Convolutional Codes

Jinlong Li

Nature Inspired Computation and Applications Laboratory
University of Science and Technology of China (USTC)
Hefei 230027, Anhui, China
jlli@ustc.edu.cn · http://staff.ustc.edu.cn/~jlli

November 1, 2011

Contents

简介

卷积码的特点(1)

- Elias 提出，Viterbi 提出最大似然的译码方法：Viterbi 译码算法；
- 卷积码是非分组码，且有记忆，即任意时段，编码器的输出不仅仅与当期输入相关而且和以前的输入相关；
- 卷积码可表示为(n,k,L)码，k 为信息块的大小，n 为编码输出块的大小，而 m 表示记忆的“深度”，也就是说编码输出和 L+1 级的输入相关，称 L+1 为编码约束长度；
- 译码时，根据约束长度内所有的接受序列（L+1 个），来提取当前组的信息序列；

卷积码的特点（2）

- 优点：利用组之间的相关性，n 和 k 较小时，同样的 R 和设备复杂性，较分组码性能更好；
- 不足：数学分析工具缺乏，码的参数如何获得？困难！
- 一般设计码时，n 和 k 较小，而 L 较大，此时码简单、性能好。

卷积码的编码

卷积码的编码（1）

- k 个输入端，n 个输出端，L 节移位寄存器所构成的有限状态记忆系统（时序网络），如下图所示

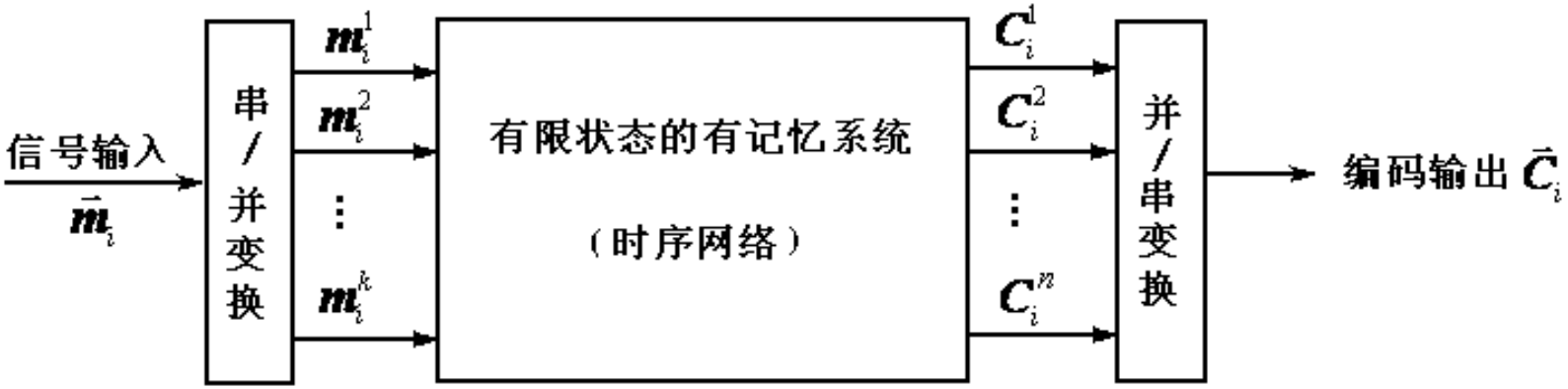


Figure: 卷积码编码原理图

卷积码的编码（2）

编码器的内部连接关系图：

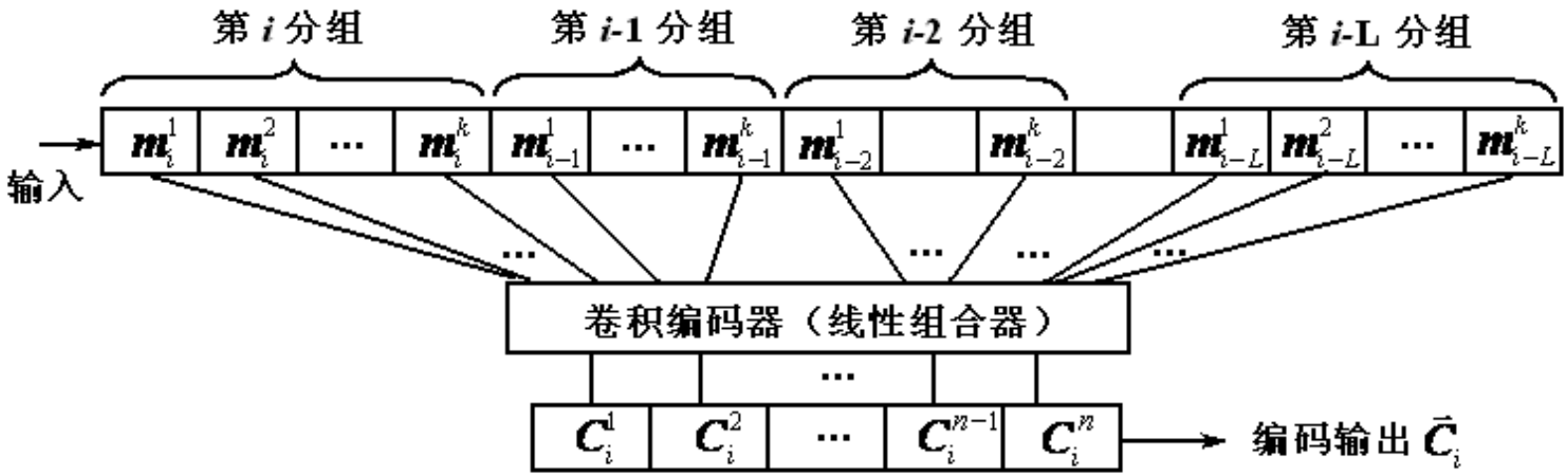


Figure: 编码器内部连接关系

卷积码的编码（3）

更详细的编码器原理图：

