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. 卷积码
   - 编码器
   - 卷积码图表示与译码算法
   - 译码：维特比算法

2. Turbo码
   - 用短码构造长码的方式
   - 级联码
   - 译码框架
   - 仿真

卷积码

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 1

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

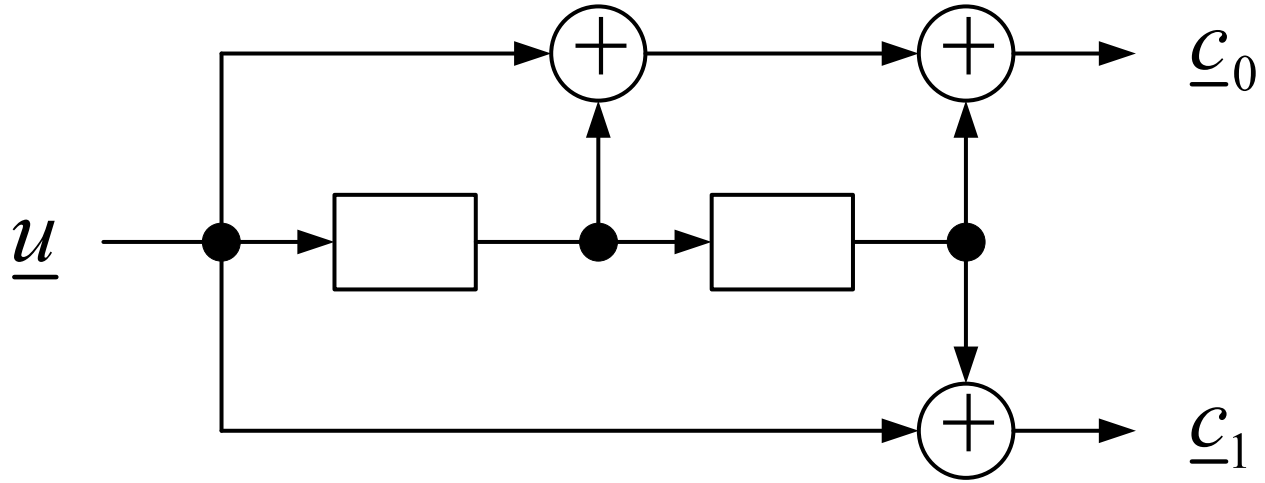


Figure 1: 非系统编码器 $G(D) = (1 + D + D^2,\ 1 + D^2)$

编码器

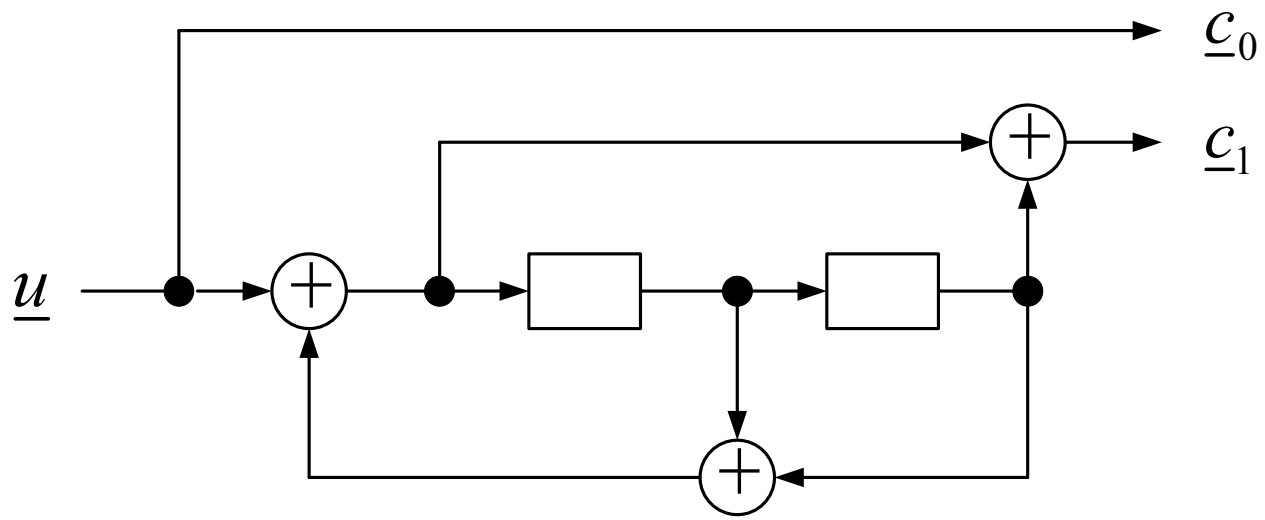


Figure 2: 递归系统编码器 $\tilde{G}(D) = (1, \frac{1+D^2}{1+D+D^2})$

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。

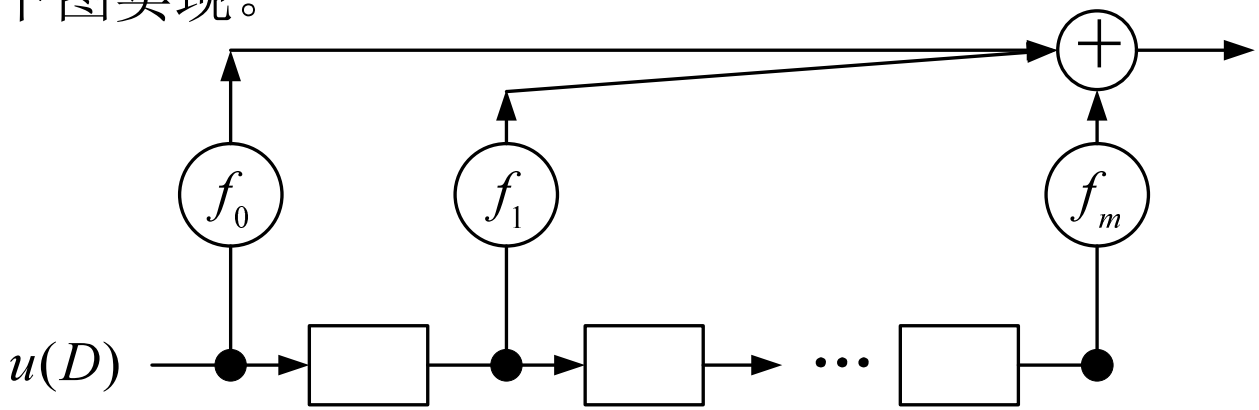


Figure 3: 多项式乘法器，$v(D) = u(D)f(D)$

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。

