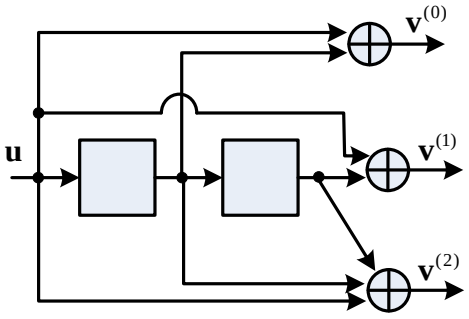
第五章 卷积码的译码算法

卷积编码器自身具有网格结构，基于此结构我们给出两种译码算法：Viterbi 译码算法和 BCJR 译码算法。基于某种准则，这两种算法都是最优的。1967 年，Viterbi 提出了卷积码的 Viterbi 译码算法，后来 Omura 证明 Viterbi 译码算法等效于在加权图中寻找最优路径问题的一个动态规划（Dynamic Programming）解决方案，随后，Forney 证明它实际上是最大似然（ML，Maximum Likelihood）译码算法，即译码器选择输出的码字通常使接收序列的条件概率最大化。BCJR 算法是 1974 年提出的，它实际上是最大后验概率（MAP，Maximum A Posteriori probability）译码算法。这两种算法的最优化目标略有不同：在 MAP 译码算法中，信息比特错误概率是最小的，而在 ML 译码算法中，码字错误概率是最小的，但两种译码算法的性能在本质上是相同的。由于 Viterbi 算法实现更简单，因此在实际应用比较广泛，但在迭代译码应用中，例如逼近 Shannon 限的 Turbo 码，常使用 BCJR 算法。另外，在迭代译码应用中，还有一种 Viterbi 算法的变种：软输出 Viterbi 算法（SOVA，Soft-Output Viterbi Algorithm），它是 Hagenauer 和 Hoeher 在 1989 年提出的。

5.1 Viterbi 算法

为了理解 Viterbi 译码算法，我们需要将编码器状态图按时间展开（因为状态图不能反映出时间变化情况），即在每个时间单元用一个分隔开的状态图来表示。例如（3，1，2）非系统前馈编码器，其生成矩阵为：