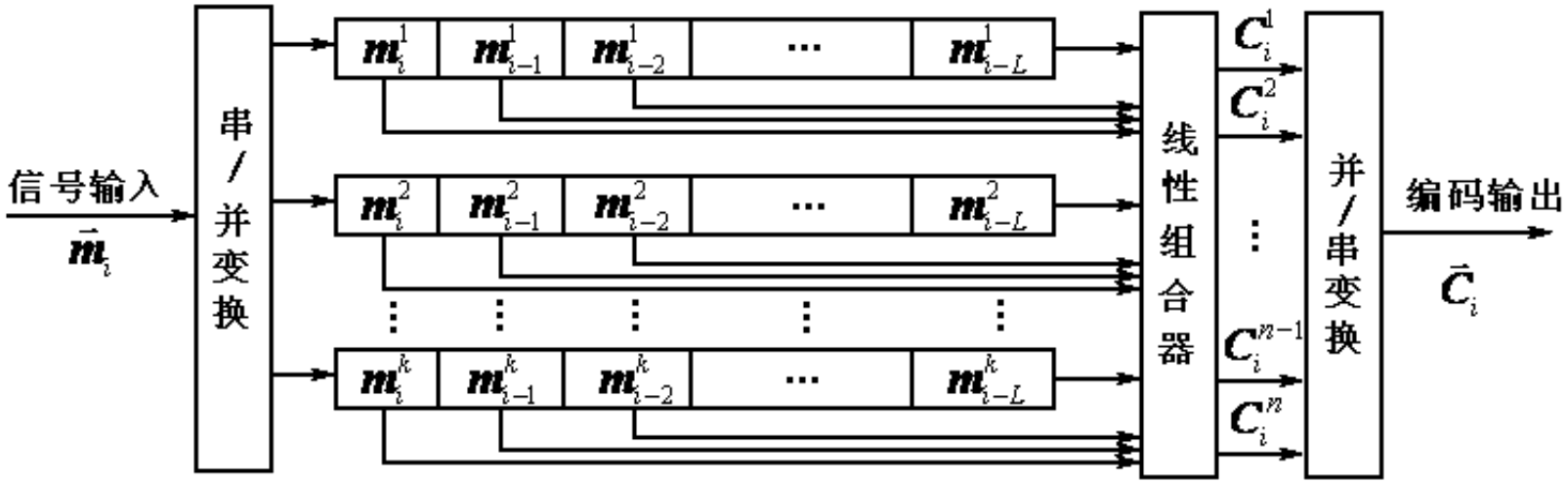


Figure: 编码器原理详图

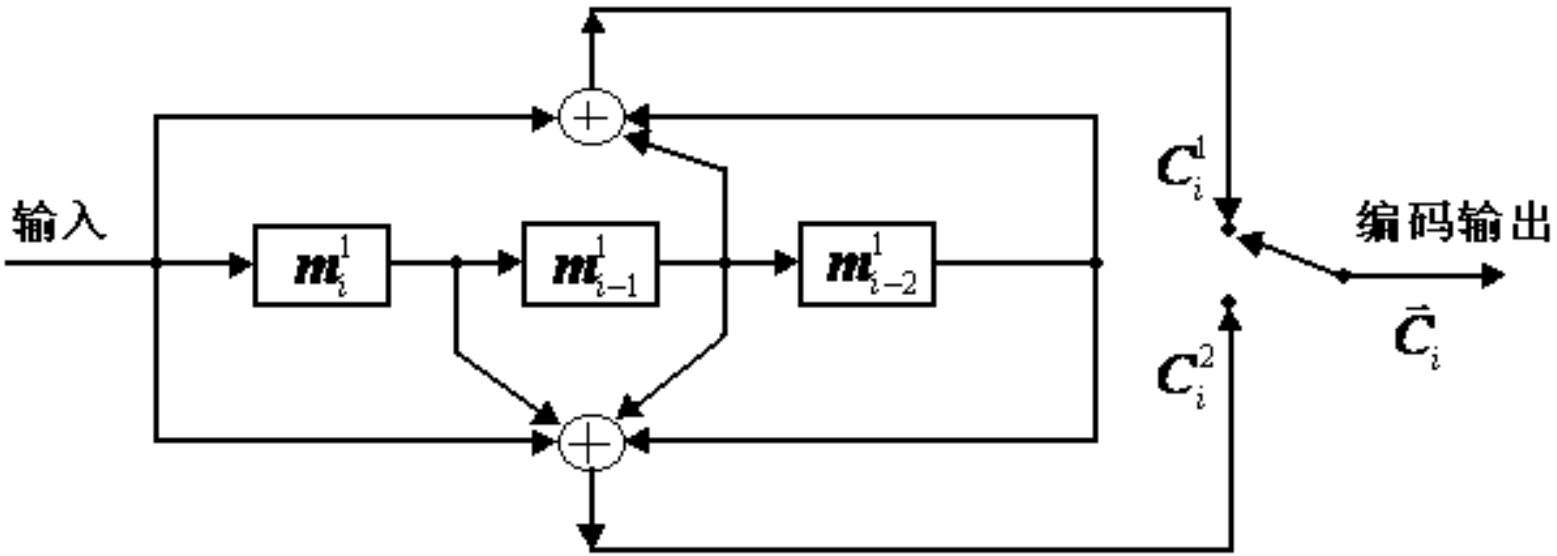
卷积码的描述方法

- 两大类：
  - 解析法：离散卷积，生成矩阵，码多项式等，用于编码的描述；
  - 图方法：状态图，树图法，网格法等，多用于译码的描述

例子问题描述

Example

例 1： 下图所示的二元卷积码的编码器结构，设输入信息为（1 0 1 1 1），求输出的编码后的序列。

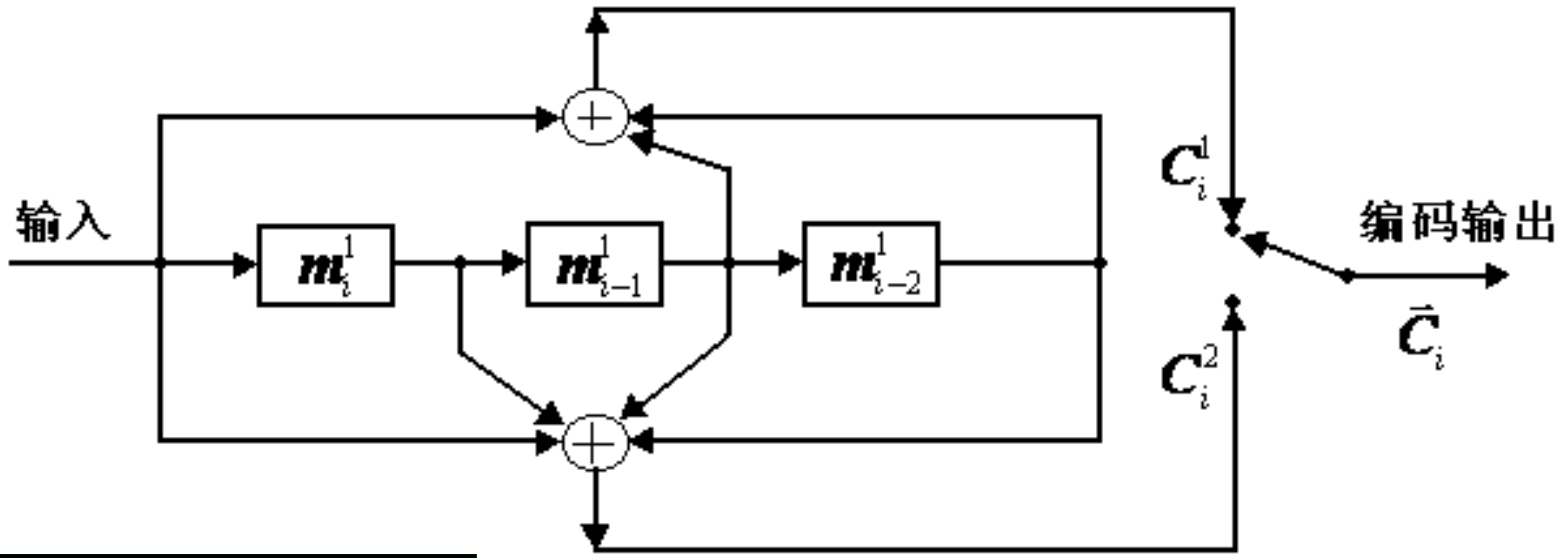


Figure： 二元卷积码的编码器例子

例子问题描述

Example

例 1： 下图所示的二元卷积码的编码器结构，设输入信息为（1 0 1 1 1），求输出的编码后的序列。



由图可知：k=1,n=2,L=3

Figure: 二元卷积码的编码器例子

离散卷积法

1. 第 i 时刻，输入编码器的信息是 m_{i+1}，移位寄存器内前三个时刻的数据 m_i, m_{i-1}, m_{i-2} 移动更新，输出此时的两个码元 c_i^1, c_i^2
   - $c_i^1 = m_{i+1} + m_{i-1} + m_{i-2}$
   - $c_i^2 = m_{i+1} + m_i + m_{i-1} + m_{i-2}$
2. 称 $\mathbf{c_i} = (c_i^1, c_i^2)$ 为子码，每个时刻输入 k(=1)个信息位，输出 n(=2)个码元，这 n 个码元组成卷积码的一个子码或码段或子组。
3. 写成矩阵形式有：$\mathbf{c_i} = [m_{i+1}\ \ m_i\ \ m_{i-1}\ \ m_{i-2}][g_1\ \ g_2]$，where $g_0 = [1\ \ 0\ \ 1\ \ 1]^t, g_2 = [1\ \ 1\ \ 1\ \ 1]^t$