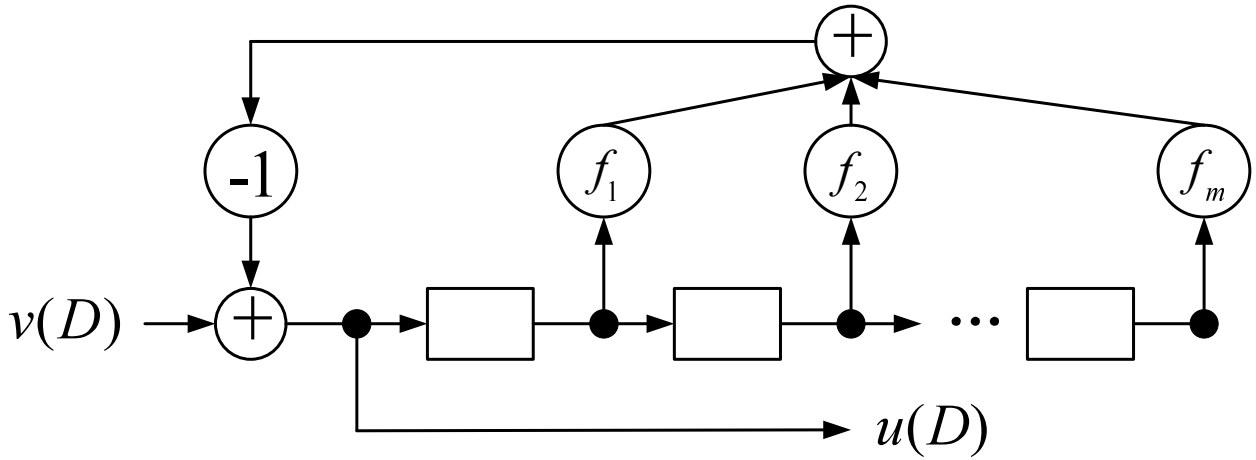


Figure 4: 多项式除法器，$u(D) = \frac{v(D)}{f(D)}$

编码器

Example 2

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。

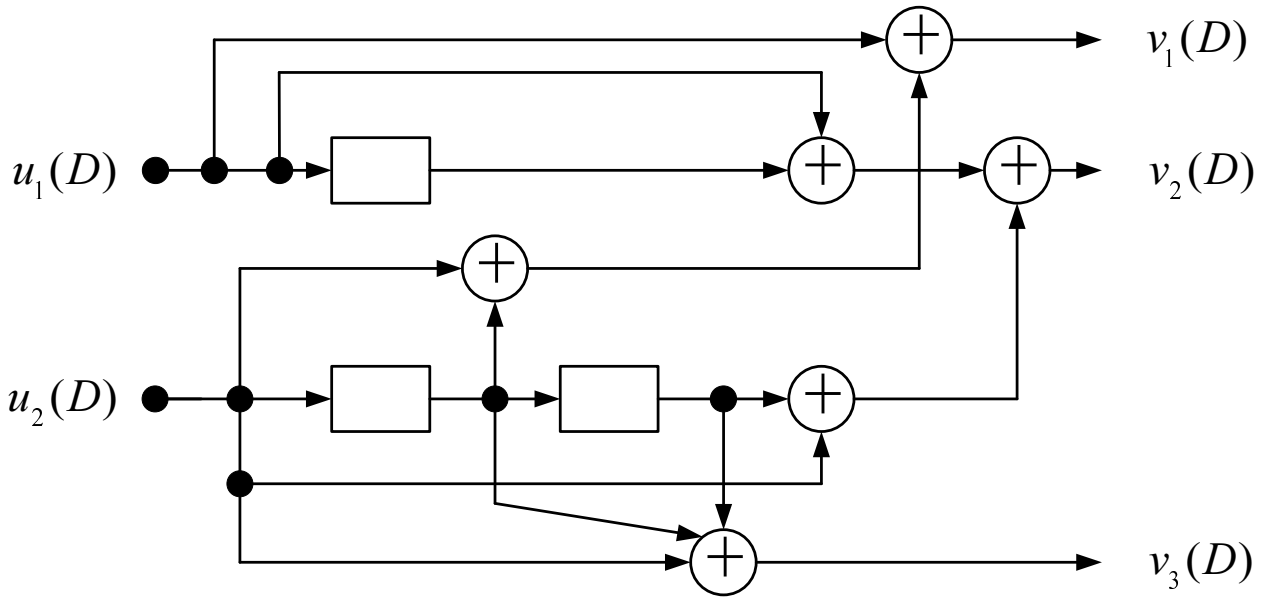


Figure 5: 码率为 $\frac{2}{3}$ 的卷积码对应的编码器实现图

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

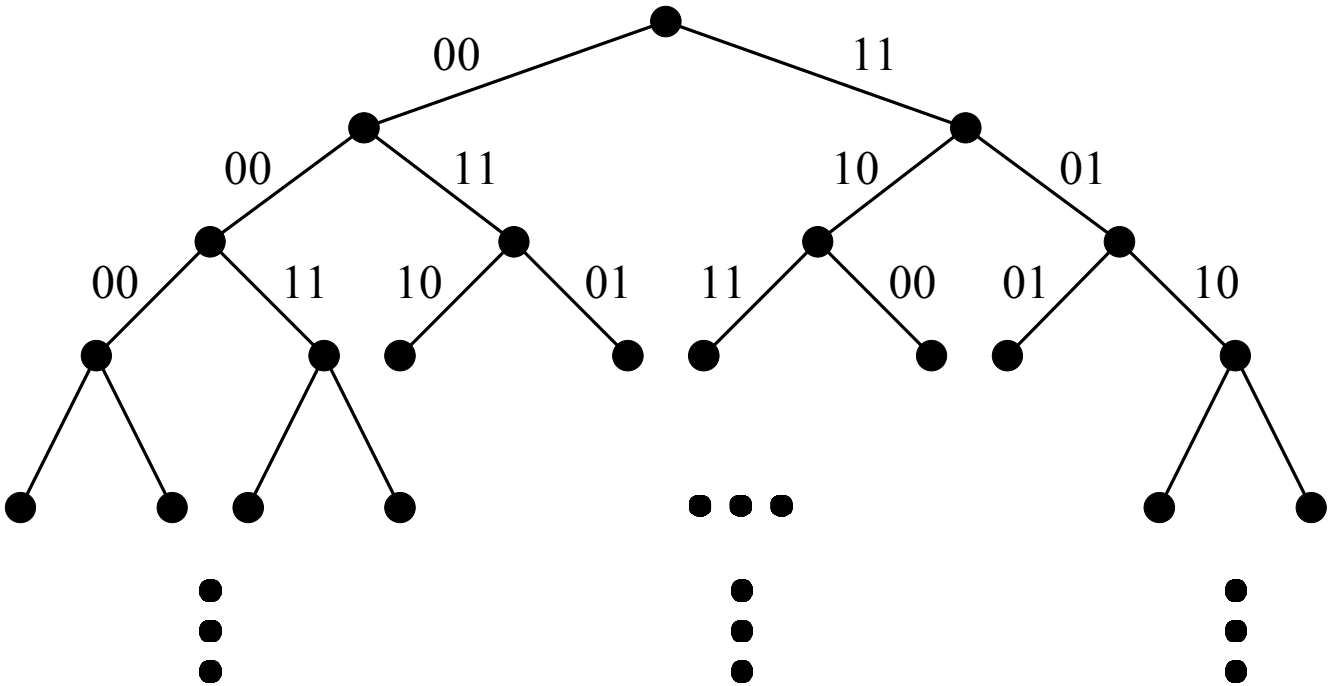


Figure 6: 卷积码 $G(D) = (1 + D + D^2, 1 + D^2)$ 对应的树图

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。

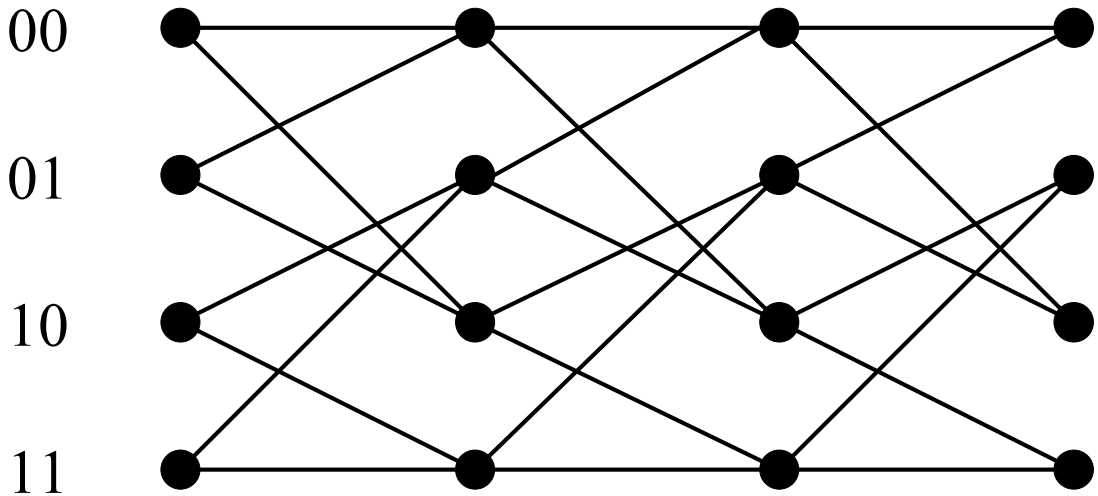


Figure 7: 卷积码 $G(D) = (1 + D + D^2, 1 + D^2)$ 对应的网格图

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。

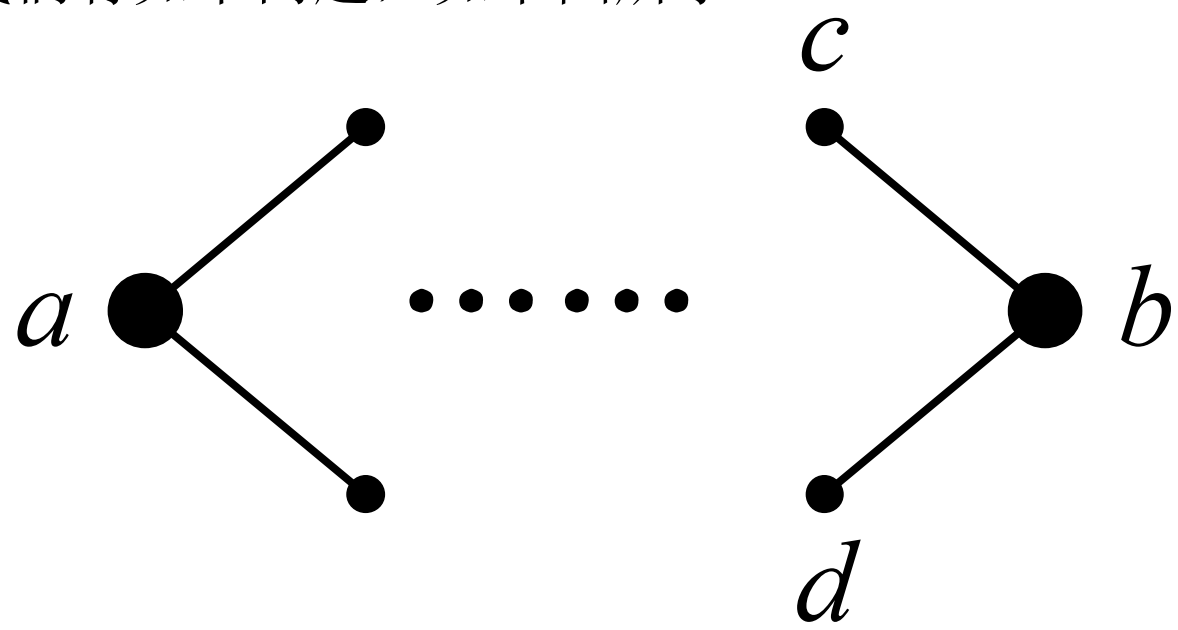


Figure 8: 网格图

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。

