به نام زندگی

آزمون میان ترم: سه‌شنبه ۹,۲۲ ساعت 13:00

کدینگ منبع — Lossless

کدینگ — Shannon – Fano

$$L \geq H(X)$$

$$L = E\{l(x)\} = \sum_x l(x) P(x)$$

$$H(X) = E\left\{\log \frac{1}{P(x)}\right\} = \sum_x \frac{1}{\log P(x)} P(x)$$

$$\Rightarrow \quad l(x) = \left[\log_M \frac{1}{p(x)}\right]^+$$

M-ary Coding

* اگر برای منبع X ، خصوصیات آماری منبع به گونه‌ای باشد که برای تمام الفبای منبع $\log_M \frac{1}{p(x)}$ یک عدد صحیح باشد ، آن منبع را یک منبع M-adic می‌گوییم در این صورت کدینگ Shannon-Fano یک کد بهینه به دست می‌دهد که در آن $L = H(X)$

$$\forall x \in X, \quad \exists n \in \mathbb{N} \; ; \; \log_m \frac{1}{p(x)} = n$$

$$\Rightarrow \frac{1}{p(x)} = m^{n}, \qquad p(x) = m^{-n}$$

در غیر این صورت که Shannon-Fano حداکثر به میزان 1 واحد از مقدار
بهینه فاصله دارد. زیرا می‌دانیم که

$$l(x) = \left\lceil \log_m \frac{1}{p(x)} \right\rceil^{+}$$

$$\Rightarrow \quad \log_M \frac{1}{p(x)} \leq \ell(x) < 1 + \log_M \frac{1}{p(x)}$$

$$\xrightarrow[\text{ضرب می‌کنیم}]{\text{طرفین را در } P(x)} \quad p(x) \log_M \frac{1}{p(x)} \leq p(x)\,\ell(x) < p(x) + p(x) \log_M \frac{1}{p(x)}$$

$$\xRightarrow{\sum_x} \quad \underbrace{\sum_x p(x) \log_M \frac{1}{p(x)}}_{H_M(X)} \leq \underbrace{\sum_x p(x)\,\ell(x)}_{L} < \underbrace{\sum_x p(x)}_{1} + \underbrace{\sum_x p(x) \log_M \frac{1}{p(x)}}_{H_M(X)}$$

Shannon
$\Rightarrow$
Fano

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

$\Leftarrow$ کد شنون shannon-fano حداکثر به اندازه‌ی یک واحد از مقدار بهینه $H_M(X)$
فاصله دارد و یا به عبارت دیگر این که، حداکثر 1 واحد سربار (overhead)
در اطلاعات ارسالی باقی می‌گذارد. (کد shannon-fano زیر بهینه است)

$\Leftarrow$ برای طراحی کد shannon-fano با توجه به مشخص بودن طول کلمات کد و
با توجه به اینکه هیچ کلمه‌ی کدی نباید پیشوند کلمه کد دیگری باشد، با استفاده از

یک ساختار درختی می توانیم کلمات کد را برای هر یک از الفبای منبع به دست بیاوریم. این موضوع را با یک مثال بررسی می کنیم.

مثال ۱ - برای یک منبع X با مشخصات زیر، کد Shannon-Fano را به دست بیاورید (کدیابی)

x	$P(x)$
1	$1/2$
2	$1/3$
3	$1/12$
4	$1/12$

$$H(X) = \sum_x p(x) \log \frac{1}{p(x)}$$

$$= \frac{1}{2}\underbrace{\log 2}_{1} + \frac{1}{3}\underbrace{\log 3}_{1.585} + 2 \times \frac{1}{12}\overbrace{\log 12}^{3.585} = 1.626$$

x	$P(x)$	$l(x) = \lceil \log \frac{1}{p(x)} \rceil$	$c(x)$
1	$1/2$	1	0
2	$1/3$	2	10
3	$1/12$	4	1100
4	$1/12$	4	1110

← کد Shannon-Fano یک کد زیربهینه است.