离散卷积法

1. 第 i 时刻，输入编码器的信息是 m_{i+1}，移位寄存器内前三个时刻的数据 m_i, m_{i-1}, m_{i-2} 移动更新，输出此时的两个码元 c_i^1, c_i^2
   - $c_i^1 = m_{i+1} + m_{i-1} + m_{i-2}$
   - $c_i^2 = m_{i+1} + m_i + m_{i-1} + m_{i-2}$
2. 称 $\mathbf{c_i} = (c_i^1, c_i^2)$ 为子码，每个时刻输入 k(=1)个信息位，输出 n(=2)个码元，这 n 个码元组成卷积码的一个子码或码段或子组。
3. 写成矩阵形式有：$\mathbf{c_i} = [m_{i+1}\ \ m_i\ \ m_{i-1}\ \ m_{i-2}][g_1\ \ g_2]$，where $g_0 = [1\ \ 0\ \ 1\ \ 1]^t, g_2 = [1\ \ 1\ \ 1\ \ 1]^t$

g_0, g_1 仅仅与编码器的链接方式相关，即卷积码的编码器由上述编码方程的系数矩阵确定。

例子的输出

输入序列为 $m = (1\ 0\ 1\ 1\ 1)$

- 在 0 时刻之前，假设移位寄存器中全都存储 0；则有 0 时刻，输入 1，有输出子码：$\mathbf{c_0} = (1,1)$;
- 在 1 时刻，输入 0，则有子码：$\mathbf{c_1} = (0,1)$
- 在 2 时刻,. . .
- 假设时刻 4 之后，输入的全是 0

例子的输出

输入序列为 $m = (1\ 0\ 1\ 1\ 1)$

- 在 0 时刻之前，假设移位寄存器中全都存储 0；则有 0 时刻，输入 1，有输出子码：$\mathbf{c_0} = (1,1)$;
- 在 1 时刻，输入 0，则有子码：$\mathbf{c_1} = (0,1)$
- 在 2 时刻,. . .
- 假设时刻 4 之后，输入的全是 0

最后，得到编码输出：$c = (\mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c_7}) = (11,01,00,01,01,01,00,11)$

例子的输出

输入序列为 $m = (1\ 0\ 1\ 1\ 1)$

- 在 0 时刻之前，假设移位寄存器中全都存储 0；则有 0 时刻，输入 1，有输出子码：$\mathbf{c_0} = (1,1)$;
- 在 1 时刻，输入 0，则有子码：$\mathbf{c_1} = (0,1)$
- 在 2 时刻,. . .
- 假设时刻 4 之后，输入的全是 0

最后，得到编码输出：$c = (\mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c_7}) = (11,01,00,01,01,01,00,11)$

非系统前馈卷积码

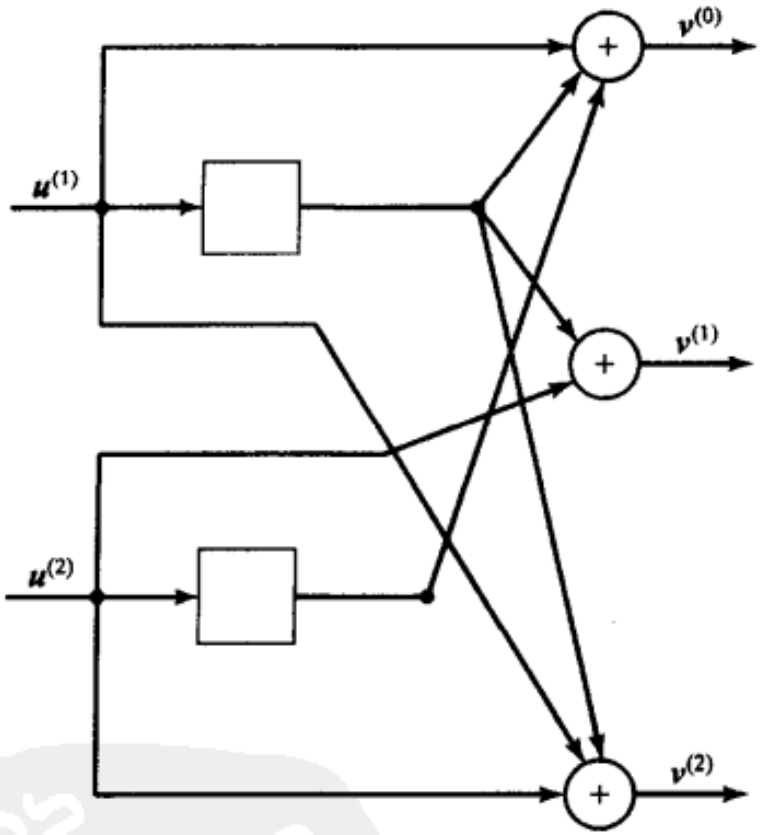
生成矩阵

- 将 g_0, g_1 交织，即各自轮流取 1 个分量排成一个 $2*L$ 的向量 g
- 我们称这个半无限的矩阵 **G** 为生成矩阵，**G** 的第 i 行，从第 i 列开始放置 g，本行其它位置全置 0

$$\mathbf{G} = \begin{pmatrix} g & 0 & 0 & 0 & \dots \\ 0 & g & 0 & 0 & \dots \\ 0 & 0 & g & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

- 输入序列也是无限的： $m = (m_0, m_1, m_2, \dots,)$
- 输出序列为：$c = m\mathbf{G}$

卷积码的第二个例子



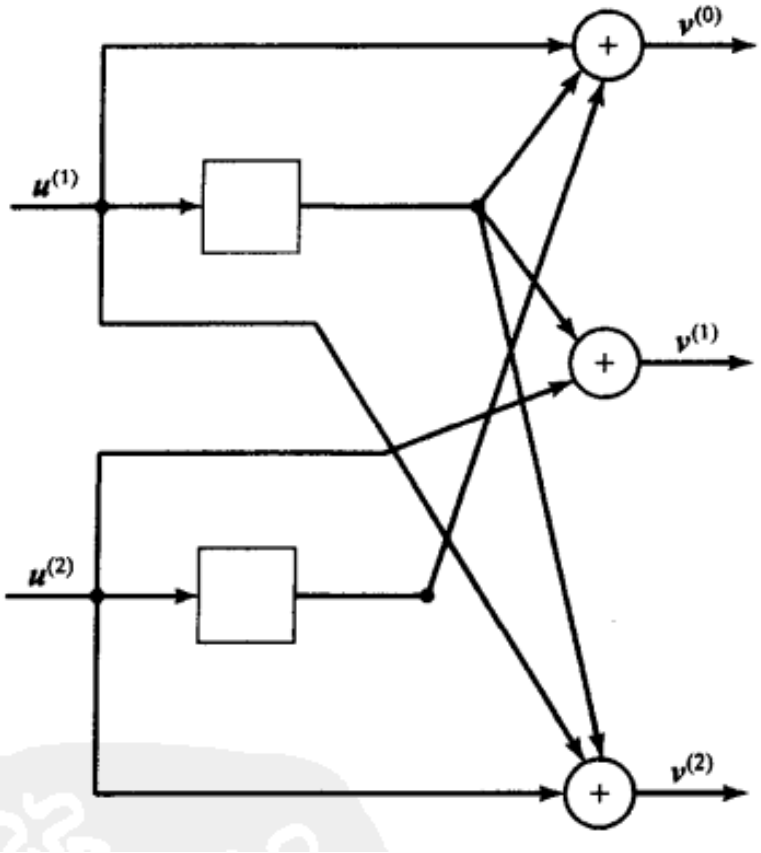
- 左图中 n=3,k=2,L=1，即(3,2,2)码
- 每个输出位的编码方程系数矩阵为：
-

$$g = \begin{pmatrix} 1\,0\,1\,1\,1\,1 \\ 0\,1\,1\,1\,0\,0 \end{pmatrix}$$

Figure: R=2/3 的非系统前馈卷积码编码器

卷积码的第一个例子

当输入(11,01,10)时，输出(110,000,001,111)



