search bufferu anebo kódy délky slova a všech jeho symbolů

- varianta A1: 4 bity pro binární reprezentaci délky $2 - 16 = L$ slova následované pozicí (vzdáleností) $1 - 4096 = K$ kódovanou do 12 bitů anebo délky $1 - 16$ slova s prefixem $\mathbf{0000}$ následované symboly
- varianta A2: zobecněné unární kódy (start-step-stop kódy) pro délku $2 - 2044$ ($(2, 1, 10)$ kód, $3 - 18$ bitů) následovanou pozicí až $K = 2^{10} + 2^{12} + 2^{14}$ ($(10, 2, 14)$ kód, $11 - 16$ bitů) anebo délku $1 - 63$ ($(0, 1, 5)kód$, $1 - 10$ bitů) následovanou symboly, další vylepšení s postupně narůstajícím K od 21 a pozicí kódovanou do postupně $1 - 16$ bitů ($(10 - d, 2, 14 - d)$ kód, d postupně $10, \ldots, 0$) aj.
- další varianty B1, B2, C1 a C2

Další

- LZRW1
- LZRW4

LZW

= nejpopulárnější varianta LZ78, Terry Welch, 1984

- počáteční slovník (komprese i dekomprese) = všechny symboly vstupní abecedy A

= kódování (neprázdného) slova na vstupu kódem indexu stejného slova ve slovníku, a uložení do slovníku zřetězení slova a dalšího symbolu na vstupu

$x \leftarrow$ prázdný řetězec;
while načti ze vstupu symbol $a \in A$ **do**
    if xa je ve slovníku **then**
        $i \leftarrow$ index xa ve slovníku;
        $x \leftarrow xa$;
    else
        ulož xa jako další položku do slovníku;
        zapiš na výstup kód i;
        $x \leftarrow a$;
if $x \neq$ prázdný řetězec **then**
    zapiš na výstup kód i;

PRIKLAD

LZW

Kódování

1. další položka xa ve slovníku: x slovo ve slovníku na aktuálně kódovaném indexu i, a první symbol slova na dalším kódovaném indexu
2. slovo na dalším kódovaném indexu může být posledně uložená položka ve slovníku

Dekódování

1. další položka xa ve slovníku: x slovo na předchozím dekódovaném indexu, a první symbol slova na aktuálně dekódovaném indexu i
2. slovo na aktuálně dekódovaném indexu i se má právě vložit jako další položka do slovníku $\Rightarrow a =$ první symbol slova x na předchozím dekódovaném indexu

LZW

$x_p \leftarrow$ prázdný řetězec;
while načti ze vstupu a dekóduj kód indexu i ve slovníku **do**
    if i je index další položky ve slovníku **then**
        $x \leftarrow x_p$;
    else
        $x \leftarrow$ slovo na indexu i ve slovníku;
    if $x_p \neq$ prázdný řetězec **then**
        $a \leftarrow$ první symbol x;
        ulož $x_p a$ jako další položku do slovníku;
    $x_p \leftarrow x$;
    zapiš na výstup slovo $x \in A^+$ na indexu i ve slovníku;

PRIKLAD

Varianty LZW

LZC

- postupně zdvojnásobovaná velikost slovníku od 512 (9 bitů/index) do 2^{16}, při naplnění statický a při poklesu kompresního poměru pod mez vymazání
- vylepšení: při kódování indexu slova vyloučení slov ve slovníku začínajících symbolem, kterým slovo pokračuje v jiném slově ve slovníku (exclusion principle)

LZT (Tischer)

- varianta LZC
- při naplnění slovníku vyřazení slova s nejdéle nekódovaným indexem – kromě slovníku i seznam slov ze slovníku setříděný podle počtu kódování indexu slova, podobné LZ77, ale „posuvné okno“ slov podle počtu kódování indexu, ne pořadí slov na vstupu

Varianty LZW

LZMW (Miller, Wegman)

- při naplnění slovníku vyřazení slova s nejdéle nekódovaným indexem – nejstarší ze slov bez pokračování v jiném slově (tzn. index slova nebyl kódován), vyžaduje dat. strukturu slov podle jejich „věku“
- uložení do slovníku zřetězení slova na vstupu, které je ve slovníku, a dalšího slova na vstupu, které je ve slovníku (místo pouze jeho prvního symbolu) $\Rightarrow$ rychlejší adaptace na vstup – do slovníku se přidávají slova o víc než jen o 1 symbol delší
- datová reprezentace slovníku nemůže být trie – ve slovníku nemusí být každý prefix slova $\rightarrow$ dodání s příznakem platnosti, ale při vyhledávání vracení se od neplatných

LZAP (All Prefixes)

- varianta LZMW
- zřetězení nejen s dalším slovem, ale se všemi jeho neprázdnými prefixy (včetně celého slova) $\rightarrow$ větší slovník, ale nevracení se při vyhledávání

Další

- LZJ
- LZY

LZW

Implementace

- pro slovník hešovací tabulka – řádků tabulky pro dekódování ukládající trie LZ78 slovníku
- compress na UNIXu (LZC)

Aplikace

- narůstající až do poplatků z patentu
- v kompresi obrazu GIF – podobně jako compress
- v kompresním módu modemových přenosů dat po telefonní síti podle standardu V.42bis – podobně jako compress, ale při naplnění slovníku jeho promazávání o dlouho nepoužitá slova, nekódování posledně uloženého slova ve slovníku, ale jeho složek