$$\mathbf{G}(D)=\left[1+D \quad 1+D^2 \quad 1+D+D^2\right] \qquad (5.1)$$



（a）

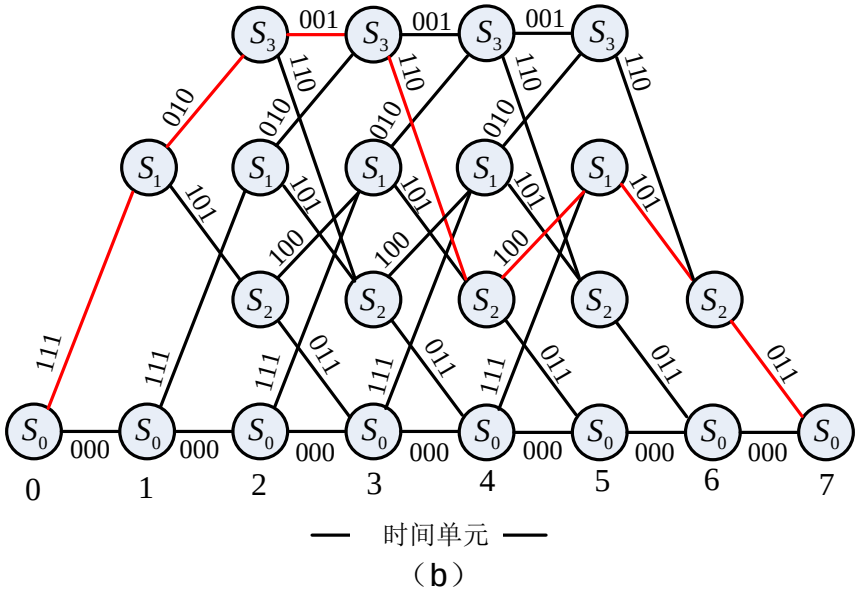


图 5.1 （a）（3，1，2）编码器 （b）网格图（h＝5）

假定信息序列长度为 h＝5，则网格图包含有 h＋m＋1＝8 个时间单元，用 0 到 h＋m＝7 来标识，如图 5.1（b）所示。假设编码器总是从全 0 态 S_0 开始，又回到全 0 态，前 m＝2 个时间单元对应于编码器开始从 S_0 “启程”，最后 m＝2 个时间单元对应于向 S_0 “返航”。从图中我们也可以看到，在前 m 个时间单元或最后 m 个时间单元，并不是所有状态都会出现，但在网格图的中央部分，在每个时间单元都会包含所有状态，且在每个状态都有 $2^k=2$ 个分支离开和到达。离开每个状态的上面分支表示输入比特为 1（即 $u_i=1$，i 表示第 i 个时间单元），下面的分支表示输入比特为 0。每个分支的输出 $\mathbf{v}_i$ 由 n 个比特组成，共有 $2^h=32$ 个码字，每个码字都可用网格图中的唯一路径表示，码字长度 N＝n(h＋m)＝21。例如当信息序列为 **u**＝（11101）时，对应的码字如图 5.1（b）中红线所示，**v**＝（111，010，001，110，100，101，011）。在一般的（n，k，v）编码器情况下，信息序列长度 K*=kh，离开和进入每个状态都有 2^k 个分支，有 2^{K^*} 个不同路径通过网格图，对应着 2^{K^*} 个码字。

假设长度 $K^*=kh$ 的信息序列 $\mathbf{u}=(\mathbf{u}_0,\mathbf{u}_1\cdots\mathbf{u}_{h-1})$ 被编码成长度为 $N=n(h+m)$ 的码字 $\mathbf{v}=(\mathbf{v}_0,\mathbf{v}_1\cdots\mathbf{v}_{h+m-1})$，在经过一个二进制输入、Q-ary 输出的离散无记忆信道（DMC，Discrete memoryless Channel）后，接收序列为 $\mathbf{r}=(\mathbf{r}_0,\mathbf{r}_1\cdots\mathbf{r}_{h+m-1})$。也可表示为：$\mathbf{u}=(u_0,u_1\cdots u_{K^*-1})$，$\mathbf{v}=(v_0,v_1\cdots\mathbf{v}_{N-1})$，$\mathbf{r}=(r_0,r_1\cdots r_{N-1})$，译码器对接收到的序列 **r** 进行处理，得到 **v** 的估计 $\hat{\mathbf{v}}$。在离散无记忆信道情况下，最大似然译码器是按照最大化对数似然函数 $\log P(\mathbf{r}\mid\mathbf{v})$ 作为选择 $\hat{\mathbf{v}}$ 的准则。因为对于 DMC，

$$P(\mathbf{r}\mid\mathbf{v})=\prod_{l=0}^{h+m-1}P(\mathbf{r}_l\mid\mathbf{v}_l)=\prod_{l=0}^{N-1}P(r_l\mid v_l) \qquad (5.2)$$

两边取对数后为：

$$\log P(\mathbf{r}\mid\mathbf{v})=\sum_{l=0}^{h+m-1}\log P(\mathbf{r}_l\mid\mathbf{v}_l)=\sum_{l=0}^{N-1}\log P(r_l\mid v_l) \qquad (5.3)$$

其中 $P(r_l\mid v_l)$ 是信道转移概率，当所有码字等概时，这是个最小错误概率译码准则。

对数似然函数 $\log P(\mathbf{r}\mid\mathbf{v})$，用 $M(\mathbf{r}\mid\mathbf{v})$ 表示，称为路径度量（path metric）；$\log P(\mathbf{r}_l\mid\mathbf{v}_l)$，称为分支度量（branch metric），用 $M(\mathbf{r}_l\mid\mathbf{v}_l)$ 表示；$\log P(r_l\mid v_l)$ 称为比特度量（bit metric），用 $M(r_l\mid v_l)$ 表示，这样（5.3）式可写为：

$$M(\mathbf{r}\mid\mathbf{v})=\sum_{l=0}^{h+m-1}M(\mathbf{r}_l\mid\mathbf{v}_l)=\sum_{l=0}^{N-1}M(r_l\mid v_l) \qquad (5.4)$$

如果我们只考虑前 t 个分支，则部分路径度量可表示为：

$$M(\mathbf{r}\mid\mathbf{v})=\sum_{l=0}^{t-1}M(\mathbf{r}_l\mid\mathbf{v}_l)=\sum_{l=0}^{nt-1}M(r_l\mid v_l) \qquad (5.5)$$

对于接收序列 **r**，Viterbi 算法就是通过网格图找到具有最大度量的路径，即最大似然路径（码字）。在每个时间单元的每个状态，都增加 2^k 个分支度量到以前存储的路径度量中（加）；然后对进入每个状态的所有 2^k 个路径度量进行比较（比），选择具有最大度量的路径（选），最后存储每个状态的幸存路径及其度量。

Viterbi 算法：

Step 1: 在 t＝m 时间单元开始，计算进入每个状态的单个路径的部分度量，存储每个状态的路径（幸存）及其度量；

Step 2: t←t＋1，对进入每个状态的所有 2^k 个路径计算部分度量，并加上前一时间单元的度量。对于每个状态，比较进入该状态的所有 2^k 个路径度量，选择具有最大度量的路径，存储其度量，并删掉其他路径。

