信息论与编码

马啸
maxiao@mail.sysu.edu.cn

计算机学院
中山大学

2021 年春季学期

1. 循环码
   - 循环码的数学描述
   - 循环码的译码

2. 卷积码
   - 多项式环与形式幂级数
   - 卷积码
   - 编码器
   - 卷积码图表示与译码算法
   - 译码：维特比算法

循环码的数学描述

Definition 1

一个(n,k) 线性码C, 若对任意$c=(c_0,c_1,\cdots,c_{n-1})\in C$, 恒有$c'=(c_{n-1},c_0,\cdots,\mathbf{c}_{n-2})\in C$, 称$C$ 为循环码。

循环码的码字可以用向量表示之外,还可以用x 的多项式表示为

$$c(x)=c_0+c_1x+\cdots+c_{n-1}x^{n-1}$$

称$c(x)$ 为码字多项式。若以x 乘以$c(x)$,并用多项式x^n-1 去除可以得到

$$\begin{aligned}xc(x)&=\left(c_0x+c_1x^2+\cdots+c_{n-1}x^n\right)\bmod\left(x^n-1\right)\\&=c_{n-1}+c_0x+\cdots+c_{n-2}x^{n-1}\end{aligned}$$

这样,循环码的循环移位可由模x^n-1 下的码字多项式$c(x)$ 乘以x 运算给出,循环码的研究可以利用模x^n-1 的多项式代数R_n 进行。R_n 是$GF(q)$ 上低于n 次的所有q^n 个多项式的集合。

循环码的数学描述

Theorem 2

若$c(x)$ 为一个循环码多项式, $b(x) \in R_n$, 则$b(x)c(x) \bmod x^n - 1$ 也是一个码多项式。

证明： 令$b(x) = b_0 + b_1 x + \cdots + b_{n-1}x^{n-1}$, 由于循环码的定义可知, $xc(x), x^2c(x), \cdots, x^{n-1}c(x)$ 都是码多项式。又因为循环码是线性码,这些码字的线性组合还是循环码的一个码字, 因此$b_0c(x) + b_1xc(x) + \cdots + b_{n-1}x^{n-1}c(x)$ 是一个码多项式，证毕。

定理说明码多项式的倍式仍是一个码多项式。

循环码的数学描述

在一个循环码当中,存在一个次数最低的非零多项式$g(x)$,该多项式的最高次项系数为1，称为首一多项式。对$g(x)$ 有下述定理。

Theorem 3

循环码中的所有码多项式$c(x)$ 都是$g(x)$ 的倍式, $g(x)$ 称为循环码的生成多项式而且是唯一的。

证明： 设$g(x)$ 是循环码C 中次数最低的首一多项式,对任意的$c(x) \in C$, 有$c(x) = q(x)\ g(x) + r(x), r(x)$ 是$c(x)$ 除$g(x)$ 的余式, 所以, $r(x)$ 次数小于$g(x)$ 的次数。由于$q(x)g(x) \in C$,有$r(x) = c(x) - q(x)g(x) \in C$, 与$g(x)$ 是循环码C 中次数最低的首一多项式假设矛盾,所以$r(x) = 0$, 即所有码多项式都是$g(x)$ 倍式。
设$h(x)$ 和$g(x)$ 都是C 中次数最低的首一多项式,则由$h(x) - g(x) \in C$ 知,C 中有次数更低的多项式存在,与假定矛盾。因此C 中次数最低的首一多项式是唯一的，证毕。

循环码的数学描述

Theorem 4

若C 是R_n 中的循环码,则C 的最低次首一生成多项式$g(x)$ 是$x^n - 1$ 的因式。

证明： 令$g(x)$ 是C 的最低次首一多项式,则有$x^n - 1 = q(x)g(x) + r(x)$, 其中$r(x)$ 的次数低于$g(x)$ 的次数。而$r(x) = q(x)g(x) \bmod x^n - 1$, 所以$r(x)$ 在C 中。由于$g(x)$ 是C 中次数最低的首一多项式，所以必有$r(x) = 0$, 因此$g(x)$ 是$x^n - 1$ 的因式。

循环码的数学描述

由上述几个定理知道,构造循环码在于从分解x^n-1 的分解中取出一个因式$g(x)$, 以它作为生成多项式就可构成一个循环码。x^n-1 的分解可以查表[Peterson(1961)]。

给定一个$n-k$ 次首一多项式$g(x)$, ,则$g(x), xg(x), x^2g(x), \cdots, x^{k-1}g(x)$的系数矢量是线性独立的,由它们可以构成码的生成矩阵

$$\mathbf{G}=\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 \\ \vdots & & \ddots & & & & \ddots & \vdots \\ 0 & 0 & \cdots & g_0 & & \cdots & & g_{n-k} \end{bmatrix}$$

循环码的数学描述

如果将待编码的消息序列$\mathbf{u} = (u_0, u_1, \cdots, u_{k-1})$ 表示为多项式

$$u(x) = u_0 + u_1 x + \cdots + u_{k-1} x^{k-1}$$

则以x^{n-k} 乘以$u(x)$,再以$g(x)$ 除,得

$$x^{n-k} u(x) = q(x) g(x) + r(x)$$

其中$q(x)$ 是商式。$r(x) = r_0 + r_1 x + \cdots + r_{n-k-1} x^{n-k-1}$ 是余式。次数低于$g(x)$ 的次数。重新排列上式,得

$$c(x) = x^{n-k} u(x) - r(x) = q(x) g(x) \in C$$

其中$c(x)$ 的系数为$(-r_0, -r_1, \cdots, -r_{n-k-1}, u_0, u_1, \cdots, u_{k-1})$ 正是系统码形式的码字。由此得出系统循环码的编码步骤如下:

1. 以x^{n-k} 乘以$u(x)$ 。
2. 以$g(x)$ 除$x^{n-k} u(x)$ 得余式$r(x)$ 。
3. 组合$r(x)$ 和$x^{n-k} u(x)$ 得到码字$[-r(x), u(x)]$。

循环码的数学描述

Example 5

$q = 2$ 时选$g(x) = 1 + x + x^3$, 它除尽$1 - x^7$ 。若$u(x) = 1 + x^3$, 则由

$$x^3\left(1 + x^3\right) = x^3 + x^6 = \left(x + x^2\right) \bmod g(x)$$

得出码多项式$c(x) = x + x^2 + x^3 + x^6$, 即码字$c = (0111001)$。

循环码的数学描述

设 $x^n - 1 = g(x)h(x)$, $g(x) = \sum_{i=0}^{n-k} g_i x^i$, $h(x) = \sum_{i=0}^{k} h_i x^i$. 由 $g(x)$ 生成的循环码的生成矩阵是

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 \\ & \ddots & \ddots & & & \ddots & \vdots \\ 0 & \cdots & 0 & g_0 & & \cdots & g_{n-k} \end{bmatrix}$$