- 左图中 n=3,k=2,L=1，即(3,2,2)码
- 每个输出位的编码方程系数矩阵为：
-

$$g = \begin{pmatrix} 1\,0\,1\,1\,1\,1 \\ 0\,1\,1\,1\,0\,0 \end{pmatrix}$$

Figure: R=2/3 的非系统前馈卷积码编码器

卷积码的几个概念和定义

Definition

编码器存储级数 memory order，k 个输入信息位各对应一组移位寄存器，每组移位寄存器长度的最大值，记做 m

第一个例子中 $m = 3$，第二个例子中 $m = 1$

Definition

编码器整体约束长度 k 组移位寄存器的长度之和，也就是所有移位寄存器的总数，即做 ν

第一个例子中 $\nu = 3$，第二个例子中 $\nu = 2$

卷积码的第三个例子

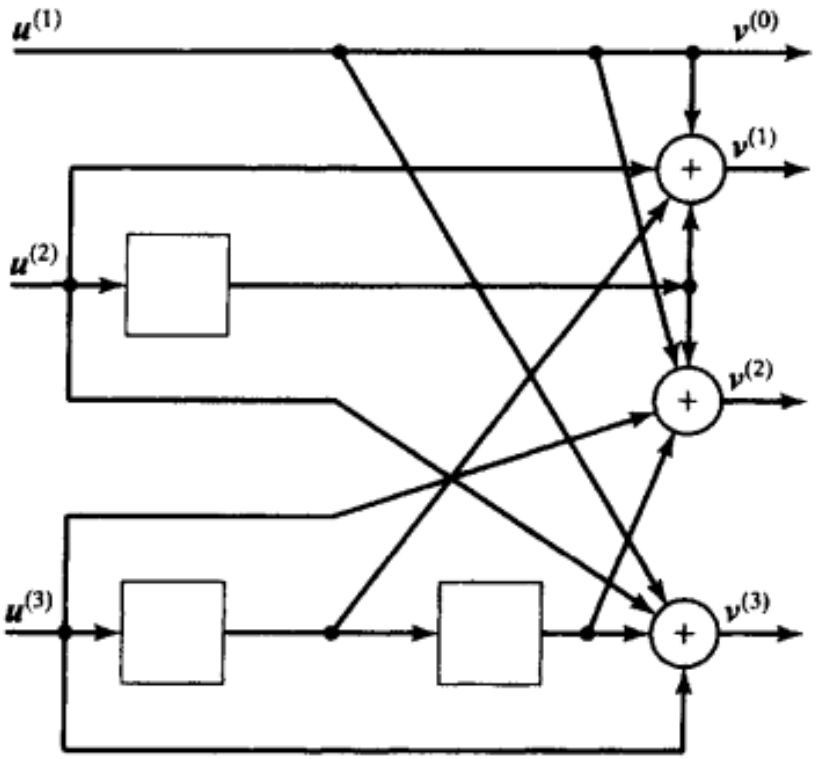


Figure: 卷积码的第三个例子

- 右图中 $m=2, k=3, n=4, v=3$，故这是一个 $(n,k,v)=(4,3,3)$ 码
- 各个输出计算方程的系数矩阵是什么？g 是怎么交织出来的？
- 卷积码是生成矩阵的线性组合，是线性码。

卷积码的多项式形式

- 在第一个例子中，两个生成多项式为：
  $g_0 = (1011) = 1 + x^2 + x^3, g_1 = (1111) = 1 + x + x^2 + x^3$
- 输入信息序列： $u = (10111) = 1 + x^2 + x^3 + x^4$
- 故有输出：
  $c^1 = ug_0 = (1 + x^2 + x^3 + x^4)(1 + x^2 + x^3) = 1 + x^7 = (1000001)$，$c^2 = ug_1 = (1 + x^2 + x^3 + x^4)(1 + x + x^2 + x^3) = 1 + x + x^3 + x^4 + x^5 + x^7 = (11011101)$
- 而输出序列：
  $(11010001010010011) = 1 + x + x^3 + x^7 + x^9 + x^{11} + x^{14} + x^{15}$
- 令 $\mathbf{G}(\mathbf{x}) = g_0(x^2) + xg_1(x^2) = (11011111)$，交织了 g_0, g_1 的系数.

卷积码的系统形式

Definition

卷积码的系统形式： k 个输入，n 个输出的前 k 个输出和 k 个输入一样。

- 前 k 个输出称为输出信息序列，后 n-k 个输出称为输出校验序列。
- 设 k 个输入的信息序列为 u，n 个输出信息序列为 v，则有 $v = u\mathbf{G}(\mathbf{D})$，若 $k * n$ 矩阵 $\mathbf{G}(\mathbf{D})$ 包含 k 阶的单位阵，则此生成矩阵 $\mathbf{G}$ 就是系统形式的，否则称为非系统的。

卷积码系统形式的例子

- 生成矩阵为：

$$\mathbf{G}(\mathbf{x}) = \begin{pmatrix} 1 & 0 & 1+x+x^2 \\ 0 & 1 & 1+x \end{pmatrix}$$

- x 的幂代表延时的单元数
- 其校验矩阵为：
  $H(x) = [1+x+x^2 \quad 1+x \quad 1]$

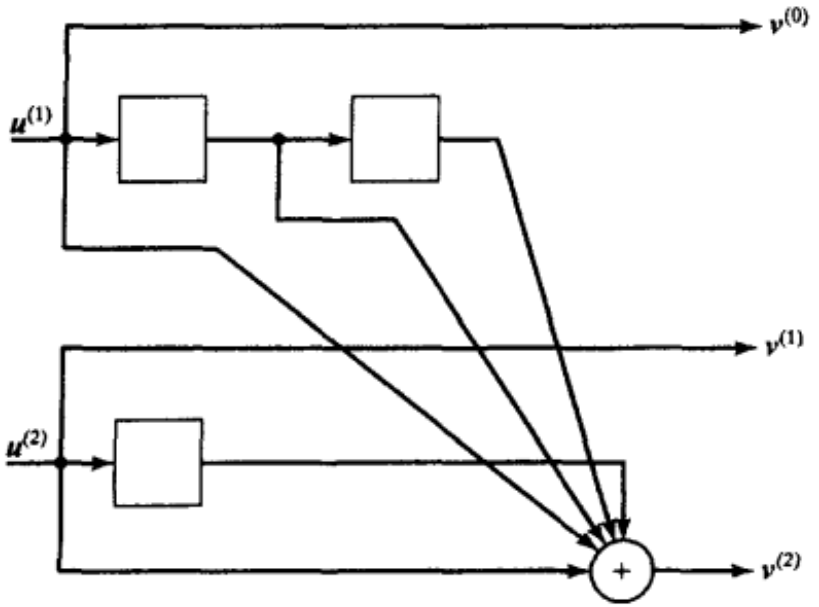


Figure: 系统形式的卷积码

编码器的状态数

- 移位寄存器的个数，即整体约束长度 v
- 状态数是：2^v（假设每个寄存器有两个状态）
- 如何找到状态数最小的编码器？
- 最大似然译码和最大后验概率译码都要求译码的复杂度和状态数成正比。