维特比算法

Example 3

对于二元(2,1,2)码，$G(D) = (1 + D + D^2, 1 + D^2)$。设发送码序列 $\mathbf{c} = (11\ 10\ 00\ 01\ 10\ 01\ 11)$, 相应的信息序列 $\mathbf{u} = (10111)$。现在通过BSC传送,接收序列为 $\mathbf{R} = (10\ 10\ 01\ 01\ 10\ 01\ 11)$。在BSC下,最大似然译码就是最小距离译码,路径值可用式 $d(\mathbf{R}, \mathbf{c}_i) = \sum_{l=0}^{L+m-1} d(\mathbf{R}, \mathbf{c}_{il})$代替。从状态 a_0 到状态 a_7,共有32种可能的路径,任一条路径或任一条从 a_0 到 a_7 的状态序列与接收序列 $\mathbf{R}$ 之间的距离易于从图9求得。例如对 $a_0 b_1 c_2 a_3 a_4 b_5 c_6 a_7$, 相应码字为 (11 10 11 00 11 10 11) 与接收序列 $\mathbf{R}$ 的距离为 $1 + 0 + 1 + 1 + 1 + 2 + 0 = 6$。按Viterbi 算法进行的结果如图10中粗线所示。在各节点上都注出了到达此节点的幸存路径的长度,最后得出最短路径为 $a_0 b_1 c_2 b_3 d_4 d_5 c_6 a_7$, 长度为2, 译码器输出为 $\hat{\mathbf{u}} = (10111)$。