循环码的数学描述

其校验矩阵可以按如下方式推导

Theorem 6

由 $x^n - 1 = g(x)h(x)$, 我们可以得到：

$$\begin{aligned} &g_0 h_0 = 1 \\ &\sum_{l=0}^{n-k} g_l h_{i-l} = 0, for 1 \leqslant i \leqslant n-1 \\ &g_{n-k} h_k = 1 \end{aligned}$$

$$\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & & \cdots & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & \cdots & 0 \\ & \ddots & \ddots & & & \ddots & \vdots \\ 0 & \cdots & 0 & h_k & & \cdots & h_0 \end{bmatrix}$$

G 的第一行与 H 的各行的“内积”依次是 $x^k, x^{k+1}, \cdots, x^{n-1}$ 的系数。

循环码的数学描述

Example 7

由$x^7-1=(1+x)\left(1+x+x^3\right)\left(1+x^2+x^3\right)$, 例7 选$g(x)=1+x+x^3$,
则$h(x)=(1+x)\left(1+x^2+x^3\right)=1+x+x^2+x^4$, 即

$$h^*(x)=x^4\left(1+x^{-1}+x^{-2}+x^{-4}\right)=1+x^2+x^3+x^4$$

由此得

$$\mathbf{H}=\begin{bmatrix}1&0&1&1&1&0&0\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix}$$

由式$u(x)=u_0+u_1x+\cdots+u_{k-1}x^{k-1}$, 当给定$\mathbf{u}=(u_0,u_1,\cdots,u_{k-1})$ 时,按多项式可计算

$$x^{n-k}u(x)=u_0x^{n-k}+u_1x^{n-k+1}+\cdots+u_{k-1}x^{n-1}$$

其系数矢量和$\mathbf{H}$ 矩阵相应的最后一行的内积为0 可算出码的相应位c_{n-k-1}, 即

$$c_{n-k-1}h_k+u_0h_{k-1}+\cdots+u_{k-1}h_0=0$$

因为$h^*(x)$ 是首一多项式,所以$h_k=1$ 。因此

$$c_{n-k-1}=u_0h_{k-1}+\cdots+u_{k-1}h_{00}$$

然后将已知的$k+1$ 个高次项系数组成矢量与$\mathbf{H}$ 矩阵的倒数第二行进行内积运算,令其结果为0 可以算出c_{n-k-2}。依此类推就可以得到系统码的码字。

常见通信标准中的CRC

- 以太网–CRC32 **[IEEE 802.3 from Wikipedia]**:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

- 5G NR **[3GPP TS 38.212 V15.8.0 (2019-12)]**:

5.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, \ldots, a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, \ldots, p_{L-1}$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{\text{CRC24A}}(D)=[D^{24}+D^{23}+D^{18}+D^{17}+D^{14}+D^{11}+D^{10}+D^{7}+D^{6}+D^{5}+D^{4}+D^{3}+D+1]$ for a CRC length $L=24$;
- $g_{\text{CRC24B}}(D)=[D^{24}+D^{23}+D^{6}+D^{5}+D+1]$ for a CRC length $L=24$;
- $g_{\text{CRC24C}}(D)=[D^{24}+D^{23}+D^{21}+D^{20}+D^{17}+D^{15}+D^{13}+D^{12}+D^{8}+D^{4}+D^{2}+D+1]$ for a CRC length $L=24$;
- $g_{\text{CRC16}}(D)=[D^{16}+D^{12}+D^{5}+1]$ for a CRC length $L=16$;
- $g_{\text{CRC11}}(D)=[D^{11}+D^{10}+D^{9}+D^{5}+1]$ for a CRC length $L=11$;
- $g_{\text{CRC6}}(D)=[D^{6}+D^{5}+1]$ for a CRC length $L=6$.

常见通信标准中的CRC

- LTE(4G+) **[3GPP TS 36.212 V10.9.0 (2015-09)]:**
  - 5.1.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$. A is the size of the input seque and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ and;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ for a CRC length $L = 24$ and;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1]$ for a CRC length $L = 16$.
- $g_{CRC8}(D) = [D^8 + D^7 + D^4 + D^3 + D + 1]$ for a CRC length of $L = 8$.

循环码的译码

循环码译码步骤和线性码一样,也是先计算接收矢量的伴随式, 然后根据它判断是否有错，当有错时进而判断错误图样并进行纠正。由于循环码的循环结构使得它的译码可以较线性码简单而易于实现。
设接收矢量$v = (v_0, v_1, \cdots, v_{n-1}) \in V_n(q)$, 相应多项式表示为$v(x) = v_0 + v_1 x + \cdots + v_{n-1}x^{n-1}$, 以$g(x)$ 除得

$$v(x) = q(x)g(x) + s(x) = s(x) \bmod g(x)$$

称低于$n-k$ 次的余式$s(x) = s_0 + s_1 x + \cdots + s_{n-k-1}x^{n-k-1}$ 为接收多项式$v(x)$ 的伴随多项式。若$s(x) = 0 \bmod g(x)$,就认为无错(可能含有不可检错误图样)。若$s(x) \neq 0 \bmod g(x)$, 则$v(x)$ 中必有错误存在,如果将错误图样表示成多项式$e(x) = e_0 + e_1 x + \cdots + e_{n-1}x^{n-1}$, 则接收多项式$v(x) = c(x) + e(x)$,相应的伴随多项式为$s(x) = v(x) = c(x) + e(x) = e(x) \bmod g(x)$,即它只与信道错误图样$e(x)$ 有关。

循环码的译码

Theorem 8

若$s(x)$ 是$v(x)$ 的伴随多项式,则$s^{(1)}(x) = xs(x) \bmod g(x)$ 是$v^{(1)}(x) = xv(x) \bmod (x^n - 1)$ 的伴随多项式。

证明： 以x 乘$v(x) = q(x)g(x) + s(x)$ 的两边,并以$g(x)$ 为模,则左边为$s^{(1)}(x)$, 右边即为

$$xs(x) \bmod g(x).$$

卷积码

多项式环与形式幂级数

粗略地讲，环是较域更弱的一个结构体，可以做“加、减、乘”三种运算，但未必可以做除法。换言之，域是环的一个特殊例子。我们常见的环有所有整数构成的集合 $\mathbb{Z}$。在 $\mathbb{Z}$ 中，我们可以定义 $x+y, x-y, x\cdot y$，但一般情况下没有办法定义除法，或者说没有合适的方式定义一个数 x 的乘法逆。也就是说，对于一般的整数 $x \neq 0$，我们找不到整数 y，使 $x\cdot y = 1$。整数环 $(\mathbb{Z}, +, \cdot)$ 是一类特殊的环，具有唯一分解性：任何一个非零整数可以唯一地分解成素数的连乘积。

在编码领域用得较多的是多项式环与形式幂级数环。我们讨论如下。

多项式环与形式幂级数

设 $\mathbb{F}$ 是一个域，形如 $f(D) = f_0 + f_1 D + \cdots + f_m D^m$ 的式子称为多项式，其中 $f_i (0 \le i \le m) \in \mathbb{F}$ 称为系数，而 D 是“哑元”，在编码中暗示延迟（delay）算子。若 $f(D) = \sum_{i=0}^{m} f_i D^i$ 中 $f_m \neq 0$，我们称 f_m 为最高项系数或者首项系数，定义 $f(D)$ 的次数是 m，记为 $\deg(f)$。