系统反馈卷积码编码器

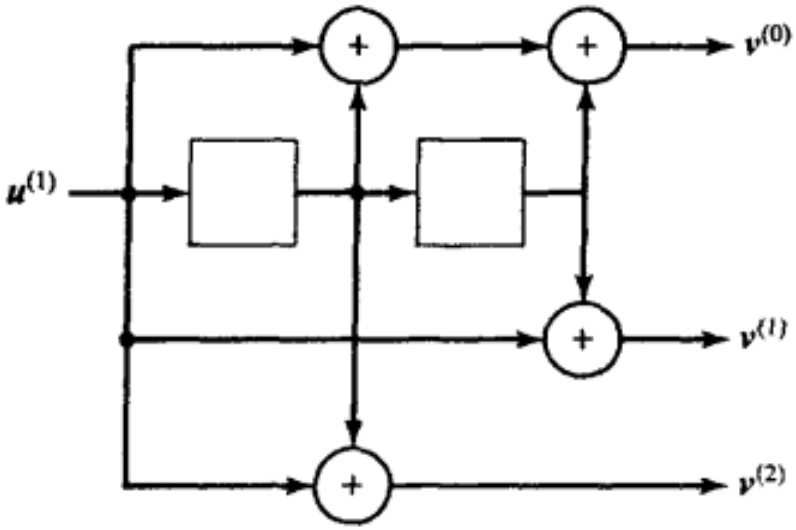


Figure: (3,1,2)非系统前馈卷积码编码器

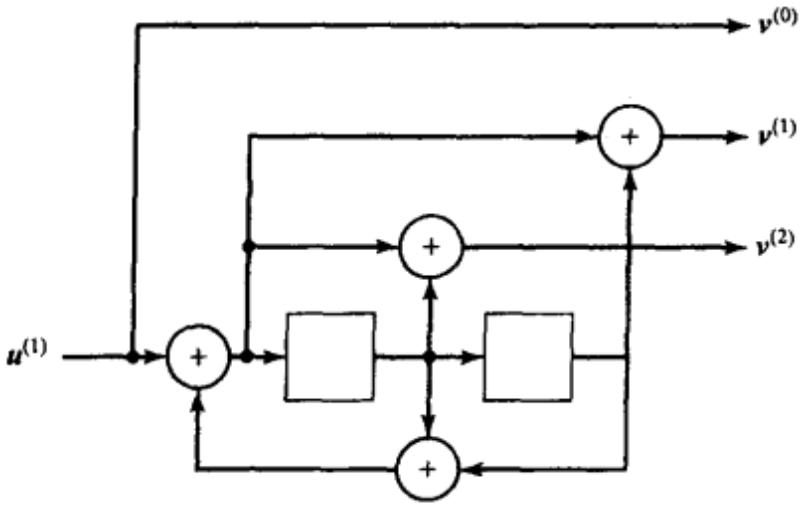


Figure: 等价的系统反馈卷积码编码器

- 左图生成矩阵：$\mathbf{G}(\mathbf{x}) = [1+x+x^2 \quad 1+x^2 \quad 1+x]$
- 变换 $\mathbf{G}(\mathbf{x})$，得到 $\mathbf{G}'(\mathbf{x}) = [1 \quad \frac{1+x^2}{1+x+x^2} \quad \frac{1+x}{1+x+x^2}]$，为右图生成矩阵

非系统卷积码变换成系统反馈卷积码的方法

- 设原非系统卷积码的生成矩阵为 **G**(**x**)；
- 通过矩阵处等变换，把 **G**(**x**) 前 k 列变换成单位阵，因为是初等变换，故得到的新生成矩阵和原来生成矩阵表示的码相同；
- 因变换后的生成矩阵 **G**′(**x**) 包含延时因子 x 的有理函数（分母是个 x 的多项式），故导致了反馈编码器。
- 分母多项式导致无穷项，单个输入会产生无穷个输出，所以上例也称为“递归卷积码”；
- 两个等价的前馈和反馈卷积码编码器，虽然编码后的码字集合是一样的，但是二者的消息序列到码字序列的映射是不同的。

卷积码的其它表示形式

多用于编码：

- 离散卷积法，方程组，卷积计算；
- 矩阵表示；
- 多项式表示法；

多用于译码：

- 状态图；
- 树图；
- 网格图。

状态图法

- 用一个有限状态机来描述：
- 图的每个结点是寄存器的一个状态，寄存器有 v 个，则状态数（图的结点数）为 2^v（假设为 2 进制信息序列）；
- 每一时刻有两个可能的输入（0 或者 1）将会替换寄存器中的某一个，因此可能的输出也是两个，下一状态也可能是两个；
- 图的边表示为“输出串/输入”或者“输出串(输入)”

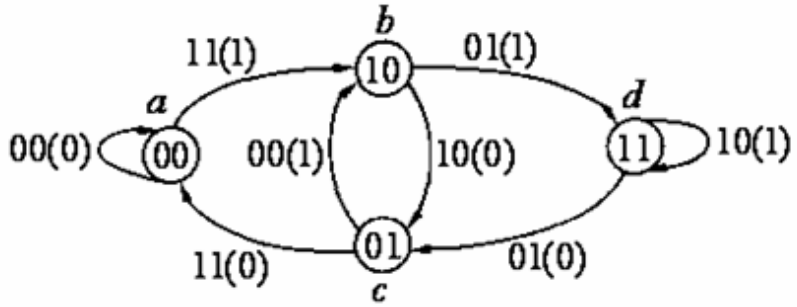


Figure: 第一个例子的状态图

状态图法

- 用一个有限状态机来描述：
- 图的每个结点是寄存器的一个状态，寄存器有 v 个，则状态数（图的结点数）为 2^v（假设为 2 进制信息序列）；
- 每一时刻有两个可能的输入（0 或者 1）将会替换寄存器中的某一个，因此可能的输出也是两个，下一状态也可能是两个；
- 图的边表示为“输出串/输入”或者“输出串(输入)”

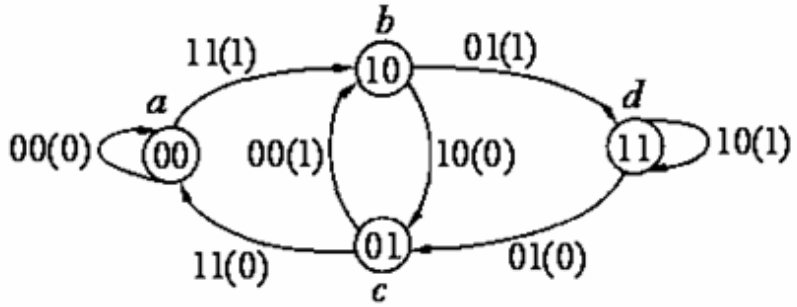


Figure: 第一个例子的状态图

图中 a,b,c,d 是 4 个状态的简记

树图法

- 初态：树根，初始结点，状态 00，结点处于时刻 $l'=0$；
- 每个结点代表一个状态，有两个输入，有两个输出，上是输入 0 的分支，边上权值是输出；下是输入 0 的分支。
- 随着时刻的推移，每个时刻的状态数呈指数增长
- 树图向右可无线延伸下去

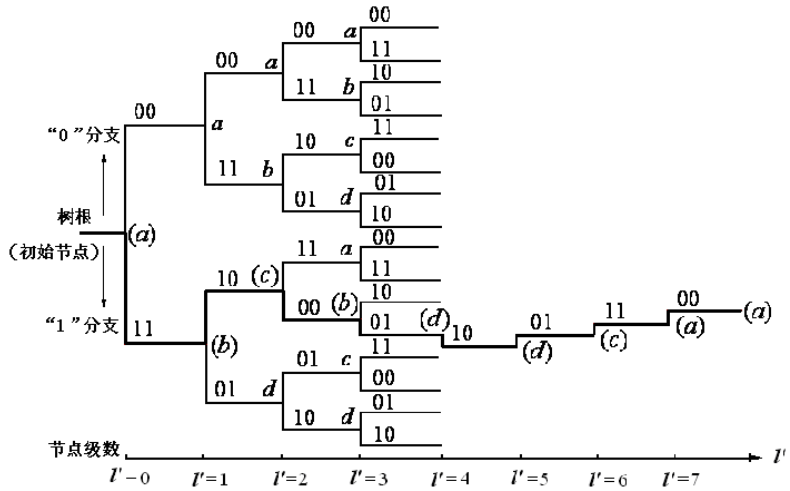


Figure: 第一个例子的树图

树图法的特点

- 按时间顺序展开状态；
- 对于一个给定的输入序列，在树图中存在唯一一条沿时间展开的路径对应；
- 缺点是结构复杂，结构重复性高；
- 上图中显示了输入为（10111000)时的路径，由图可以简单直观地给出状态迁移过程：abcbddcaa，以及编码序列：(1110000110011100)