维特比算法

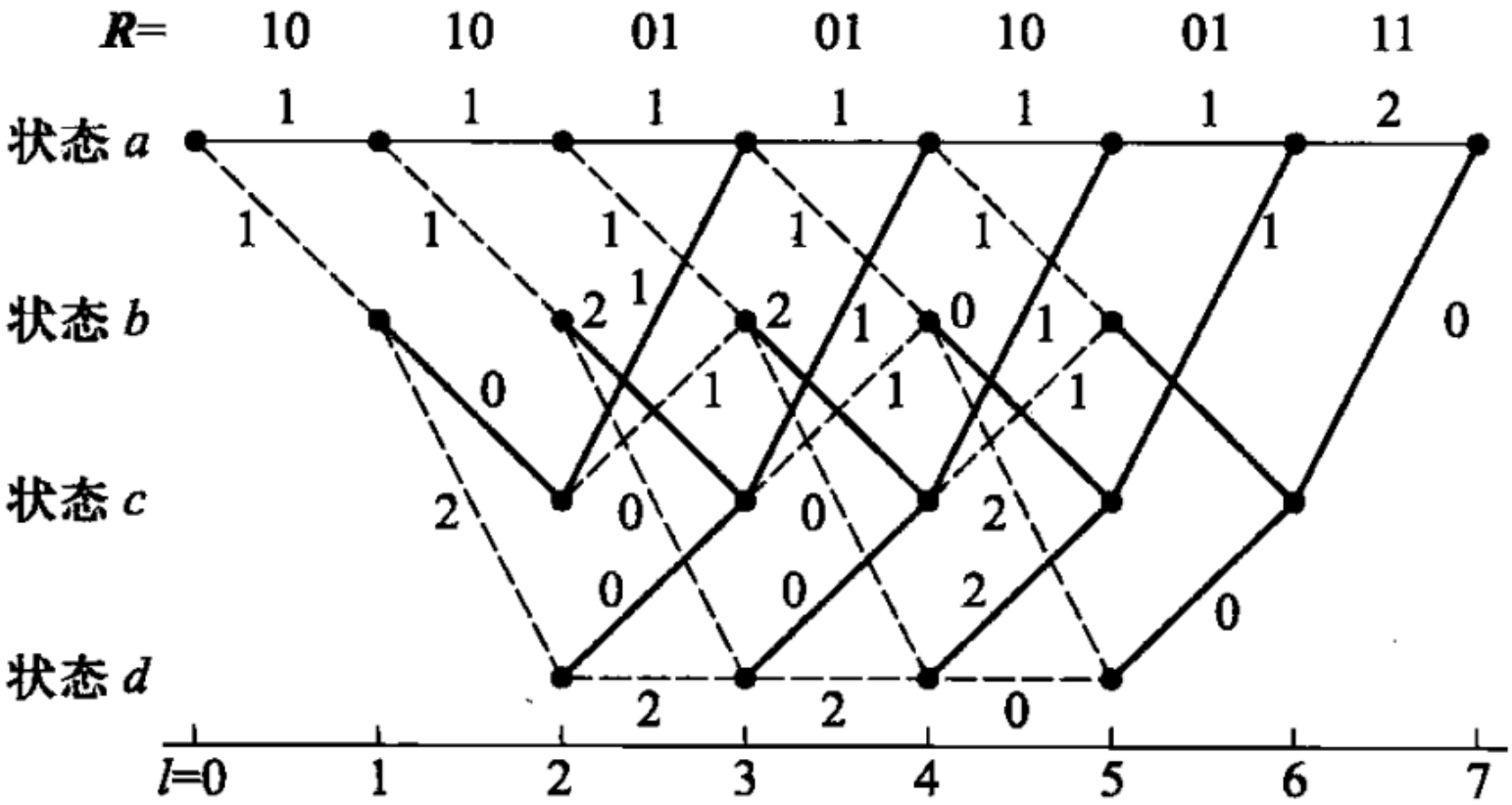


Figure 9: Viterbi译码的距离度量

维特比算法

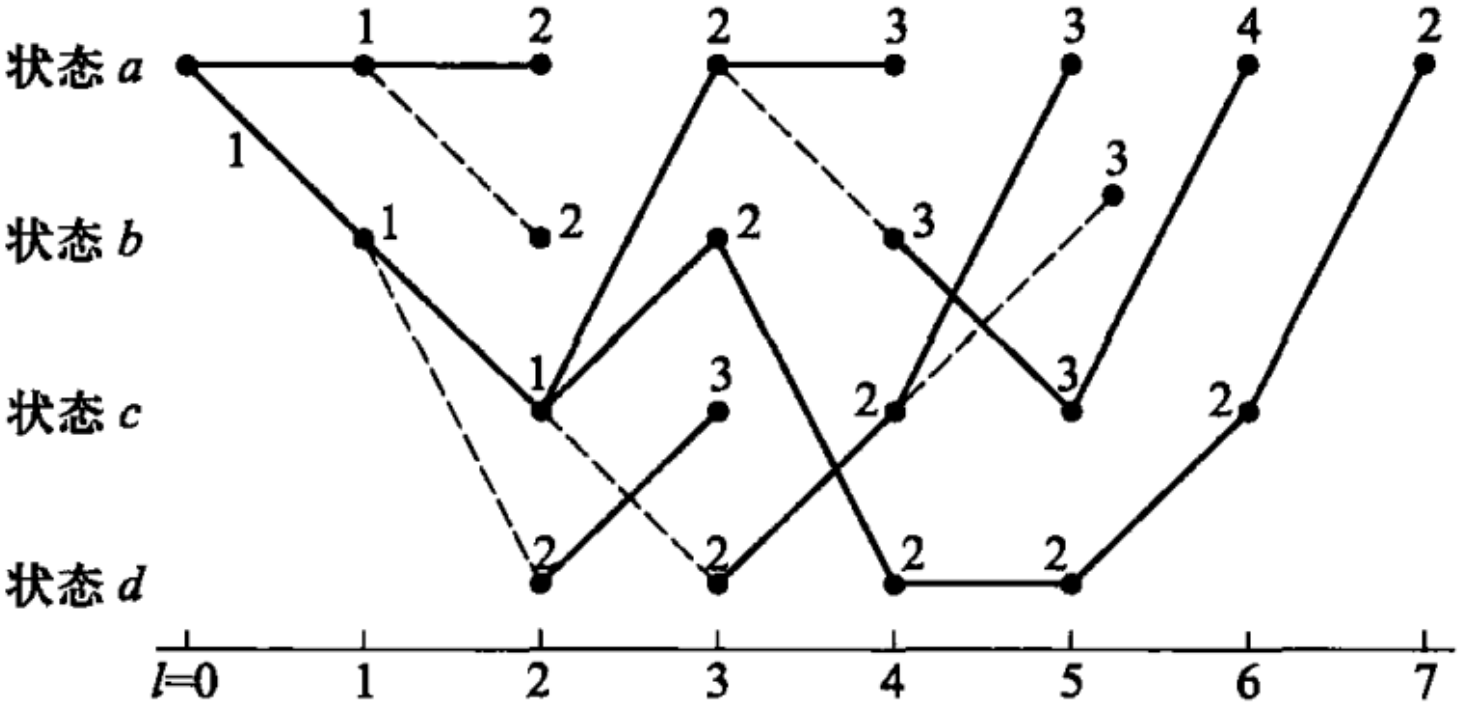
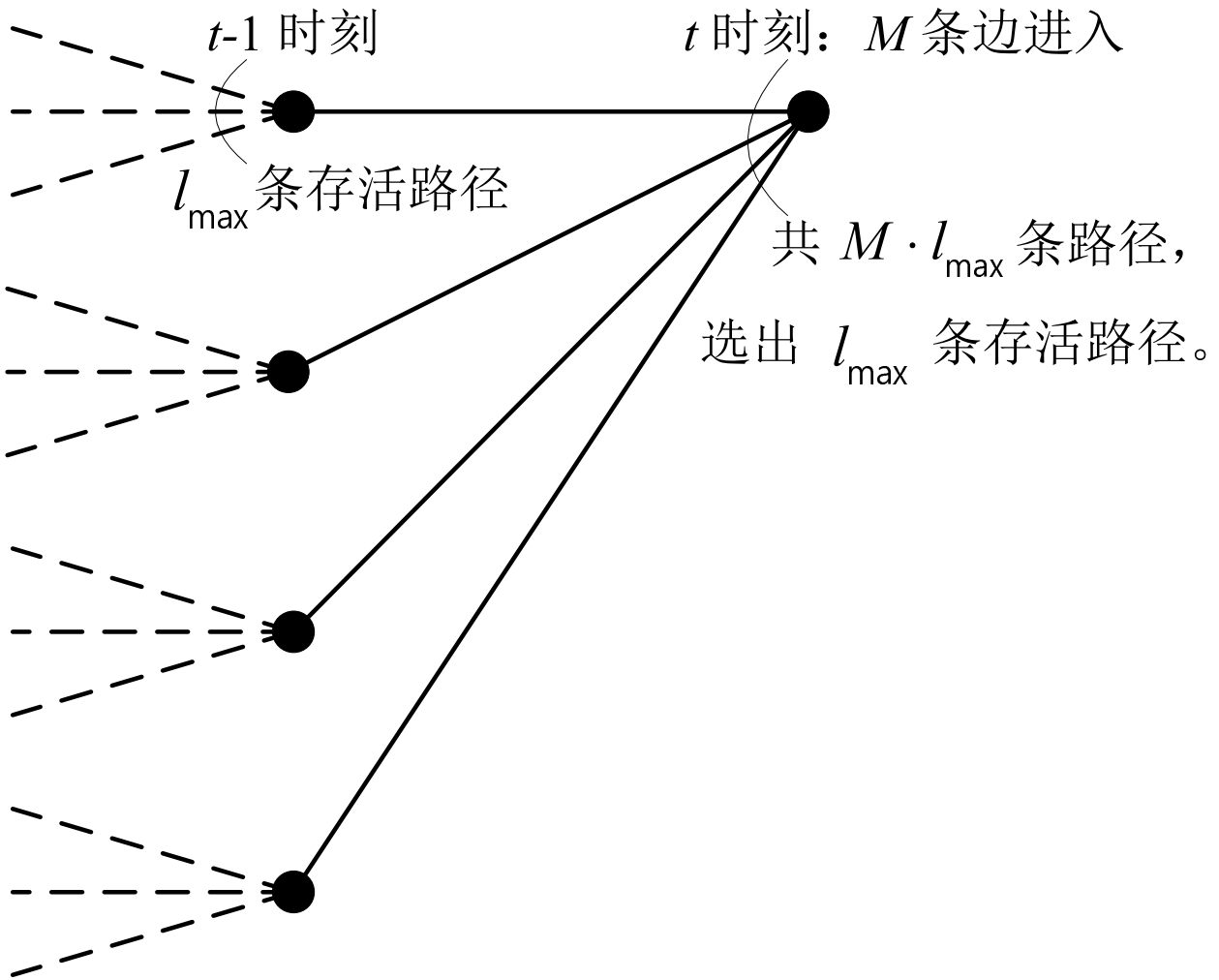


Figure 10: 幸存路径

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



Turbo码

用短码构造长码的方式

通常来讲，短码是指码长较短的分组码或者约束度较小的卷积码，其编译码算法可以在工程中实现。但从信道编码定理看，要逼近信道容量，码长需要充分长。在编码领域，一种实用的方法就是从简单码出发，构造性能好的长码。比较著名的例子有乘积码、级联码等，分别简述如下。