设 $f(D)$ 与 $g(D)$ 是两个多项式，则可以定义加法 $f(D) + g(D)$ 与乘法 $f(D) \cdot g(D)$。这些定义我们应该不陌生。

$$f(D) + g(D) = \sum_{i=0}^{\max(\deg(f), \deg(g))} (f_i + g_i) D^i$$

$$f(D) \cdot g(D) = \sum_i h_i D^i, \text{ 其中 } h_i = \sum_\ell f_\ell g_{i-\ell} = \sum_\ell g_\ell f_{i-\ell}.$$

多项式环与形式幂级数

可以验证，两个多项式相加，次数不超过最大的那个（还有可能减小），而两个非零多项式相乘，次数是 $\deg(f)+\deg(g)$。我们约定零多项式的次数是 $-\infty$。

可以看出，次数不超过 m 的多项式 $f(D)$ 与有限长序列 $f=(f_0,f_1,\cdots,f_m)$ 是一一对应的。因而，我们可以把二者等同起来，只不过用多项式记号使我们更方便地定义加法与乘法。可以验证，域 $\mathbb{F}$ 上所有的多项式（记为 $\mathbb{F}[D]$）在上述定义的加法与乘法意义下构成一个环。

多项式环与形式幂级数

形式上，我们也可以把无穷序列 $a = (a_0, a_1, \cdots)$ 与无穷幂级数 $a(D) = \sum_{i=0}^{\infty} a_i D^i$ 对应起来，称为形式幂级数。注意这只是个形式，不要求其具有“收敛”性质。域 $\mathbb{F}$ 上所有的幂级数的全体记作 $\mathbb{F}[[D]]$。对于两个形式幂级数 $a(D)$ 与 $b(D)$，我们可以形式地定义 $a(D) + b(D)$ 与 $a(D) \cdot b(D)$，定义方式与多项式运算的定义一致。同样可以验证，$\mathbb{F}[[D]]$ 是一个环。

多项式环与形式幂级数

与整数环 $\mathbb{Z}$ 类似，$\mathbb{F}[[D]]$ 可以扩展成一个域，即 Laurent（洛伦）级数全体，其中元素具有 $x(D)=\sum_{i=r}^{\infty} x_i D^i$ 的形式，r 是一个整数。也就是说，一个洛伦级数是包含至多有限项负幂次的幂级数。既然是域，就可以定义乘法逆或者除法。这个除法的定义是按照长除法来定义的。我们按如下规则描述长除法：设 $f(D)$ 与 $g(D)$ 是两个洛伦级数（注意，多项式、幂级数都可以看作是洛伦级数的特殊情况），其中 $g(D)\neq 0$，我们可以把 $g(D)$ 写作 $D^{-r}g_0(D)$ 的形式，$g_0(D)$ 中无负幂次，这样 $\frac{f(D)}{g(D)}=\frac{D^r f(D)}{g_0(D)}$。要计算 $\frac{f(D)}{g_0(D)}$，我们把分子、分母均按照升次排序，然后试商也是按照升次排序。

多项式环与形式幂级数

考虑如下 $\mathbb{F}_2((D))$ 中的例子，注意系数的运算按照 mod 2 进行。

Example 9

$f(D) = 1 + D, g(D) = 1 + D + D^2$，求 $\frac{f(D)}{g(D)}$

解：按照长除法

$$
\begin{array}{r|l}
1 + D + D^2 & 1 + D^2 + D^3 + D^4 + D^5 \cdots \\
\hline
1 + D & \\
1 + D + D^2 & \\
\hline
D^2 & \\
D^2 + D^3 + D^4 & \\
\hline
D^3 + D^4 & \\
D^3 + D^4 + D^5 & \\
\hline
D^5 &
\end{array}
$$

依此类推，可以求出各项系数。当然，这是理解定义的方式，在实践中，或有些情况下是有闭式解的。

卷积码

给定一个域 $\mathbb{F}$，我们定义了多项式环 $\mathbb{F}[D]$，进而可以定义有理函数域 $\mathbb{F}(D) = \left\{ \frac{f(D)}{g(D)} | f(D) \text{ 与 } g(D) \text{ 是多项式，且 } g(D) \neq 0 \right\}$。我们也可以定义形式幂级数环 $\mathbb{F}[[D]]$，进而定义洛伦级数域 $\mathbb{F}((D))$。

从一般码的角度，我们可以定义 $\mathbb{F}((D))$ 上的分组码，其中一个码字具有形式 $c(D) = (c_0(D), c_1(D), \cdots, c_{n-1}(D))$。与常见的分组码相比，这里的分量（不失一般性）可以设为形式幂级数，代表的是序列，因而码长 n 通常比较小。

编码器

一般而言，卷积编码器接受 k 路序列，
即 $\underline{u}(D) = (u_0(D), u_1(D), \cdots, u_{k-1}(D))$，输出 n 路序列，
即 $\underline{c}(D) = (c_0(D), c_1(D), \cdots, c_{n-1}(D))$，而输入—输出的关系可以通过一个生成矩阵 $G(D)$ 来关联，即 $c(D) = u(D)G(D)$。生成矩阵 $G(D)$ 有如下形式 $G(D) = (g_{i,j}(D)), 0 \leq i \leq k-1, 0 \leq j \leq n-1$，其中 $g_{i,j}(D)$ 是有理函数的形式，即两个多项式相除的形式 $\frac{f(D)}{q(D)}$。前面已经提到，传统卷积码的 k, n 通常比较小。

编码器

Example 10

考虑一个码率为 $\frac{1}{2}$ 的卷积码，其生成矩阵 $G(D) = (1 + D + D^2,\ 1 + D^2)$。我们知道，生成矩阵作为码字空间的一组基，可以做初等变换，得到的码字空间不变，但是信息序列与码字序列之间的对应关系会有变化。例如，$G(D)$ 可以等价变换为 $\tilde{G}(D) = \frac{1}{1+D+D^2} G(D) = (1,\ \frac{1+D^2}{1+D+D^2})$。从多项式环、有理分式域等的角度描述卷积码，提供了一些代数工具，可以研究卷积码的一些结构。在这里，我们这些描述主要是表明一个观点，卷积码也是一类“分组码”，只不过码字符号是形式级数罢了。这些描述有点儿抽象，更多时候是利用移位寄存器等直接描述编码。例如，对应 $G(D)$ 与 $\tilde{G}(D)$ 的编码器结构如图所示。