Step 3: 如果 t<h+m，返回 step 2；否则，就停止。

Viterbi 算法的基本计算“加、比、选”体现在 step 2。注：实际工程中，在每个状态存储（在 step 1 和 step 2）的是对应于幸存路径的信息序列，而不是幸存路径自身，这样当算法结束时，就无需再通过估计码字 $\hat{\mathbf{v}}$ 来恢复信息序列 $\hat{\mathbf{u}}$。

从时间单元 m 到 h，有 2^ν 个幸存路径，每个状态（共有 2^ν 个状态）一个。随后，幸存路径数就会变少，因为当编码器回到全 0 态时，状态数就会变少。最后，在时间单元 h＋m，就只有一个状态（即全 0 态），因此，也就只有一个幸存路径了，算法中止。

定理 5.1：在 Viterbi 算法中最后的幸存路径 $\hat{\mathbf{v}}$ 是最大似然路径，即

$$M(\mathbf{r}\mid\hat{\mathbf{v}})\geq M(\mathbf{r}\mid\mathbf{v}),\quad for\ \mathbf{v}\neq\mathbf{v} \qquad (5.6)$$

从实现的角度看，用正整数度量来表示要比用实际的比特度量表示更方便。比特度量 $M(r_l\mid v_l)=\log P(r_l\mid v_l)$ 可用 $c_2\left[\log P(r_l\mid v_l)+c_1\right]$ 来代替，其中 c_1 是任意实数，c_2 是任意正实数。可证明，如果路径 **v** 最大化 $M(\mathbf{r}\mid\mathbf{v})=\sum_{l=0}^{N-1}M(r_l\mid v_l)=\sum_{l=0}^{N-1}\log P(r_l\mid v_l)$，则

它也最大化 $c_2\left[\log P(r_l \mid v_l)+c_1\right]$，因此可以使用修正的度量，且不影响 Viterbi 算法的性能。

如果选择 c_1 使最小度量为 0，则 c_2 可选择为使所有度量近似为整数。这样，由于用整数来近似表示度量，Viterbi 算法的性能变成了次最优算法，但通过选择 c_1 和 c_2 可使得这种性能降低非常小。

＝＝

例 5.1：对于二输入、4-ary 输出的 DMC 信道下的 Viterbi 算法

二输入、4-ary 输出的 DMC 如图 5.2 所示。该信道的比特度量如图 5.3（a）所示（按照底为 10 的对数计算），选择 c_1＝1，c_2＝17.3，得到整数度量表如图 5.3（b）所示。

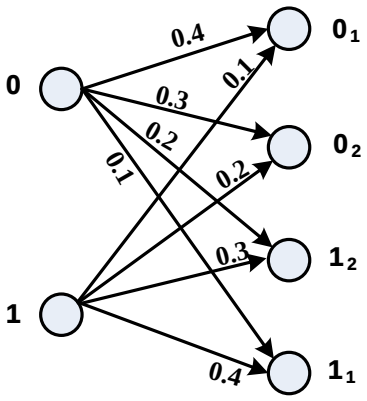


图 5.2 二输入、4-ary 输出 DMC 信道模型

v_l \ r_l	0_1	0_2	1_2	1_1
0	-0.4	-0.52	-0.7	-1.0
1	-1.0	-0.7	-0.52	-0.4

(a)

v_l \ r_l	0_1	0_2	1_2	1_1
0	10	8	5	0
1	0	5	8	10

(b)

图 5.3 度量表

假设图 5.1 中的一个码字在这样的信道中传输，接收到的序列为：

$$\mathbf{r}=(1_1 1_2 0_1,\ 1_1 1_1 0_2,\ 1_1 1_1 0_1,\ 1_1 1_1 1_1,\ 0_1 1_2 0_1,\ 1_2 0_2 1_1,\ 1_2 0_1 1_1) \qquad (5.7)$$