用短码构造长码的方式

乘积码最早由 Elias 提出。设 $\mathscr{C}_1[n_1, k_1]$ 与 $\mathscr{C}_2[n_2, k_2]$ 是两个线性分组码，则乘积码的构造（也即编码）可以结合如下示意图进行描述。

$k_2 \times k_1$ 信息位	行码校验
列码校验	校验之校验

第一步，把信息位（共 $k_1 \times k_2$）排列成一个矩阵，每行 k_1 位，每列 k_2 位。

第二步，按行编码，得到行校验位，换句话讲，得到 k_2 个码字，取自 $\mathscr{C}_1$，这些码字排列在码字矩阵的前 k_2 行。

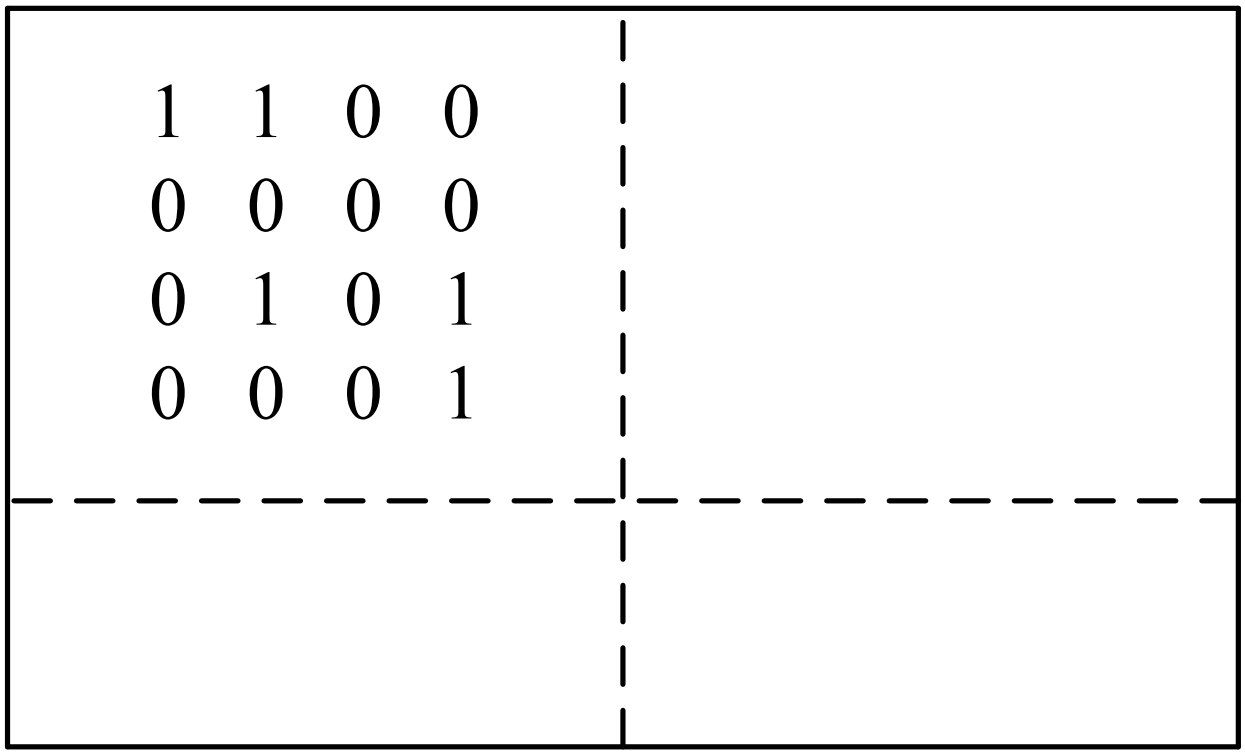
第三步，按列编码，即，对于上述码字矩阵的每一列，按照 $\mathscr{C}_2$ 的规则进行编码，得到列校验或者校验之校验。