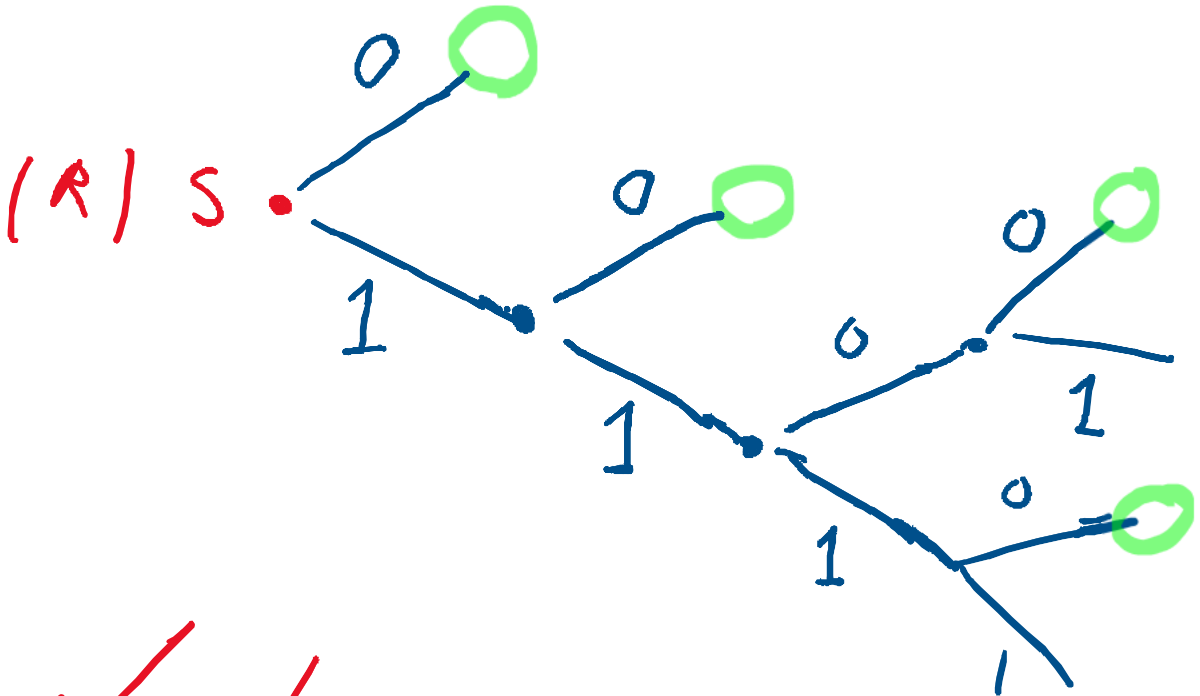
$$\Rightarrow L = \sum_{x} l(x)\, p(x) = \frac{1}{2} \times 1 + \frac{1}{3} \times 2 + 2 \times \frac{1}{12} \times 4$$

$$\Rightarrow L = \frac{1}{2} + \frac{4}{3} = \frac{11}{6} = 1.833$$

$$, \quad H(X) = 1.626$$

(بیشتر از 0.2 از مقدار بهینه فاصله دارد) $1.833 = L > H(X) = 1.626$

برای هر کلمه کد یک برگ قرار می‌دهیم. از برگ، دیگر شاخه‌ای منشعب نمی‌شود.



همان طور که می‌بینیم بر اساس درخت، کلمات کد 110 و 111 کلمات کد مناسب‌تری
برای $x=3$ و $x=4$ هستند که طول کمتری دارند و لحظه‌ای بودن را نیز برآورده می‌کنند.

* کدینگ منبع Huffmann

در روش کدینگ هافمن با استفاده از یک ساختار درختی خاص، کلمات کدی برای الفبای منبع انتخاب می‌شوند که در آنها میانگین طول کلمات کد تا حد ممکن به حد آنتروپی نزدیک است و هیچ کلمه کدی، پیشوند کلمه‌ی کد دیگری نیست.

به این ترتیب اشکالی که در مورد کدینگ Shannon-Fano در مورد طول کلمات کد، دیدیم، در اینجا برطرف می‌شود.