网格图

- 又名格图，篱笆图等；
- 在概率译码中使用较多；现代编码技术常用的表示方法；
- Viterbi 译码算法的发明表明网格图的作用；
- 能很容易就从树图转换到网格图。

网格图的例子

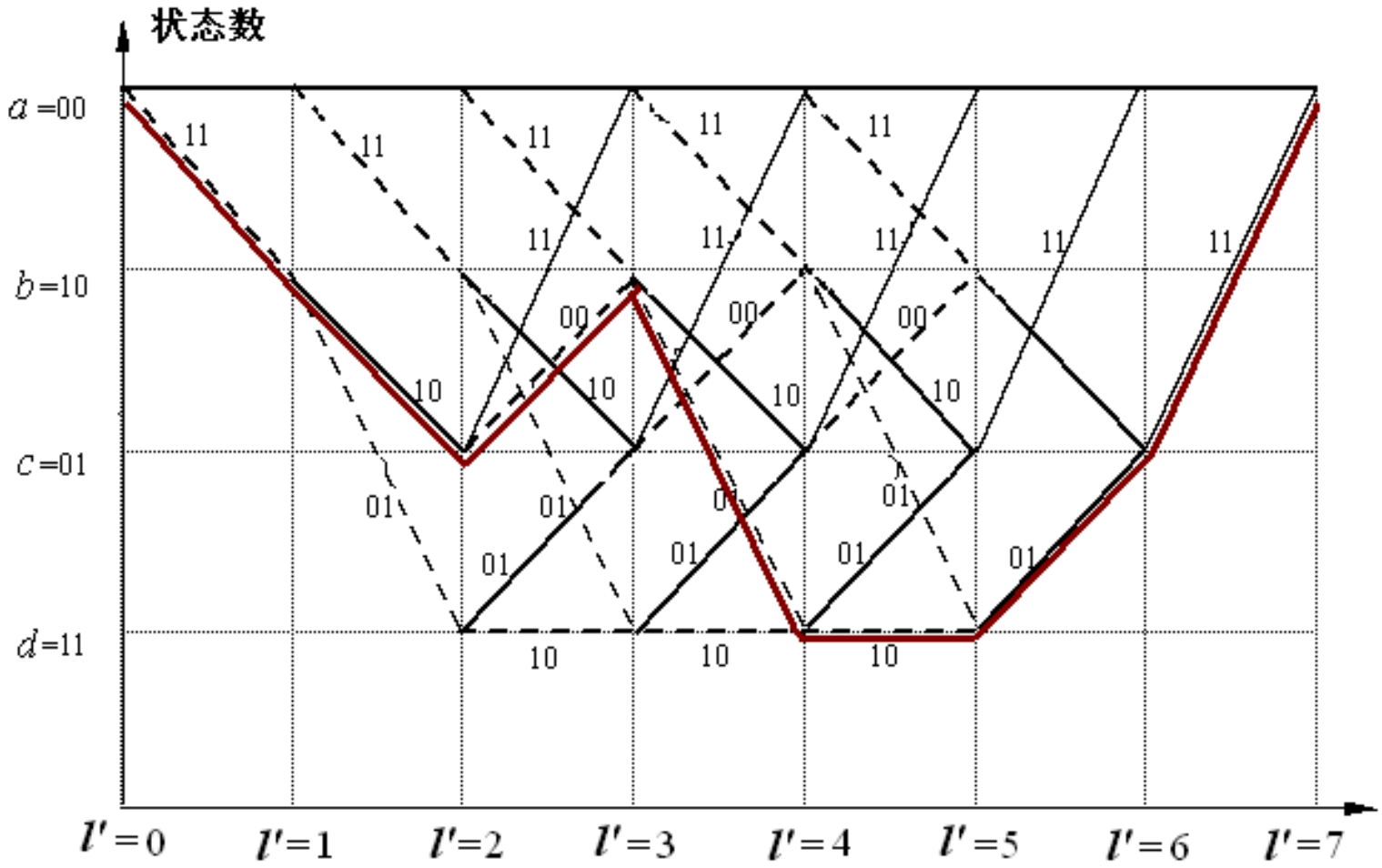


Figure: 网格图的例子

网格图构建方法

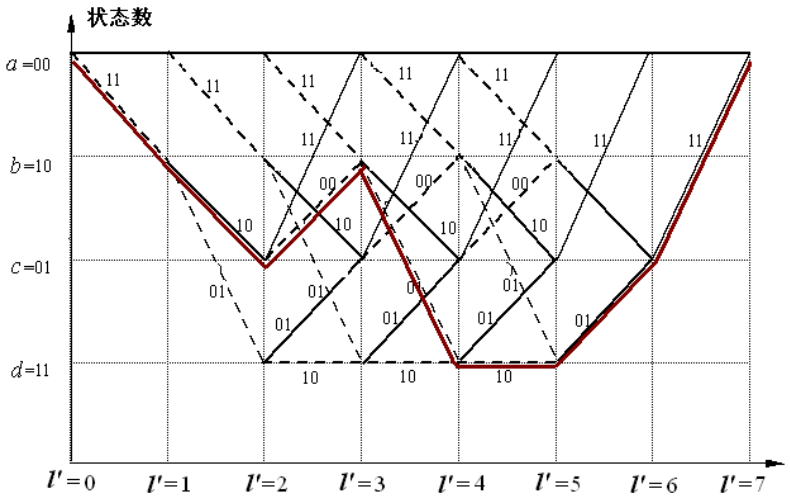


Figure: 网格图的构建

1. 横轴表示时刻，纵轴表示状态，虚线表示输入为 1 时走的分支，实现表示输入为 0 时走的分支
2. 就是合并了树图中的重复部分（忽略时刻）
3. 实线或虚线上标记的是输出序列。每个时刻，每个状态的每个输入及其输出和状态迁移都标记在图上。

网格图构建方法

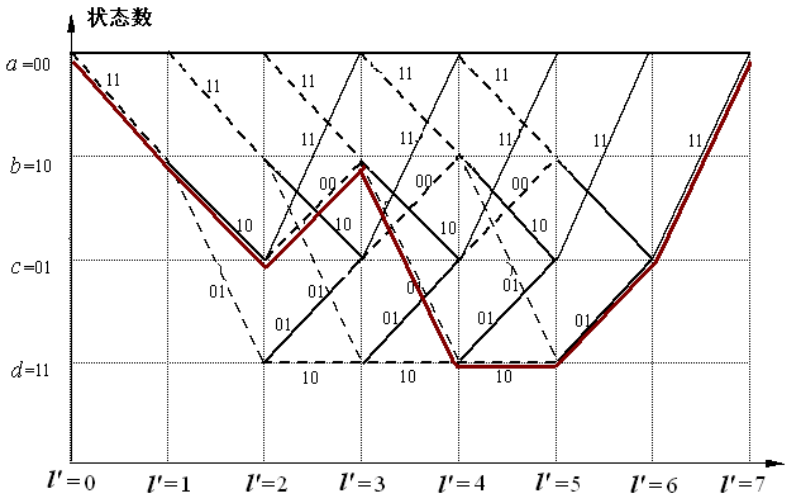


Figure: 网格图的构建

1. 横轴表示时刻，纵轴表示状态，虚线表示输入为 1 时走的分支，实现表示输入为 0 时走的分支
2. 就是合并了树图中的重复部分（忽略时刻）
3. 实线或虚线上标记的是输出序列。每个时刻，每个状态的每个输入及其输出和状态迁移都标记在图上。