编码器

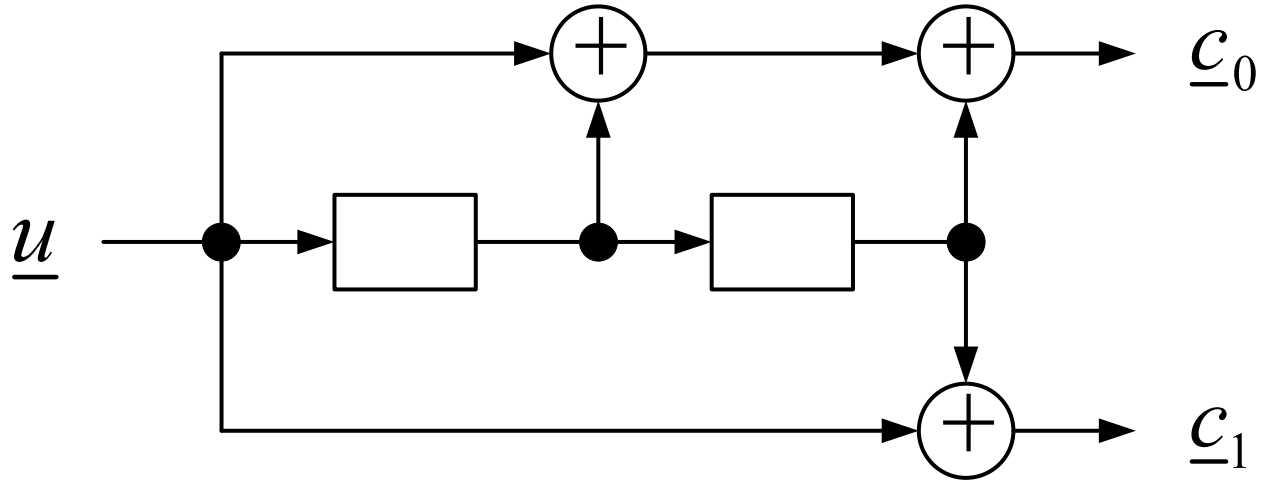


Figure: 非系统编码器 $G(D) = (1 + D + D^2,\ 1 + D^2)$

编码器

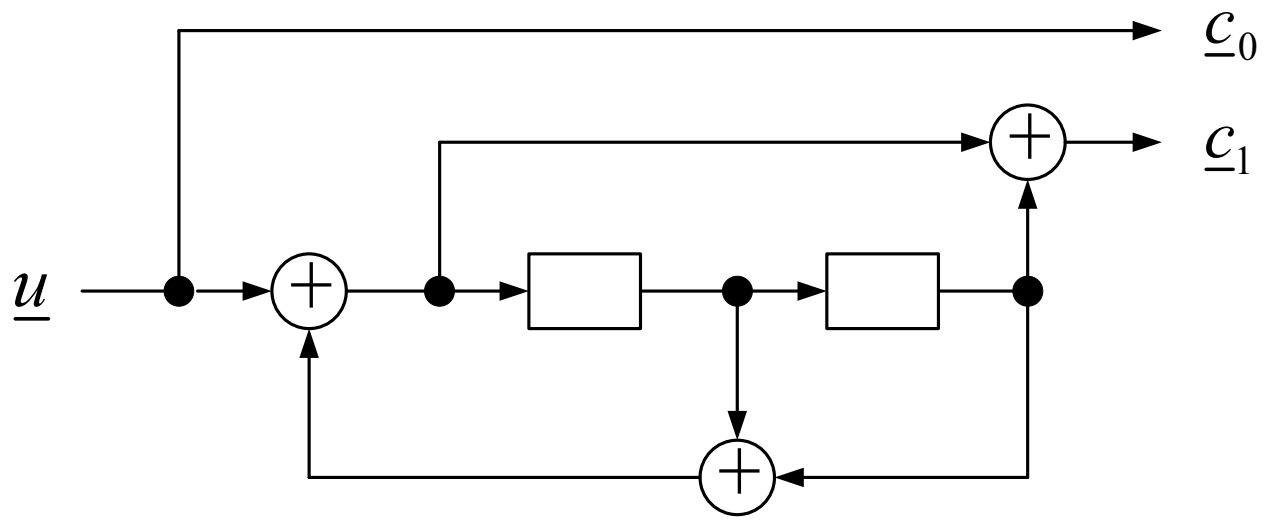


Figure: 递归系统编码器 $\tilde{G}(D) = (1, \frac{1+D^2}{1+D+D^2})$

编码器

对于 k 输入，n 输出的卷积码，我们需要 k 组寄存器，每组寄存器的个数是由生成矩阵的相应多项式的次数确定的。在给出更多例子之前，我们先给出一般的乘法与除法的实现框图，

设 $f(D) = f_0 + f_1 D + \cdots + f_n D^n$，则一个输入序列 $u(D)$ 乘以 $f(D)$ 的输出可以按照下图实现。

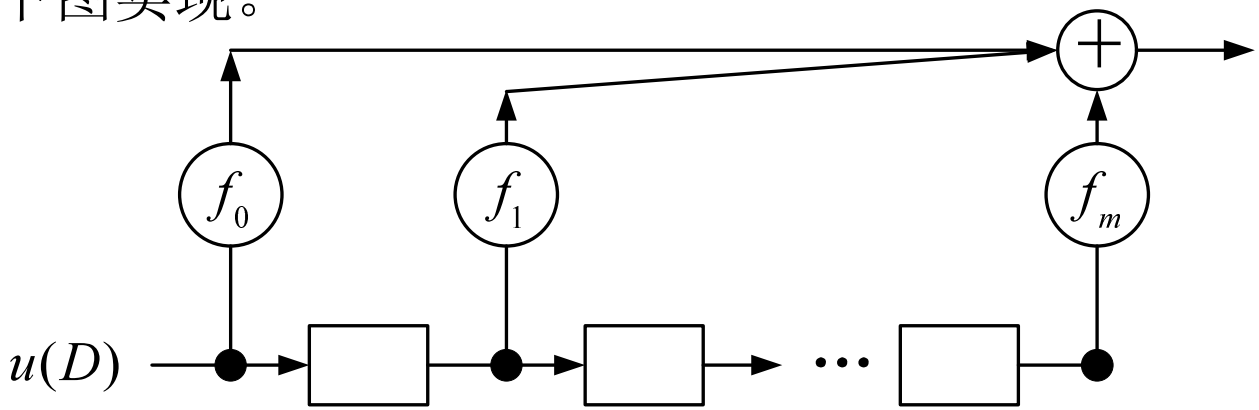


Figure: 多项式乘法器，$v(D) = u(D)f(D)$

注意，$u(D)$ 输入的顺序是 $u_0, u_1, \cdots, u_t, \cdots$。在时刻 t，寄存器中的内容是 $u_{t-1}, u_{t-2}, \cdots, u_{t-m}$。特别地，我们假定 u_t 在 $t < 0$ 时是 0，这样，在 t 时刻的输出就表示为

$$v_t = f_0 u_t + f_1 u_{t-1} + \cdots + f_m u_{t-m},$$

与乘法定义中 D^t 的系数的定义一致。

编码器

乘法器的实现容易理解，而除法可以看作乘法的逆，但要求除式 $f(D)$ 的常数项必须为 1，即 $f(D) = 1 + f_1 D + f_2 D^2 + \cdots + f_m D^m$。从 $v(D) = u(D)f(D)$ 可以看出，$u(D)$ 在 t 时刻的系数 u_t 可以表示为 $u_t = v_t - f_1 u_{t-1} - f_2 u_{t-2} - \cdots - f_m u_{t-m}$，因此，除法器可以利用反馈实现。