对该序列进行 Viterbi 译码如图 5.4 所示。

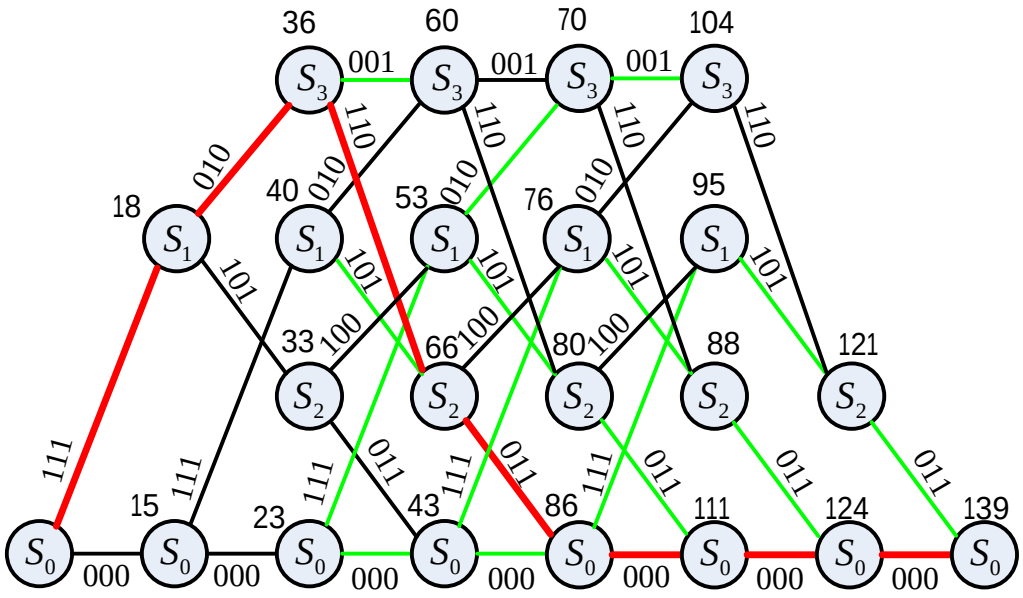


图 5.4 DMC 信道下的 Viterbi 算法

每个状态上的数字表示幸存路径的度量，另一个路径就将被删除（绿线部分）。这样最后的码字（红线部分的输出）判决为：

$$\hat{\mathbf{v}} = (111,010,110,011,000,000,000) \qquad (5.8)$$

它对应的输入序列为$\hat{\mathbf{u}} = (11000)$。注意：网格图中最后的 m＝2 个分支是清空寄存器的，不能算作为输入信息序列。

＝＝

在 BSC 信道情况下，转移概率为 p<1/2，接收序列 **r** 是 2-ary 输出的，此时对数似然函数为：

$$\log P(\mathbf{r}\,|\,\mathbf{v}) = d(\mathbf{r},\mathbf{v})\log\frac{p}{1-p} + N\log(1-p) \qquad (5.9)$$

其中$d(\mathbf{r},\mathbf{v})$是 **r** 和 **v** 之间的 Hamming 距离。由于$\log\frac{p}{1-p} < 0$，且$N\log(1-p)$对所有 **v** 来说都是一个常数，因此最大似然译码（$\max\log P(\mathbf{r}\,|\,\mathbf{v})$）就是最小化 Hamming 距离（$\min d(\mathbf{r},\mathbf{v})$）。

$$d(\mathbf{r},\mathbf{v}) = \sum_{l=0}^{h+m-1} d(\mathbf{r}_l,\mathbf{v}_l) = \sum_{l=0}^{N-1} d(r_l,v_l) \qquad (5.10)$$

因此，当我们将 Viterbi 算法应用到 BSC 信道时，$d(\mathbf{r}_l,\mathbf{v}_l)$成为分支度量，$d(r_l,v_l)$为

比特度量，该算法就是寻找具有最小度量的路径，即与 **r** 汉明距离最近的路径。该算法运算仍然相同，只是用 Hamming 距离代替了似然函数作为度量，在每个状态的幸存路径是具有最小度量的路径。