- 当输入是(10111000)，图中表示为棕色的粗线路径
- 沿着棕线得到输出是 :(11100001100111)，状态迁移为：abcbddca

网格图的特点

- 不同的信息序列在网格图上对应不同的路径，即不完全重合；
- 在格图上，两个不同的信息序列的区分可以用它们不重合的部分来实现；
- 计算不相重合的路段来区分两个信息序列是译码的一个重要出发点，特别是 Viterbi 译码。

卷积码的译码

译码介绍

- 卷积码的译码可以分为两大类：
- 代数译码：解代数方程组，大数逻辑译码，门限译码
- 概率译码：软判决等，序列译码算法，维特比算法（最大似然译码）

维特比译码算法

- 1967 年，Viterbi 提出一种卷积码译码算法；
- 1969 年，Omura 证明维特比译码算法等价于通过一个加权图求最短路径问题的动态规划；
- 1973 年，G.D.Forney 指出维特比译码算法就是最大似然译码，即选择的码字使得对数似然函数值最大；
- 二进制对称信道条件下，维特比译码算法就是最小距离译码。

维特比译码算法：思想

- 在不同时刻：$l' = L, L = L+1, L+2, \ldots, L+l-1$，考虑每个状态，即同列的所有状态点，编码器当前时刻的状态，按照最大似然准则比较所有以它为终点的路径，仅仅保留具有最大似然值的路径，并称这条路经为“幸存路径”；其他路径被堵住，弃之不用。
- 下一时刻，仅对幸存路径的延伸出来的路径进行比较；获得更长的幸存路径；
- 如此继续，接收一段，计算一段，保留一段（幸存路径），如此反复，直到最后；
- 在最后时刻获得的一条幸存路径就是所要求的最大似然译码的解。

维特比译码算法的优点

- 路径度量是可加的，格图的格子结构，使得每次局部判断都会获得全局最优的一部分；
- 局部最优性的判决，及时去掉了大量的非优路径，不让其进入下一时刻的延伸，而且如果有重复部分，重复部分可以不计算，只需要比较开始分离的不同路段的值即可，节约了计算量；
- 算法简单，容易实现。

维特比算法步骤

1. 从 $l' = L$ 时刻开始，计算进入每一状态的单个路径的部分度量值，并存储每一状态下的幸存路径及其度量值；
2. $l' = l' + 1$，将进入每一状态的分支度量值与前一时间段的幸存度量值相加，然后计算进入该状态的所有最大度量的路径或最小海明距路径（幸存路径）及其度量，并删除所有其他路径；
3. 若 $l' < L + l$，重复第二步，否则停止

维特比算法步骤

1. 从 $l' = L$ 时刻开始，计算进入每一状态的单个路径的部分度量值，并存储每一状态下的幸存路径及其度量值；
2. $l' = l' + 1$，将进入每一状态的分支度量值与前一时间段的幸存度量值相加，然后计算进入该状态的所有最大度量的路径或最小海明距路径（幸存路径）及其度量，并删除所有其他路径；
3. 若 $l' < L + l$，重复第二步，否则停止

第二步是关键，包括两个部分：计算和比较度量值；保存幸存路径

维特比译码算法：几个概念

Definition

似然函数： 设发送序列为 v，接受序列为 r，则似然函数定义为 $p(r|v)$，对数似然函数定义为：$\ln p(r|v)$

维特比译码算法：几个概念

Definition

似然函数： 设发送序列为 v，接受序列为 r，则似然函数定义为 $p(r|v)$，对数似然函数定义为：$\ln p(r|v)$

- 对于离散无记忆信道，似然函数有：$p(r|v)=\prod_{i=0}^{n-1}p(r_i|v_i)$
- 对应对数似然函数有：$\ln p(r|v)=\sum_{i=0}^{n-1}\ln p(r_i|v_i)$
- 最大似然译码要求在所有可选的码字序列中，选择使得上式的似然函数值最大的码字序列，作为发送序列（译码结果）
- 网格图上找一条使得似然函数值最大的路径！

维特比码译码算法：几个概念

Definition

路径度量： 网格上的路径 v 的路径度量定义为：$\ln p(r|v)$，记为 $\lambda(r|c)$，分支 v_i 的度量定义为：$\ln p(r_i|v_i)$，记为 $\lambda(r_i|v_i)$.

- 利用路径度量和分支度量表示方法，可以把似然函数改写成 $\lambda(r|v) = \sum_{i=0}^{n-1} p(r_i|v_i)$
- 从上式可以知道：一条路径的路径度量是该路径上各分支度量之和。

维特比译码算法：几个概念

Definition

部分路径度量： 网格图上的一条路径 v 的前 l 个分支所构成的称为部分路径，其度量称为部分路径度量，记为 $\lambda((r|v)_0^{l-1}) = \sum_{i=0}^{l-1} \lambda(r_i|v_i)$.