可以看出，乘积码是分组码，其维数是 $k_1 \times k_2$，码长是 $n_1 \times n_2$。

用短码构造长码的方式

Example 4

以汉明码 $[7,4,3]$ 为例进行说明，我们可以构造一个乘积码，其维数是 16。设输入信息序列 $\underline{u} = (1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,1)$，按照上述步骤，可以得到如下码字矩阵。



可以验证，先按列编码，再按行编码，可以得到相同的码字矩阵。

用短码构造长码的方式

级联码由 Forney 提出，由一个分组码（通常是代数码、RS 码和 BCH 码等）和一个卷积码按照串行的方式构成，如图 11 所示。

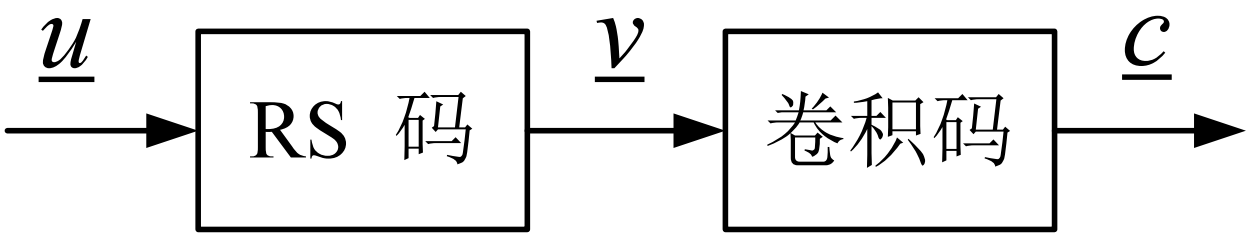


Figure 11: 级联码

信息序列 $\underline{u}$ 分组之后进入代数编码器，如 RS 编码器，得到序列 $\underline{v}$，然后进入卷积码编码器，得到码字序列 $\underline{c}$。我们称卷积码是内码，RS 码是外码。译码时，先从接收序列估计内码的码字 $\hat{c}$，再用外码译码器纠正 $\hat{c}$ 中可能剩余的错误。按照 Forney 的观点，外码把内码的编译器连同信道一起看作一个“人工信道”。设外码的码率是 $R_o = \frac{k_o}{n_o}$，内码的码率是 $R_i = \frac{k_i}{n_i}$，则系统的码率是 $R = R_o \cdot R_i$。Forney 提出的级联码已在卫星通信标准中得到应用。

级联码

现代的级联码始于 1993 年的忒拨（Turbo）码，与传统级联码的主要差别是其利用了迭代译码算法。目前，迭代译码的原理没有完全解释清楚，人们习惯称之为“Turbo 原理”，已在多个领域得到应用。
1993 年 Berrou 等提出的忒拨码基于卷积码构造，后已被推广到很多类似构造。这里编码的一个主要部件是随机交织器，与乘积码类似，信息序列被编码两次，一次是原序列，另一次是经过交织器的信息序列。以卷积码作为例子，我们有如下的忒拨码构造。

级联码

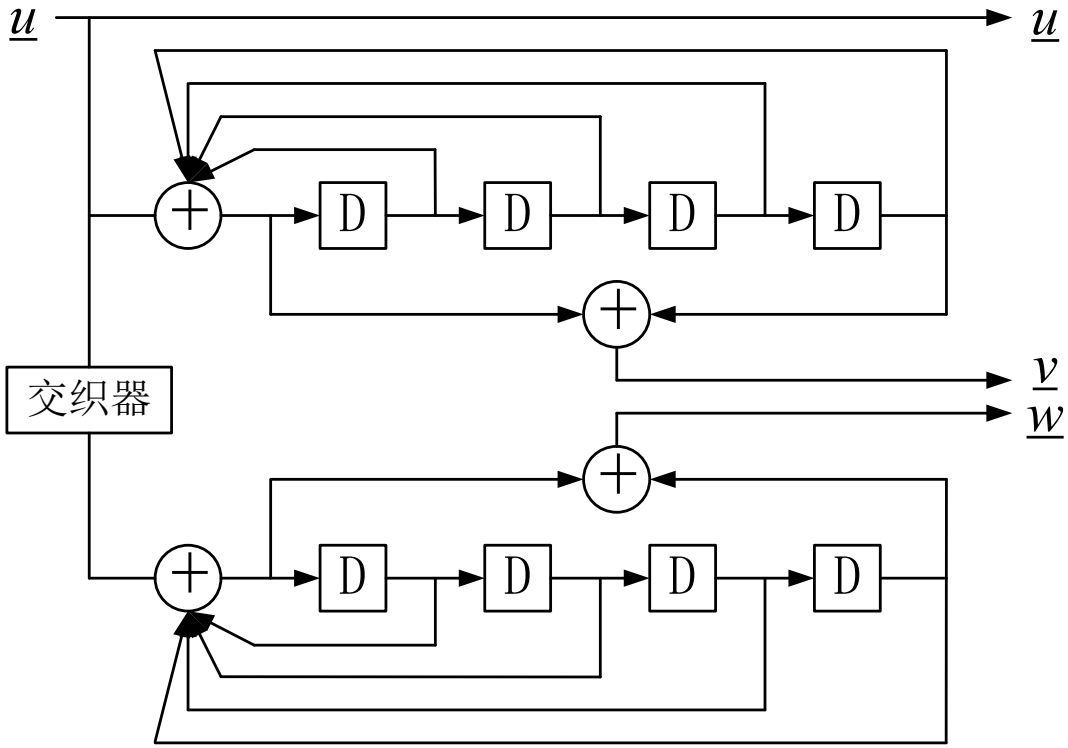


Figure 12:

从上图看出，信息序列 $\underline{u}$ 可以产生校验序列 $\underline{v}$，经过交织器后又产生序列 $\underline{w}$，最终的码字由 $(\underline{u}, \underline{v}, \underline{w})$ 构成。其中，可以“打掉”（puncture） $\underline{v}$ 与 $\underline{w}$ 序列中某些比特以调整速率。比如，如果 $\underline{v}$ 取偶数位，$\underline{w}$ 取奇数位，则得到码率是 $\frac{1}{2}$。上述 12 图是忒拨码最初的架构，显然可以推广更一般的形式，如下图。

级联码

上述 12 图是忒拨码最初的架构，显然可以推广更一般的形式，如下图。

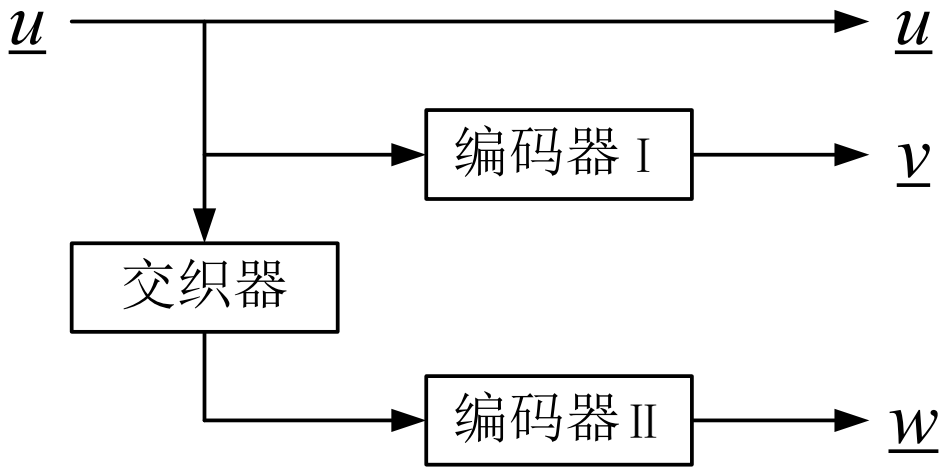
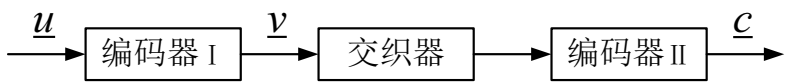


Figure 13:

我们对编码器 I 与 II 的要求是，要有有效的软输入软输出（soft input soft output, SISO）译码算法。

上述忒拨码可以看作是并行级联码，我们也可以构造串行级联码，如下图。



编码的过程与 Forney 经典的级联码一致，即信息序列 $\underline{u}$ 进入编码器 I 得到中间序列 $\underline{v}$，经过交织器后进入编码器 II，得到码字序列 $\underline{c}$。

译码框架

忒拨码，无论是串行级联的，还是并行级联的，其译码算法的本质与 LDPC 码的译码算法具有相同的译码框架。前面已经指出，忒拨码的构造需要成份码具有有效的软输入软输出译码算法。我们先看如何描述成份码的译码算法，再给出忒拨码的译码框架。

译码框架

一个码可以用一个简单的约束图来描述，左端对应输入 u，右端对应输出 v。

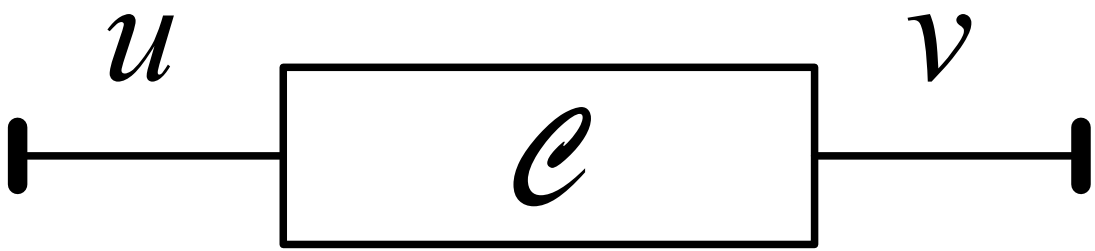


Figure 14:

我们可以认为 u 连着“信源”， v 连着“信道”，u 和 v 都是序列（可以认为是随机序列）。设 u_i 是 u 的第 i个分量，我们用它的概率质量函数 $(P_{u_i}(0), P_{u_i}(1))$ 或者其他等价的量表示 u_i 的信息或者消息。所有分量的信息表示为 $P_u^a(u)$，同样地，$P_v^a(v)$ 表示序列 v 的信息。初始地，我们用 $P_u^a(u), P_v^a(v)$ 表示相应的先验信息，这些量的计算可以从“信源”（相应地，“信道”）的统计规律获得，我们假定 u, v 的各个分量是“独立”的，不受编码约束。所谓软输入- 软输出译码算法，就是根据 $P_u^a(u), P_v^a(v)$，并考虑编码约束来计算外信息，分别记作 $P_u^c(u), P_v^c(v)$。需要指出的是，如果一个随机变量先验未知，我们可以假定对应的信息是 $(\frac{1}{2}, \frac{1}{2})$，外信息的计算可以用精确的算法，也可以考虑低复杂度的近似计算。

译码框架

从原理上讲，精确的计算公式如下，

$$P^e_{u_+}(0) \propto \sum_{u_+=0} P^a_{u'}(u) \cdot P^a_v(v)$$

$$P^e_{u_+}(1) \propto \sum_{u_+=1} P^a_{u'}(u) \cdot P^a_v(v),$$

在公式 $P^e_{u_+}(0)$ 中，求和是对所有满足编码约束，且 $u_+ = 0$ 的 u, v 进行的，而 $P^a_{u'}(u)$ 中的 u' 表示不含 $P^a_{u_+}(0) \cdot P^a_{u_+}(1)$。类似地，$v_+$ 的外信息计算如下：