الگوریتم کدینگ هافمن را با کمک یک مثال، توضیح می دهیم.

مثال ۲: یک منبع با مشخصات زیر در اختیار داریم. می خواهیم کد هافمن را برای آن پیدا کنیم (باینری)

x	$p(x)$					$c(x)$
1	0.25	0.3	0.45	0.55 ^{0}	1	01
2	0.25	0.25	0.3 ^{0}	0.45 ^{1}		10
3	0.2	0.25 ^{0}	0.25 ^{1}			11
4	0.2 ^{0}	0.2 ^{1}				000
5	0.1 ^{1}					001

۱- الفبای منبع را به ترتیب نزولی احتمال در یک جدول قرار می دهیم.

2- برای به دست آوردن کد های M-ary ، M الفبای با کمترین احتمال ها را با هم ترکیب می کنیم و یک الفبای جدید به دست می آوریم که احتمال آن برابر حاصل جمع احتمال های M الفبای انتخاب شده است. در ستون بعدی، دوباره الفبا ها را به ترتیب نزولی احتمال مرتب می کنیم. مسیر هر الفبا از یک ستون به ستون دیگر را با شاخه های یک درخت نمایش می دهیم. بر روی شاخه های الفباهای ترکیب شده عددهای ۰ تا M-1 را قرار می دهیم.

3 - مرحله 2 را آنقدر تکرار می‌کنیم تا به یک الفبا با احتمال 1 می‌رسیم.

4 - برای پیدا کردن کلمه کد مربوط به الفبای x ، روی شاخه‌های درخت از الفبای x تا انتها حرکت می‌کنیم و سمبل‌های دیده شده روی شاخه‌های درخت را به صورت MSB First به عنوان کلمه‌ی کد مربوط به x در نظر می‌گیریم.

$$H(X) = \sum_{x} p(x) \log \frac{1}{p(x)} = 2 \times \frac{1}{4} \overbrace{\log 4}^{2} + 2 \times 0.2 \overbrace{\log 5}^{2.322} + 0.1 \underbrace{\log 10}_{3.322} = 2.263$$

$$L = \sum_{x} \ell(x) p(x) = 0.25 \times 2 \times 2 + 0.2 \times 2 + 0.2 \times 3 + 0.1 \times 3 = 2.300$$

$$\Rightarrow \quad 2.300 = L > H(X) = 2.263$$

(به مقدار 0.037 از
مقدار بهینه فاصله دارد)

تمرین: برای منبع مثال 2 ، کد Shannon - Fano را به دست بیاورید و
عملکرد آن را از نظر نزدیکی به آنتروپی با کد هافمن، مقایسه کنید.

مثال 3 - کدهافمن را برای منبع مثال ۱، به دست بیاورید و عملکرد آن را با کد
Shannon-Fano مقایسه کنید (اجباری)

x	$P(x)$
1	$\frac{1}{2}$
2	$1/3$
3	$1/12$
4	$1/12$

$$L_{S-F} = 1.833$$

$$H(X) = 1.626$$

به اندازه 0.207 با مقدار بهینه اختلاف دارد

x	$p(x)$				$C(x)$
1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$ 0	1	0
2	$\frac{1}{3}$	$\frac{1}{3}$ 0	$\frac{1}{2}$ 1		10
3	$\frac{1}{12}$ 0	$\frac{1}{6}$ 1			110
4	$\frac{1}{12}$ 1				111

$$1.666 = L_H > H(X) = 1.626$$

به اندازه ی 0.04 فاصلت با
مقدار بهینه

$$L = \sum_x l(x)p(x) = \frac{1}{2} \times 1 + \frac{1}{3} \times 2 + 2 \times \frac{1}{12} \times 3 = 1.666$$

مثال ۴: برای منبع مثال ۲، کدهافمن ternary را بدست بیاورید

M-ary $\to M^* = \{0, \dots, M-1\}$ ، $M^* = \{0, 1, 2\}$

x	$P(x)$			$c(x)$
1	0.25	0.5^{0}	1	1
2	0.25	0.25^{1}		2
3	0.2^{0}	0.25^{2}		00
4	0.2^{1}			01
5	0.1^{2}			02

تمرین: برای مثال ۳، آنتروپی و L را محاسبه کنید و عملکرد کد را ارزیابی کنید.

$$H_3(X) = \sum_x p(x) \log_3 \frac{1}{p(x)}$$

$$L = \sum_x \ell(x)\, p(x)$$

• در مورد کدهای هافمن، توجه به چند نکته لازم است.