＝＝＝

例 5.2：BSC 信道下的 Viterbi 算法

假设接收到的序列为

$$\mathbf{r} = (110, 110, 110, 111, 010, 101, 101) \qquad (5.10)$$

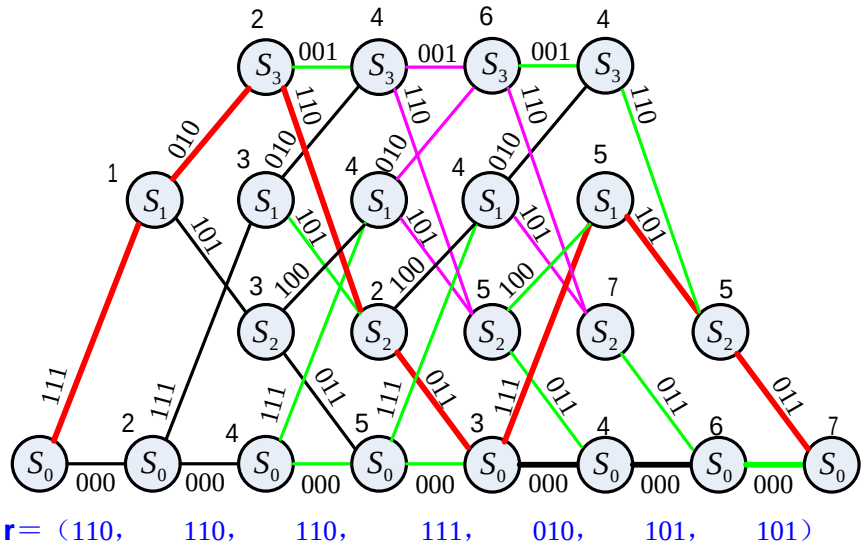


图 5.5 BSC 信道下的 Viterbi 算法

最后的码字判决为：

$$\hat{\mathbf{v}} = (111, 010, 110, 011, 111, 101, 011) \qquad (5.11)$$

它所对应的信息序列为 $\hat{\mathbf{u}} = (11001)$。

最后的幸存路径度量值为 7，表示接收序列和判决码字之间有 7 个对应位置不同，而其他路径所对应的码字和接收序列之间的位置不同数目都要大于 7。在上图中紫色对应的线表示两条路径度量值相同，此时随便选其中一条就 OK 了。

＝＝＝

现在考虑二进制输入的 AWGN 信道，解调器输出不进行量化，即二值输入、连续输出信道。假设信道输入 0 和 1 用 BPSK 信号 $\pm\sqrt{\frac{2E_s}{T}}\cos(2\pi f_0 t)$ 表示，其中映射关系 $1 \to +\sqrt{E_s}$，$0 \to -\sqrt{E_s}$。考虑码字 $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1})$，按照映射关系 $1 \to +1$ 和 $0 \to -1$ 进行取值，即 ± 1，归一化（用 $\sqrt{E_s}$ 进行归一化）的接收序列 $\mathbf{r} = (r_0, r_1, \cdots, r_{N-1})$ 是实际值（未量化）。这样在给定发送比特 v_l 接收到 r_l 的条件概率密度函数（pdf）为：

$$p(r_l|v_l)=\sqrt{\frac{E_s}{\pi N_0}}\exp\left[-\frac{\left(r_l\sqrt{E_s}-v_l\sqrt{E_s}\right)^2}{N_0}\right]$$
$$=\sqrt{\frac{E_s}{\pi N_0}}\exp\left[-\frac{E_s}{N_0}\cdot\left(r_l-v_l\right)^2\right] \qquad (5.12)$$

其中 N_0/E_s 是噪声的归一化单边 psd。如果信道是无记忆的，发送码字为 **v**、接收序列为 **r** 的似然函数为：

$$M(\mathbf{r}|\mathbf{v})=\ln p(\mathbf{r}|\mathbf{v})=\ln\prod_{l=0}^{N-1}p(r_l|v_l)=\sum_{l=0}^{N-1}\ln p(r_l|v_l)$$
$$=-\frac{E_s}{N_0}\sum_{l=0}^{N-1}(r_l-v_l)^2+\frac{N}{2}\ln\frac{E_s}{\pi N_0}$$
$$=-\frac{E_s}{N_0}\sum_{l=0}^{N-1}(r_l^2-2r_lv_l+1)+\frac{N}{2}\ln\frac{E_s}{\pi N_0} \qquad (5.13)$$
$$=\frac{2E_s}{N_0}\sum_{l=0}^{N-1}(r_lv_l)-\frac{E_s}{N_0}\left(|\mathbf{r}|^2+N\right)+\frac{N}{2}\ln\frac{E_s}{\pi N_0}$$
$$=C_1(\mathbf{r}\cdot\mathbf{v})+C_2$$

其中 $C_1=2E_s/N_0$ 和 $C_2=-\left[(E_s/N_0)(|\mathbf{r}|^2+N)-(N/2)\ln(E_s/\pi N_0)\right]$ 是常数，独立于码字 **v**，$\mathbf{r}\cdot\mathbf{v}$ 表示接收向量 **r** 和码字 **v** 的内积（相关）。由于 C_1 是正数，最大化 $\mathbf{r}\cdot\mathbf{v}$ 的网格路径（码字）同样也最大化对数似然函数 $\ln p(\mathbf{r}|\mathbf{v})$。对应于码字 **v** 的路径度量为 $M(\mathbf{r}|\mathbf{v})=\mathbf{r}\cdot\mathbf{v}$，分支度量为 $M(\mathbf{r}_l|\mathbf{v}_l)=\mathbf{r}_l\cdot\mathbf{v}_l$，$l=0,1,\cdots,h+m-1$，比特度量为 $M(r_l|v_l)=r_l\cdot v_l$，$l=0,1,\cdots,N-1$，Viterbi 算法就是要找到与接收序列相关值最大的那条路径（码字）。