$$P^e_{v_+}(0) \propto \sum_{v_+=0} P^a_u(u) \cdot P^a_{v'}(v)$$

$$P^e_{v_+}(1) \propto \sum_{v_+=1} P^a_u(u) \cdot P^a_{v'}(v).$$

译码框架

并行级联类的忒拨码可以用如下正规图表示，

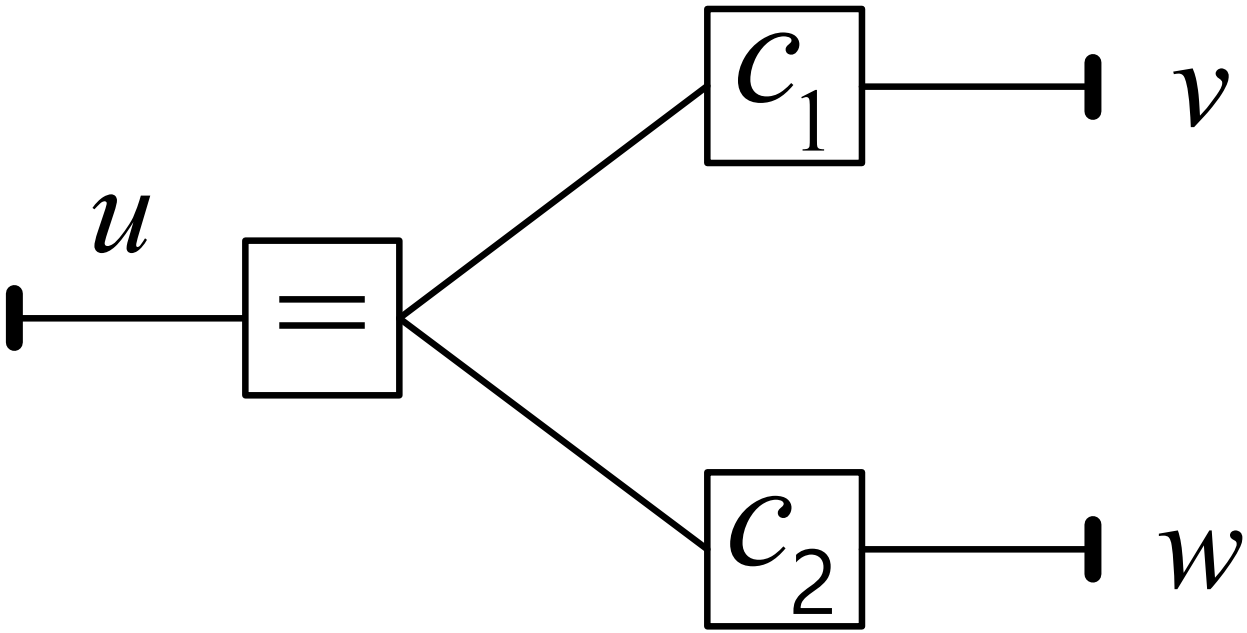


Figure 15:

其中，[=] 节点表示所连边的变量必须取相同的值，[c_1] 表示成份码 I 的约束，意指 u 对应的码字必须是 v，[c_2] 类似。

译码框架

并行级联类忒拨码的译码框架如下，

初始化：设序列 (u, v, w) 的消息均已获得，记作 $P_u^{|\to=}, P_v^{|\to c_1}, P_w^{|\to c_2}$，中间边 $\boxed{=}$—$\boxed{c_1}$ 与 $\boxed{=}$—$\boxed{c_2}$ 上的消息初始化为 $(\frac{1}{2}, \frac{1}{2})$。

迭代：做如下迭代，最大次数 I_{max} 满足某个预设条件。

1. 在 $\boxed{=}$ 节点，利用输入信息，按照外信息的方式计算 $P^{=\to c_1}$，

即 $(P_u^{|\to=}, P^{c_2\to=}|=) \xrightarrow{\text{计算}} P^{=\to c_1}$；

2. 在 $\boxed{c_1}$ 节点，按照外信息的方式计算 $(P^{=\to c_1}, P_v^{|\to c_1}|c_1) \xrightarrow{\text{计算}} P^{c_1\to=}$；

3. 在 $\boxed{=}$ 节点，按照外信息的方式计算 $(P_u^{|\to=}, P^{c_1\to=}|=) \xrightarrow{\text{计算}} P^{=\to c_2}$；

4. 在 $\boxed{c_2}$ 节点，按照外信息的方式计算 $(P^{=\to c_2}, P_w^{|\to c_2}|c_2) \xrightarrow{\text{计算}} P^{c_2\to=}$。

判决：在 $\boxed{=}$ 节点，按照全信息方式，计算 $(P_u^{|\to=}, P_{c_1}^{|\to=}, P_{c_2}^{|\to=}) \xrightarrow{\text{计算}} P_u$，根据 P_u 进行判决，得到 $\hat{u}$。

译码框架

我们在上述图 15 中略去了交织器，其信息处理较为简单，即把输入信息经过交织、解交织后得到输出信息。类似地，串行级联忒拨码的正规图如下，

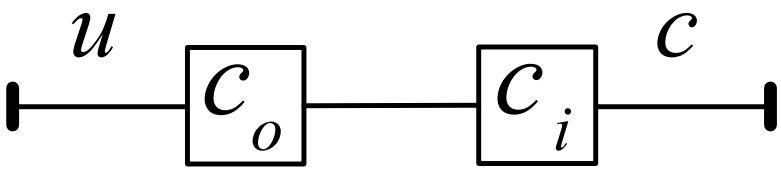


Figure 16:

译码框架如下：

初始化：设 $P_u^{|\to c_o}, P_c^{|\to c_i}$ 已经获得，边 $\boxed{c_o}$—$\boxed{c_i}$ 上的消息初始化为均匀随机变量序列。

迭代：做如下步骤，最大迭代次数 I_{max} 满足某个预设条件。

1. 在 $\boxed{c_i}$ 节点，计算 $(P^{c_o\to c_i}, P_c^{|\to c_i}|c_i) \xrightarrow{\text{计算}} P^{c_i\to c_o}$；
2. 在 $\boxed{c_o}$ 节点，计算 $(P_u^{|\to c_o}, P^{c_i\to c_o}|c_o) \xrightarrow{\text{计算}} P^{c_o\to c_i}$；

判决，在 $\boxed{c_o}$ 节点，计算全信息 $(P_u^{|\to c_o}, P^{c_i\to c_o}|c_o) \xrightarrow{\text{计算}} P_u$，根据 P_u，判决得到 $\hat{u}$。

仿真

仿真程序的主要模块包括交织器，软输入软输出译码器。交织器的实现可以有随机分组交织器、卷积交织器、确定型的交织器等。一个软输入软输出译码算法的有效实现，我们在以后的章节中讨论，当前需要明白的是一个编码模块接收的是从“信源”来的序列，输出的是面向“信道”的序列。译码模块的软输入包括信源的先验信息，也包括信道的似然信息；软输出是指两个变量的外信息，即给定变量某个特定的取值，其所参与的系统约束满足的概率。软输入既可以是针对信源符号的，也可以是针对信道符号的。

作业

Exercise 1.[王育民(2013)]
设$(3,2,1)$ 二元卷积码的生成多项式矩阵

$$G(x)=\begin{bmatrix} 1+x & 1 & 1+x \\ x & 1+x & 0 \end{bmatrix}$$

试画出它的编码电路、状态图。假定$L=6$ 的码序列通过转移概率为$P=0.01$ 的BSC 传送，若接收到的序列为 01101111110010100110 1,试用Viterbi 算法译码。

谢谢！