نکته اول اینکه در مراحل الگوریتم هافمن، ممکن است وقتی می‌خواهیم سمبل‌ها
را به ترتیب نزولی احتمال مرتب کنیم، بعضی سمبل‌ها هم احتمال باشند. در این
صورت ترتیب قرارگیری این سمبل‌ها در ستون‌های نمودار درختی، ما را به
جواب‌های مختلفی برای کتاب کد (یا دیکشنری کدها) می‌رساند اما بر روی
عملکرد کد، تاثیری ندارد. (یعنی L را تغییر نمی‌دهد)

این موضوع در مرحله‌ی آخر مثال 3، و مرحله‌ی اول مثال 2 نیز دیده می‌شود.

مثال 5 : برای منبع زیر کدهافمن باینری را بدست بیاورید:

x	$P(x)$					$C(x)$
1	0.3	0.3	0.4	0.6	1	00
2	0.3	0.3	0.3	0.4		01
3	0.2	0.2	0.3			10
4	0.1	0.2				110
5	0.1					111

$$L = \sum_x l(x)\,p(x) = 2(0.3+0.3+0.2) + 3(0.1+0.1)$$

$$L = 2.2$$

$$H(X) = \sum_x p(x) \log \frac{1}{p(x)} = 2\times 0.3 \log \frac{10}{3} + 0.2 \log 5 + 2\times 0.1 \log 10 = \dots$$

تمرین : در جدول الگوریتم هافمن در قسمتی که احتمال های مساوی ظاهر شده است ، ترتیب چینش را عوض کنید و کد حاصل را با کد شکل ۵ مقایسه کنید.

نکته ی دوم : در روند الگوریتم هافمن برای حالت $M > 2$ ، پس از ترکیب
سمبل های با کمترین احتمال، ممکن است در مرحله ی آخر به این حالت برسیم که تعداد
الفبای باقی مانده، کمتر از M است و درخت ناقص می شود و نمی توان کد بهینه
را به دست آورد. در این حالت در مرحله ی اول به تعداد لازم، الفبای Dummy
با احتمال '0' به مجموعه ی الفبای منبع اضافه می کنیم به طوری که در مرحله ی آخر
درخت کد، کامل باشد و به کد بهینه برسیم.

با توجه به اینکه در هر مرحله از الگوریتم کدینگ $M-1$ الفبا از کل تعداد الفبای منبع کم می‌شود تا مرحله آخر که به یک الفبا، با احتمال 1 می‌رسیم، بنابراین اگر شرط زیر برقرار باشد، درخت کدینگ کامل است

$$|X| = k\underbrace{(M-1)}_{\text{تعدادی که در هر مرحله کم می‌شود}} + \underbrace{1}_{\text{الفبای باقیمانده در مرحله‌ی آخر}}$$

اگر این رابطه برای منبع برقرار نباشد، در مرحله‌ی اول به تعدادی که این رابطه برقرار شود، الفبای Dummy با احتمال '0' اضافه می‌کنیم.

مثال 6: برای منبع زیر، کد هافمن ternary را بدست بیاورید

x	$p(x)$				$c(x)$
1	0.25	0.25	0.5	1	1
2	0.25	0.25	0.25		2
3	0.2	0.2	0.25		00
4	0.1	0.2			02
5	0.1	0.1			010
6	0.1				011
7	0				

$|X| = 6$

$M = 3$

$k(M-1)+1$

$2k+1 \neq |X|$

یک الفبای Dummy