对于连续输出 AWGN 信道，最大化对数似然函数等效为找到与接收序列 **r** 欧拉距离最近的那个码字 **v**，而在 BSC 信道，最大化对数似然函数等效为找到与接收序列 **r** 汉明距离最近的那个码字 **v**。前面也讨论了，软解调器判决（Q>2）相比硬判决（Q＝2）会有性能的提高，如果将前面例子中的 0_1 和 0_2 都变为 0，1_1 和 1_2 都变为 1，则软判决的 DMC 就变为硬判决 BSC 信道（转移概率 p＝0.3），但经过 Viterbi 译码后，产生的信息序列不同，对软判决情况（Q＝4），信息序列 **u**＝（11000），最后度量值是 139；硬判决情况（Q＝2），信息序列 **u**＝（11001），而这样的路径在四输出信道中的度量值为 135，很显然在软判决情况下并不是最大似然路径，因为在硬判决情况下它掩盖了软输出的区别，即对硬判决译码器来说，0_1 和 0_2 都一样，再多的软信息也没用。

5.2 卷积码的性能界

我们首先在 BSC 信道中对特定码字分析最大似然（Viterbi）译码的性能，然后再推广到一般信道。假设发送式（5.1）所示（3，1，2）编码器中的全 0 码字，该编码器的 IOWEF 为：

$$\begin{aligned} A(W,X,L) &= \frac{X^7WL^3}{1-XWL(1+X^2L)} \\ &= X^7WL^3\left[1+XWL(1+X^2L)+X^2W^2L^2(1+X^2L^2)+\cdots\right] \\ &= X^7WL^3+X^8W^2L^4+X^9W^3L^5+X^{10}(W^2L^5+W^4L^6)+\cdots \end{aligned} \qquad (5.14)$$

即该码包含一个重量为 7 的码字，它经过 3 个分支、由重量为 1 的信息序列产生的，一个重量为 8 的码字，它经过 4 个分支、由重量为 2 的信息序列产生的，…

如果全 0 路径（正确路径）在 t 时刻与竞争路径（错误路径）的火拼中第一次被删除，就产生事件错误（第一次），如图 5.6 所示。

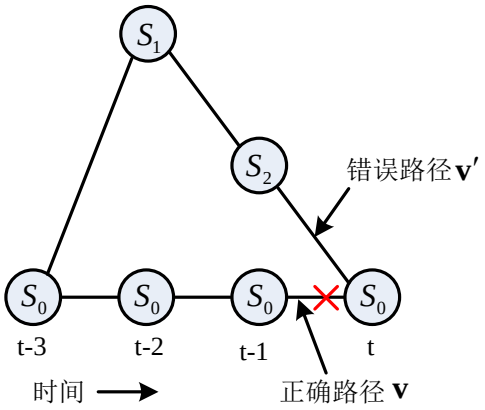


图 5.6 在 t 时刻出现第一次事件错误

错误路径必然是在前面从全 0 态分开、现在又首次回到全 0 态的某个路径，即它必然是码字 WEF A(X)中枚举的一个路径。假设它是重量为 7 的那个路径，则正确路径和错误路径之间有 7 位比特不同，如果接收序列 **r** 与错误路径在这 7 个位置处有不少于 4 个是相同的（即在这 7 个位置至少有 4 个 1），就会发生错误事件。如果 BSC 的转移概率为 p，则这个概率为：

$$\begin{aligned} P_7 &= P\left[7\text{个位置至少有4个“1”}\right] \\ &= \sum_{e=4}^{7}\binom{7}{e}p^e(1-p)^{7-e} \end{aligned} \qquad (5.15)$$

假设重量为 8 的路径是错误路径，发生第一次事件错误的概率为：

$$P_8 = \frac{1}{2}\binom{8}{4}p^4(1-p)^4+\sum_{e=5}^{8}\binom{8}{e}p^e(1-p)^{8-e} \qquad (5.16)$$

因为如果正确路径和错误路径的度量值相同，则发生事件错误的概率为 1/2。通常，如果错误路径的重量为 d，则发生首次错误事件的概率为：

$$P_d = \begin{cases} \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^e (1-p)^{d-e}, & d\text{是奇数} \\ \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} + \sum_{e=\frac{d}{2}+1}^{d} \binom{d}{e} p^e (1-p)^{d-e}, & d\text{是偶数} \end{cases} \quad (5.17)$$

这样，在 t 时刻发生首次事件错误的概率 $P_f(E,t)$ 可用一个界来表示，即为所有这些路径的错误概率之和，如果将超过 t 个分支的错误路径也包括在内，则这个界（较松）可表示为：

$$P_f(E,t) < \sum_{d=d_{free}}^{\infty} A_d P_d \quad (5.18)$$

其中 A_d 是重量为 d 的码字数目（即码字 WEF A(X)中重量为 d 的那一项的因子）。由于这个界与 t 无关（在所有时刻都成立），上式可写为：

$$P_f(E) < \sum_{d=d_{free}}^{\infty} A_d P_d \quad (5.19)$$

对上式还可进一步简化，当 d 是奇数时，

$$\begin{aligned} P_d &= \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^e (1-p)^{d-e} \\ &< \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^{d/2}(1-p)^{d/2} = p^{d/2}(1-p)^{d/2} \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} \quad (5.20) \\ &< p^{d/2}(1-p)^{d/2} \sum_{e=0}^{d} \binom{d}{e} = 2^d\, p^{d/2}(1-p)^{d/2} \end{aligned}$$