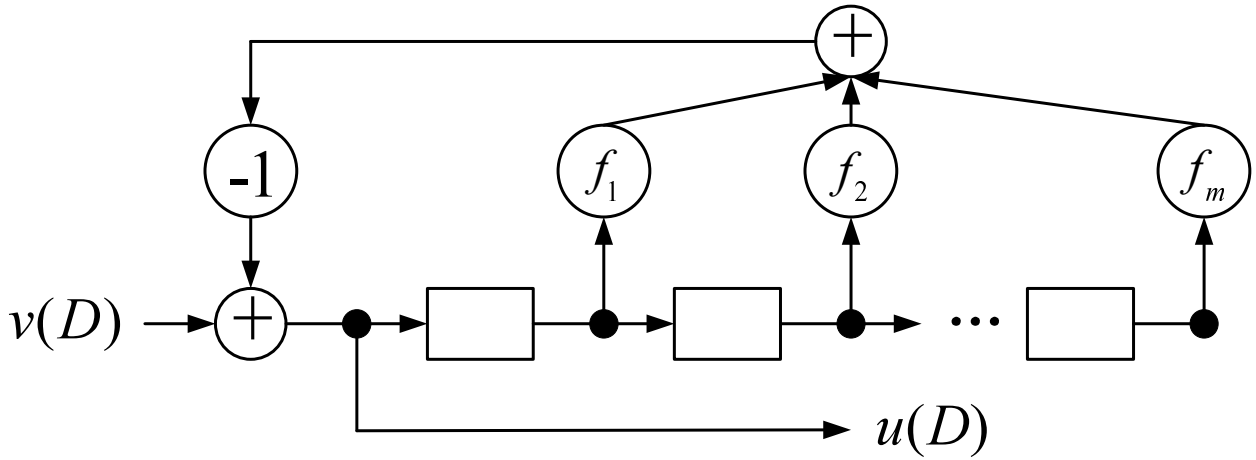


Figure: 多项式除法器，$u(D) = \frac{v(D)}{f(D)}$

编码器

Example 11

我们考虑一个码率为 $\frac{2}{3}$ 的卷积码，生成矩阵 $\mathbf{G} = \begin{bmatrix} 1 & 1+D & 0 \\ 1+D & 1+D^2 & 1+D+D^2 \end{bmatrix}$. 这个编码器的输入是两个序列 $u_1(D)$, $u_2(D)$，输出是三个序列 $v_1(D)$, $v_2(D)$, $v_3(D)$。我们需要两组移位寄存器，第一组是面向第一路输入的，有一个寄存器，而第二组是面向第二路输入的，有两个寄存器。编码器的实现如下图所示。

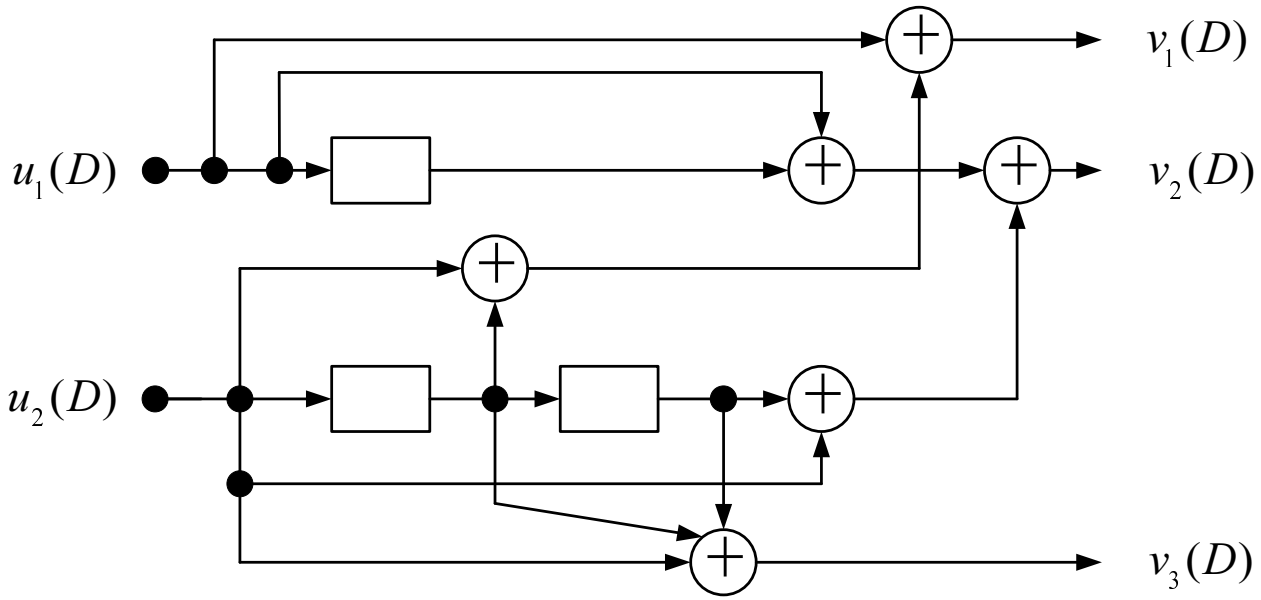


Figure: 码率为 $\frac{2}{3}$ 的卷积码对应的编码器实现图

编码器

我们前面定义了生成矩阵，其元素是有理函数 $\frac{f(D)}{g(D)}$ 的形成，是一种紧凑的表现形式，但看起来有点抽象。由于形式级数等价于序列，我们也可以考虑另外一种生成矩阵形式。我们以 $G(D) = (1 + D + D^2,\ 1 + D^2)$ 为例加以说明，其关键是考虑当输入序列为 $(1, 0, 0, \cdots, 0, \cdots)$ 时，编码器输出什么？两路输出分别是 $(1, 1, 1, 0, \cdots, 0, \cdots)$ 与 $(1, 0, 1, 0, \cdots, 0, \cdots)$。我们把这两个序列交错排列得到 $(11, 10, 11, 00, \cdots, 00, \cdots)$，这就构成了生成矩阵的第一行。一般地，生成矩阵的第 i 行应该对应于当输入是 $(\underbrace{0, \cdots, 0}_{i-1}, 1, 0, \cdots, 0, \cdots)$ 时的输出序列，这个序列就是第一行向右移位一个时刻，注意一个时刻对应两个输出符号，我们得到如下生成矩阵

$$\mathbf{G} = \begin{bmatrix} 11 & 10 & 11 & & \\ & 11 & 10 & 11 & \\ & & 11 & 10 & 11 \\ & & & & \ddots \end{bmatrix},$$