维特比算法例子

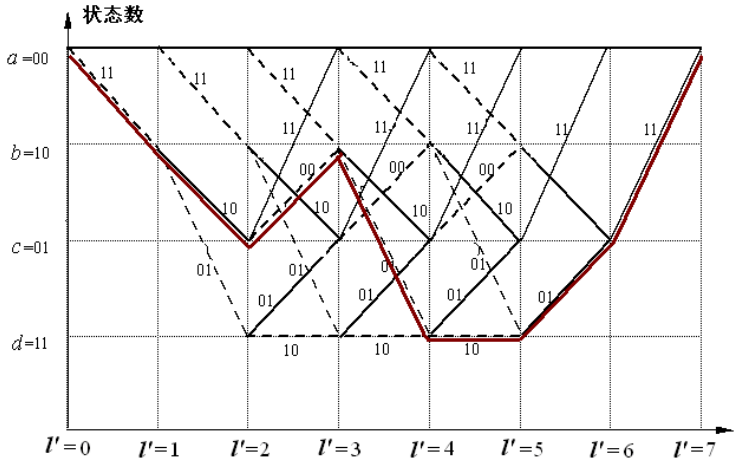


Figure: 网格图

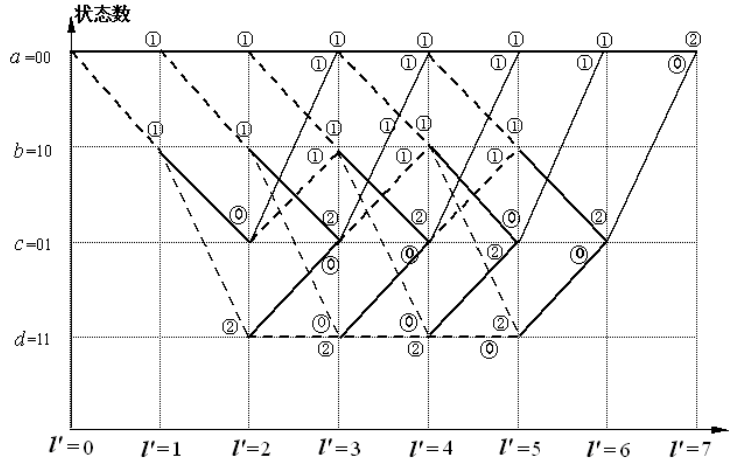


Figure: 带分支度量的网格图

维特比算法例子

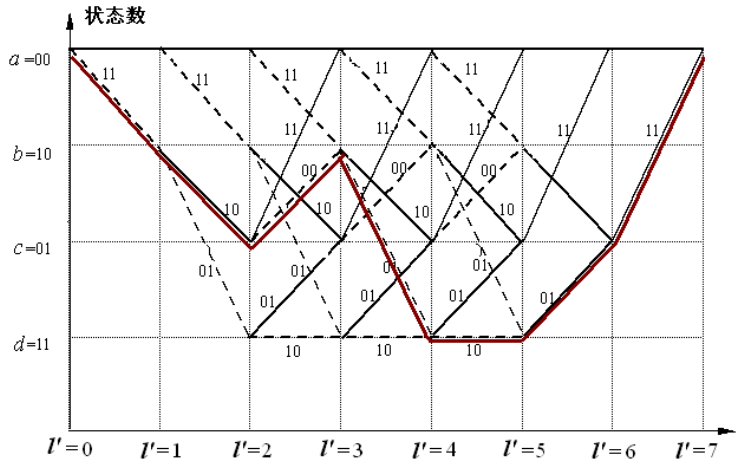


Figure: 网格图

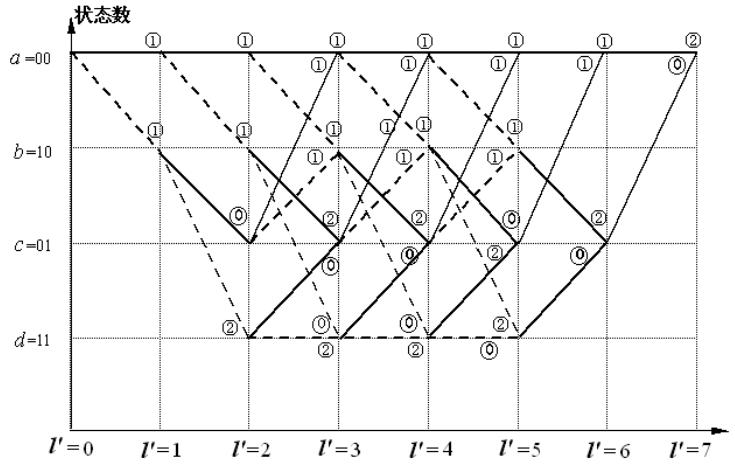


Figure: 带分支度量的网格图

- BSC 上，分支度量就是海明距，这是整数度量；还可以是实数值，对应无量化的软判决译码。
- 假设发送信息序列为(10111)，故得编码器的编码输出为(11,10,00,01,10,01,11)，而接收序列是(10,01,01,01,10,01,11)，要求从网格图中找最大似然路径。

维特比算法例子

根据左图和接受序列能得到右图

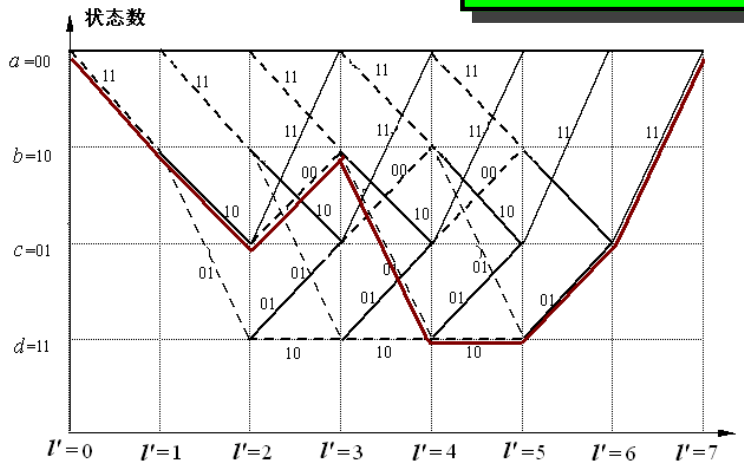


Figure: 网格图

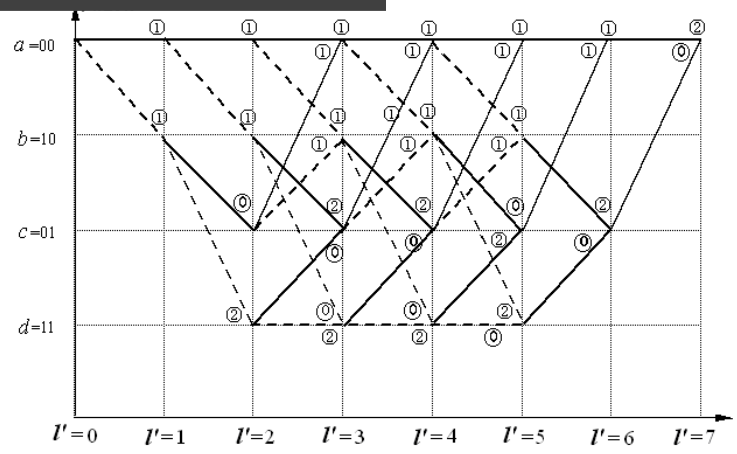


Figure: 带分支度量的网格图

- BSC 上，分支度量就是海明距，这是整数度量；还可以是实数值，对应无量化的软判决译码。
- 假设发送信息序列为(10111)，故得编码器的编码输出为(11,10,00,01,10,01,11)，而接收序列是(10,01,01,01,10,01,11)，要求从网格图中找最大似然路径。

维特比算法的实现

- 译码器存储器的长度，对编码器中每个状态都要有个存储器来存储该状态对应的幸存路径及其度量值，而编码器中寄存器的大小（整体约束长度）决定了，编码器中寄存器的个数不能太多，比如小于 10；
- 延时问题或者说路径的截断问题，一条路径完全判决了才能输出译码，路径的截断？同步？
- 软判决的问题及其性能的提高。
- 状态数少，意味着 n 不能很大，也就是码字间距不会很大，进而推出纠错能力受限。

译码性能

维特比译码算法的性能

Definition

首次错误译码概率： BSC 条件下，维特比译码器在 j 时刻之前均译码正确，而在 j 时刻发生首次错误译码，这种事件的发生概率称为首次错误译码概率，记做 $p_{fe,j}$

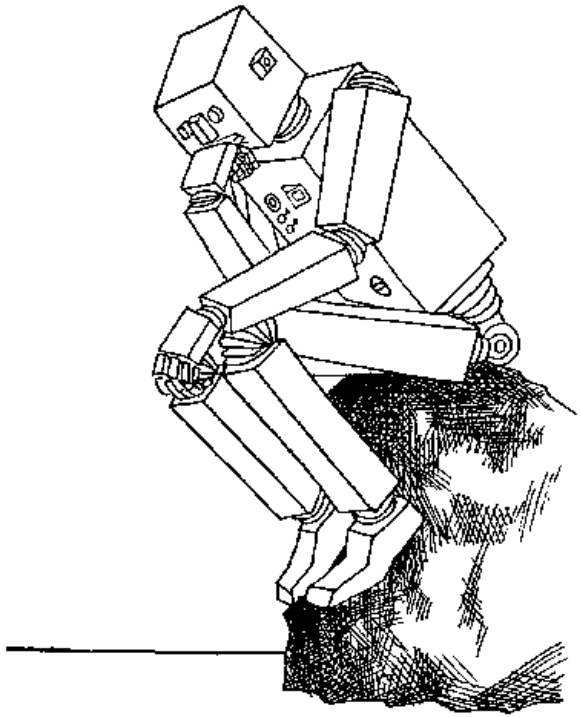
- $p_{fe,j} = p_{fe}$，也就是首次错误译码的概率和时刻无关；任何时刻都是一样的；
- 而译码错误概率 $p_e \leq p_{fe}$

适合维特比译码的卷积码

- 影响无码率的因素，主要是自由距离 d_{free}，自由距离增加，误码率指数下降；
- 还有其它的约束条件限制了采用维特比译码算法；
- 找一个卷积码，使得维特比译码算法适用，在理论不完善的条件下，这依然是个待解决的难题，现在一般用计算机来搜索具有良好性能的好码。

谢　谢！

Thank you very much for your kind attention.



Any Questions?

References

References I