【$\sum_{e=0}^{d} \binom{d}{e} = 2^d$】

同样可证明，当 d 是偶数时，仍会得到相同的结果。
因此，

$$P_f(E) < \sum_{d=d_{free}}^{\infty} A_d \left[2\sqrt{p(1-p)}\right]^d \quad (5.21)$$

对于卷积码，其码字 WEF $A(X) = \sum_{d=d_{free}}^{\infty} A_d X^d$，有

$$P_f(E) < A(X)\big|_{X=2\sqrt{p(1-p)}} \quad (5.21)$$

最后的译码路径可以和正确路径分开和合并多次，即它可能会包含多个错误事件，如图 5.7 所示。在发生一次或多次错误事件后，在全 0 态进行比较的两条路径都是错误路径，因为每个路径都至少包含一个以前的错误事件。

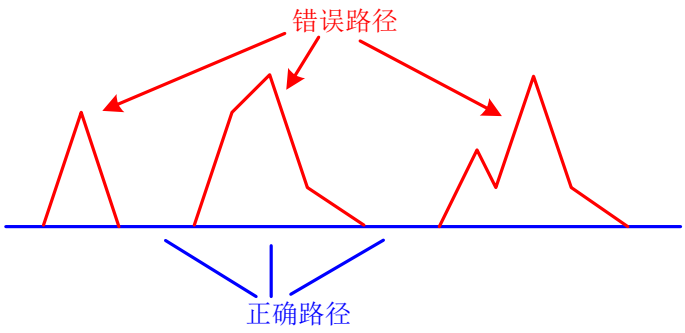


图 5.7 多个错误事件

这样，在 t 时刻发生错误事件的概率 $P(E,t) \le P_f(E,t)$，由于它在所有时刻都成立，因此可写为：

$$P(E) < \sum_{d=d_{free}}^{\infty} A_d P_d < \sum_{d=d_{free}}^{\infty} A_d \left[2\sqrt{p(1-p}\right]^d = A(X)|_{X=2\sqrt{p(1-p)}} \qquad (5.22)$$

由于 p 很小，这个界主要是由其第一项决定的，即自由距离项，因此上式可近似为：

$$P(E) \approx A_{d_{free}} \left[2\sqrt{p(1-p}\right]^{d_{free}} \approx A_{d_{free}} 2^{d_{free}} p^{d_{free}/2} \qquad (5.23)$$

==========

例 5.3：对于式（5.1）所示的（3，1，2）编码器，$d_{free}=7$，$A_{d_{free}}=1$，当 $p=10^{-2}$ 时，有：$P(E) \approx 2^7 p^{7/2} = 1.28\times10^{-5}$

==========

对式（5.22）表示的事件错误概率界进行修改，就可得到比特错误概率 $P_b(E)$ 的界。每个事件错误概率都会造成多个信息比特错误（错误路径的非 0 比特数），因此，如果每个事件错误概率项 P_d 用重量为 d 的路径上的非 0 信息比特数进行加权（注意：重量为 d 的路径可能不止一条），我们就可得到信息比特错误数的界，这个界再除以 k（每个时间单元的信息比特数），就可得到 $P_b(E)$ 的界，为：

$$P_b(E) < \sum_{d=d_{free}}^{\infty} B_d P_d \qquad (5.24)$$

其中 B_d 是所有重量为 d 的路径上的非 0 信息比特总数除以每个时间单元的信息比特数 k（即为比特 WEF $B(X)=\sum_{d=d_{free}}^{\infty} B_d X^d$ 中重量为 d 的那一项的因子）。这样，利用式（5.20）和式（5.24），可得：

$$P_b(E) < \sum_{d=d_{free}}^{\infty} B_d P_d < \sum_{d=d_{free}}^{\infty} B_d \left[2\sqrt{p(1-p}\right]^d = B(X)|_{X=2\sqrt{p(1-p)}} \qquad (5.25)$$

同样，当 p 很小时，上式近似为：

$$P_b(E) \approx B_{d_{free}} \left[2\sqrt{p(1-p)} \right]^{d_{free}} \approx B_{d_{free}} 2^{d_{free}} p^{d_{free}/2} \qquad (5.26)$$