其中空白处是 0，可以看出 $\mathbf{G}$ 是一个带状矩阵。

编码器

一般地，一个卷积码的生成矩阵可以表示为如下形式，

$$\mathbf{G} = \begin{bmatrix} G_{0,0} & G_{0,1} & G_{0,2} & \cdots & G_{0,m} & & \\ & G_{1,1} & G_{1,2} & \cdots & G_{1,m-1} & G_{1,m} & \\ & & G_{2,2} & \cdots & G_{2,m-2} & G_{2,m-1} & G_{2,m} \\ & & & & & & \ddots \end{bmatrix},$$

其中 $G_{i,j}$ 是 $k \times n$ 矩阵，且 $G_{t,t}$ 是满秩的。如果第一行是 $(G_0, G_1, \cdots, G_m, 0, 0, \cdots)$，而其他行是上一行右移一个子块生成的，则对应的卷积码称为“时不变”的；否则，称为“时变”的。

上述生成矩阵是半无限矩阵，而在实际应用中，输入序列 u 是有限长的，输出序列 v 也应是有限长的，这种情况下，我们需把生成矩阵作相应处理。一种简单“粗暴”的方法是把列限制为有限，把其他部分略掉；另一种常用的方法是对编码器进行归零处理等价于把生成矩阵上边的行留下，而把其他行略掉。

卷积码图表示与译码算法

从卷积码的编码器看出，卷积码与分组码的最大差别是，卷积码是面向流的，输入可以是连续不断地，而在 t 时刻的输出，不仅仅依赖于当前输入，还与之前的输入有关，这种记忆性是编码器的寄存器状态决定的，因此，可以用多种与有限状态机有关的方法描述。一种是用树的方式，另一种是用网格图的方式。我们以例 11 中非系统编码器为例进行说明。

设编码器的初始状态是 $(0,0)$，我们可以用一棵二元树来描述，其根节点表示初始状态，第 t 层表示 t 时刻编码器状态，从第 $t-1$ 层到第 t 层的分支上对应标有 t 时刻的输出，习惯上一般把一个节点的左儿子对应输入 0，右儿子对应输入 1。

卷积码图表示与译码算法

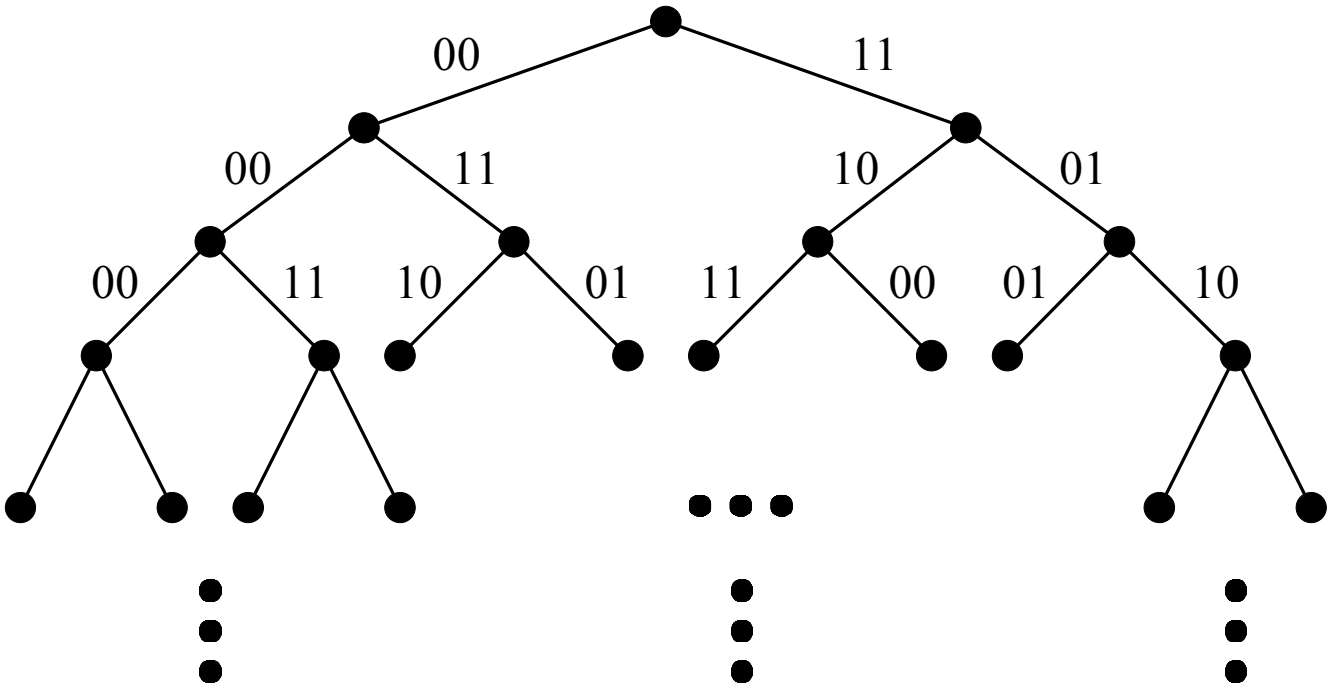


Figure: 卷积码 $G(D) = (1 + D + D^2, 1 + D^2)$ 对应的树图

卷积码图表示与译码算法

从编码树我们可以看出，一个有限长的码字序列对应从根节点到叶子节点的一条路径。例如，当输入序列是 $(0,1,0,\cdots)$ 时，对应的输出序列是“左, 右, 左, $\cdots$”，即 $(00,11,10,\cdots)$。当一个码字序列在信道中传输，收到一个有噪序列 $\underline{y}$，则译码问题就是在树中寻找一条路径，使得其对应的码字序列的似然值最大。当信道无记忆时，我们可以定义每条边的度量，进而定义每条路径的度量。有很多算法可用于搜索树上度量最大的路径，包括深度优先算法和宽度优先算法等，编码领域中基于树的译码算法通常称为序列译码算法，比较有名的是堆栈算法、Fano 算法等。

卷积码图表示与译码算法

卷积码的另一种表示图是网格图，可以看作是马尔可夫状态转移图在时间上展开，其中的状态就是寄存器中的内容。以生成矩阵 $G(D) = (1 + D + D^2, 1 + D^2)$ 为例，其有4个状态，分别是 $00, 01, 10, 11$。初始状态一般假设为 00，输入序列 $\underline{u}$ 导引状态在各个状态中转换，对应的网格图如下所示。

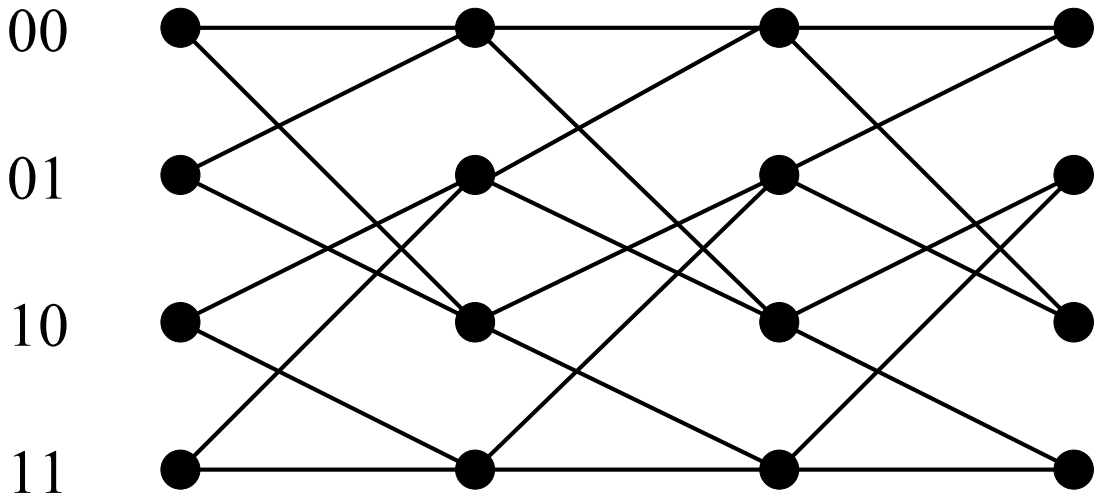


Figure: 卷积码 $G(D) = (1 + D + D^2, 1 + D^2)$ 对应的网格图

卷积码图表示与译码算法

编码器 $G(D) = (1 + D + D^2, 1 + D^2)$ 对应的网格图有 4 个状态，记为 $00, 01, 10, 11$，每条边由左状态输入、输出以及右状态刻画。我们可以用下表来表示这个网格图，为了方便起见，我们把状态用 0, 1, 2, 3 表示，即二进制对应的十进制表示。

边号	左状态	输入	输出	右状态
0	0	0	00	0
1	0	1	11	2
2	1	0	11	0
3	1	1	00	2
4	2	0	10	1
5	2	1	01	3
6	3	0	01	1
7	3	1	10	3

与树图类似，对于给定的接收序列，可以给每条边赋予一个度量值，一条路径的度量值可以由边的度量值“累积”得到。最大似然译码问题因而转化为最短路径算法，这就是著名的 Viterbi 算法。

仿真

卷积码的仿真实现主要是译码部分。仿真程序应定义生成矩阵（链接多项式）、“截断或结尾”方式等。若实现序列译码算法，则动态生成树，也就是说树图并不需要预先存储，而是边搜索边构造树。若要实现 Viterbi 算法，可以预先构造网格图。对于卷积码而言，我们只需要定义一节，包括左状态、输入、输出以及右状态等。

维特比算法

毫不夸张地说，任何时间离散的有限字符系统，如果不考虑网格图是否时变，状态数是否太多等情况，都可以用网格图来描述。

网格图（Trellis）又称为篱笆图，可以看作是由一节一节的子图拼接而成的。网格图在 t 时刻的一节可以描述为一个二分图，其顶点称为状态。左状态集合记为 $\mathcal{S}_{t-1}$，而右状态集合记为 $\mathcal{S}_t$。一个左状态与一个右状态之间可以有一条边或多条边连结，也可以没有边连结。一个边 $\underline{b}$ 可以由一个四元组刻画 $\underline{b} = (\sigma_-, i, o, \sigma_+)$，其中 σ_- 表示左状态，σ_+ 表示右状态，i 表示这条边所对应的输入，o 表示这条边所对应的输出。在有些情况下，i 与 o 可以相等，此时，四元组就可以简化成三元组。

维特比算法

最大似然算法是在篱笆图上找一条路径，使其对应的输出序列具有最大的似然函数值，即：
寻找 $\mathbf{v}^*$，使得 $P(\mathbf{v}^*)P(\mathbf{y}|\mathbf{v}^*) \geq P(\mathbf{u})P(\mathbf{y}|\mathbf{u})$，对于所有路径 $\mathbf{u}$ 都成立。
为此，对于每一条边，定义一个度量

$$\gamma(s_t, u_t, v_t, s_{t+1}) = \log P_U(u_t) + \log P(y_t|v_t).$$

一条路径的度量是其所有边的度量之和。设 $\underline{b} = (b_0, b_1, \cdots, b_{L-1})$ 是一条路径，其度量记作 $\Gamma(\underline{b}) = \sum_{t=0}^{L-1} \gamma(b_t)$。因此，最大似然译码是在篱笆图中寻找一条度量最大的路径。这是典型的动态规划问题，可以用维特比算法予以解决，其基本原理是：全局最优的路径，局部一定是最优的。

维特比算法

更具体地，我们有如下问题，如下图所示。

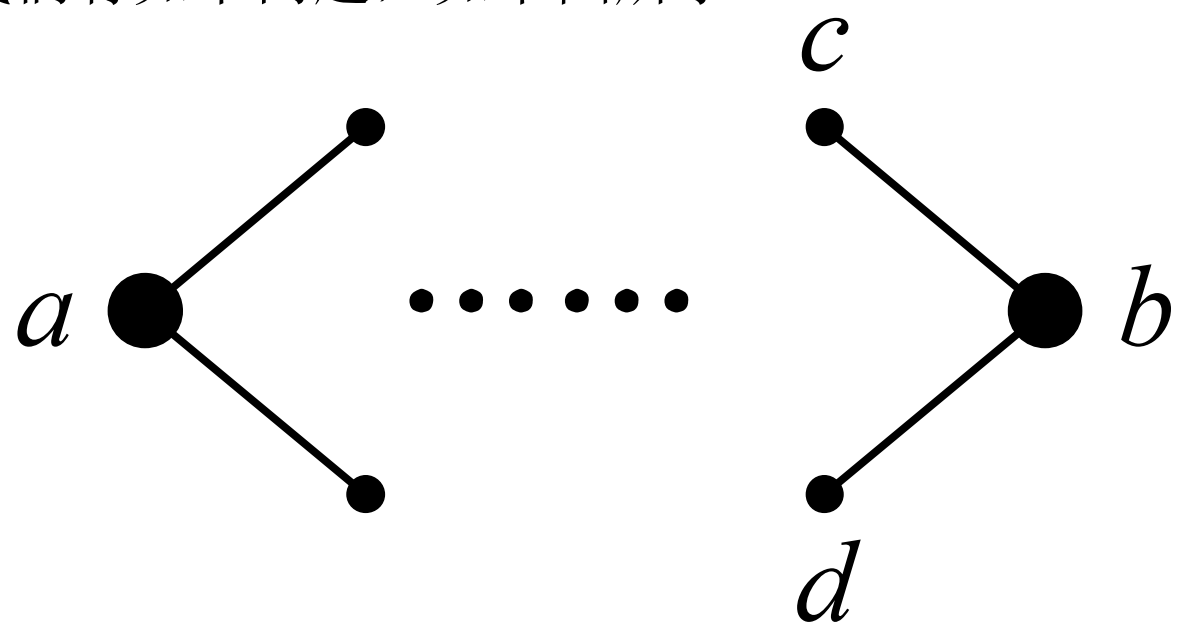


Figure: 网格图

从 a 到 b 的最优路径，要么经过 c，要么经过 d。若经过 c，则从 a 到 c 的路径必须是最优的，否则，我们可以选择从 a 到 c 的最优路径予以替换。同理，若经过 d，则从 a 到 d 的路径必须是最优的。综上，求从 a 到 b 的最优路径可以约简为计算从 a 到 c 以及从 a 到 d 的最优路径。

维特比算法

根据这一原理，得到如下算法，即维特比算法。为简单起见，假定初始状态与末状态均已知，记为 0 状态。

1. **初始化**：令 $t = 0$ 时刻的累积度量是 $\Gamma_0(s_0) = 0$，其中，$s_0 = 0$ 是已知的初始状态。若在网格图的第一节还有其他左状态 s，则置 $\Gamma(s) = -\infty$。
2. **递推**：对于每个状态 $s_t \in \mathcal{S}_t$，其中，$t = 1, 2, \cdots$，按如下方式计算进入 s_t 的最优路径的累积度量。

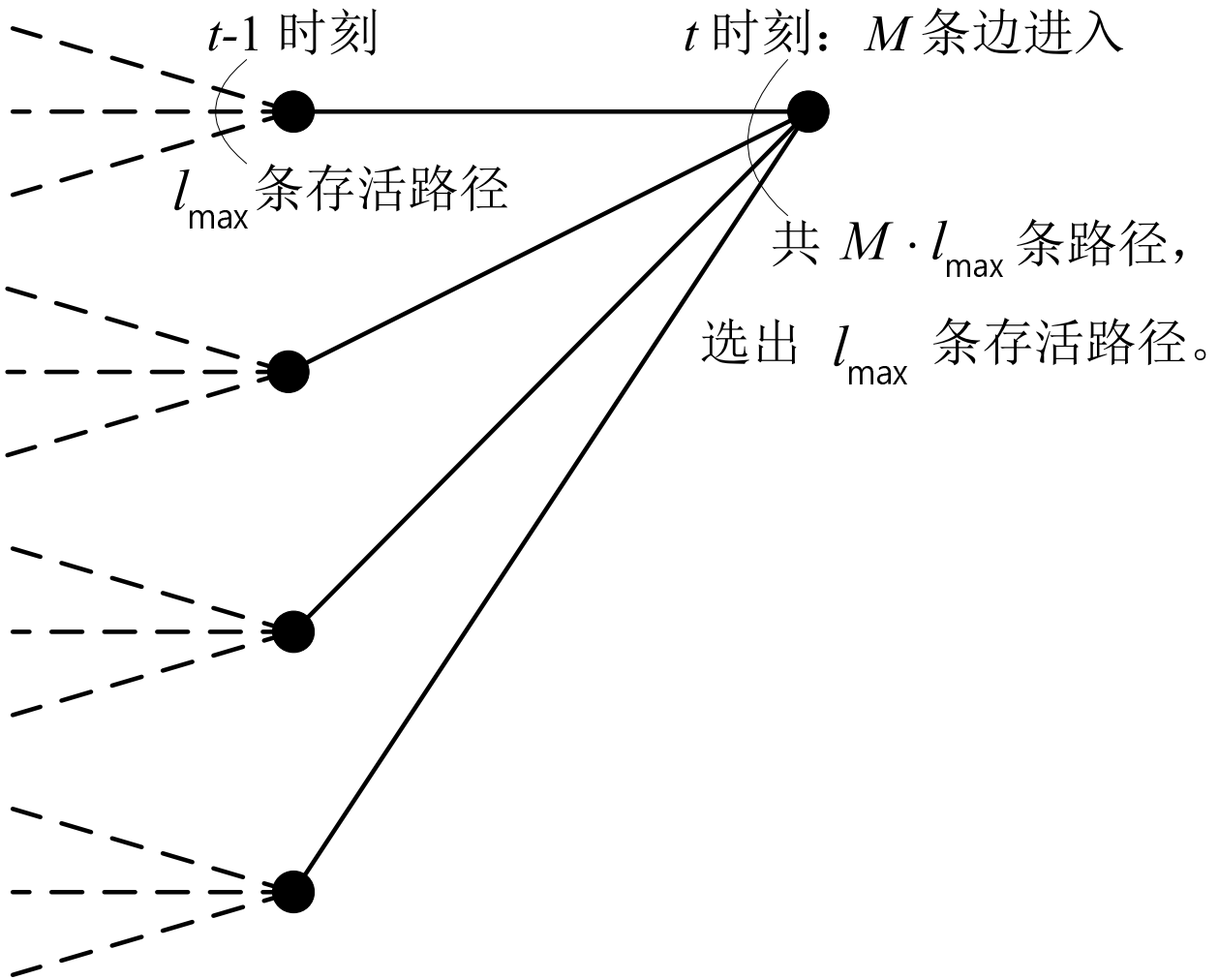
$$\Gamma(s_t) = \max\{\Gamma_{t-1}(s_{t-1}) + \gamma(s_{t-1}, u_t, v_t, s_t)\}.$$

   其中，max 操作是对所有进入 s_t 的边进行的，并以此记录最优的边 $b_t^* = (s_{t-1}, u_t, v_t, s_t)$。
3. **回溯**：在最后时刻，我们得到 $\Gamma_L(S_L = 0)$，并得到产生此最优累积度量的边 b_L^*。由此，可以得到 b_L^* 的左端点 s_{L-1}^*，而进入 s_{L-1}^* 的边也可“顺藤摸瓜”找出。于是，就得到了最优路径。

列表维特比算法

维特比算法的输出是一条具有最大度量的路径，在有些场景下，我们需要得到度量最大的 $l_{\max}$ 条路径。有两种方法实现这个功能，一种是串行的，另一种是并行的。并行的列表维特比算法很容易理解，且可以在维特比算法的基础上稍加修正即可。在每个时刻，每个状态存储至多 $l_{\max}$ 条进入的路径。若一个状态有 M 条进入的边，则进入该状态的 $l_{\max}$ 路径可以由上一时刻的最优路径拓展而来。事实上，每条边对应 $l_{\max}$ 条路径，所以，共有 $M \cdot l_{\max}$ 条路径，然后从中选出 $l_{\max}$ 条存活路径即可。下面给出一个示意图。

列表维特比算法



作业

Exercise 1.[王育民(2013)]
设有一系统$(3,1,5)$ 二元卷积码

$$\mathbf{g}^{(2)} = (101101), \quad \mathbf{g}^{(3)} = (110011)$$

求码的生成矩阵**G**。

Exercise 2.[王育民(2013)]
设系统$(3,2,3)$ 二元卷积码的生成元为

$$\mathbf{g}^{(1,3)}(x) = 1 + x^2 + x^3, \quad \mathbf{g}^{(2,3)}(x) = 1 + x + x^3$$

试画出编码电路。

谢谢！