==========

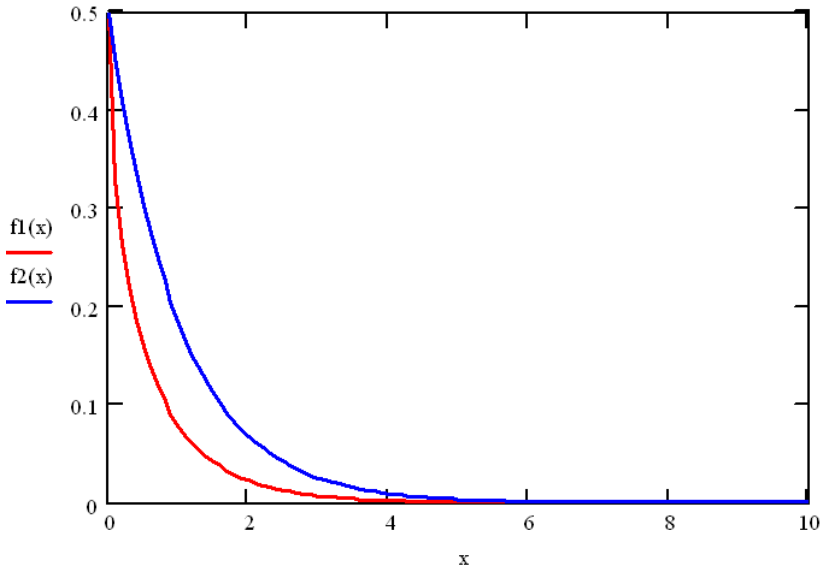
例 5.4：对于式（5.1）所示的（3，1，2）编码器，$d_{free}=7$，$B_{d_{free}}=1$，当 $p=10^{-2}$ 时，有：$P_b(E) \approx 2^7 p^{7/2} = 1.28\times10^{-5}$，与事件错误概率相同。这就是说，当 p 很小时，最可能出错的事件是译码成重量为 7 的路径，因此引起一个信息比特错误。

==========

如果 BSC 是从 AWGN 信道、BPSK 调制、最优相干检测、2-ary 输出量化（硬判决）推导的，则

$$p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) \approx \frac{1}{2} e^{-E_s/N_0} \qquad (5.27)$$

【$Q(x) \le \frac{1}{2} e^{-x^2/2}, \quad x \ge 0$】



这样【当 p 很小时，信道 SNR 就很大】

$$P_b(E) \approx B_{d_{free}} 2^{d_{free}/2} e^{-(d_{free}/2)(E_s/N_0)} \qquad (5.28)$$

由于每比特能量 $E_b \triangleq \frac{E_s}{R}$，R 是编码速率，这样上式改写为（对于较大 E_b/N_0）：

$$P_b(E) \approx B_{d_{free}} 2^{d_{free}/2} e^{-(Rd_{free}/2)(E_b/N_0)}, \quad (\text{with coding}) \qquad (5.29)$$

如果不使用编码，即 $R=1$，$E_s=E_b$，BSC 转移概率 p 是误比特率 $P_b(E)$：

$$P_b(E) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \approx \frac{1}{2} e^{-E_b/N_0}, \quad (\text{without coding}) \qquad (5.30)$$

比较式（5.29）和式（5.30），我们可看到，对于给定的 E_b/N_0，指数部分相差一个因子 $Rd_{free}/2$，由于当 E_b/N_0 较大时，指数项占主导地位，这个因子（dB 值）称为渐近编码增

益（asymptotic coding gain）γ（硬判决情况下）：

$$\gamma \triangleq 10\log_{10}\left(\frac{Rd_{free}}{2}\right)\ dB \qquad (5.31)$$

注意：当 SNR 变小时，编码增益也变小。实际上，如果 SNR（E_b/N_0）降低到编码速率 R 大于信道容量 C 的那个点时，可靠通信已经不可能了，此时未编码系统反而会有更好的性能。

对于二值输入、Q-ary 输出的 DMC 信道，性能界为：

$$P(E) < A(X)|_{X=D_0} \qquad (5.32a)$$

$$P_b(E) < B(X)|_{X=D_0} \qquad (5.32b)$$

其中 $D_0 \triangleq \sum_{j=0}^{Q-1}\sqrt{P(j|0)P(j|1)}$ 是信道转移概率的函数，称为 Bhattacharyya 参数，当 Q=2 时，$D_0 = 2\sqrt{p(1-p)}$，就是 BSC 信道。

在二值输入、连续输出的 AWGN 信道情况下，假设发送码字是全 0 码字 $\mathbf{v} = (-1,-1,\cdots,-1)$【映射关系 $1 \to +1$ 和 $0 \to -1$】，现在来计算正确路径 $\mathbf{v}$ 在 t 时刻首次被删除的概率 P_d，其中 $\mathbf{v}$ 是在与错误路径 $\mathbf{v}'$（与 $\mathbf{v}$ 的 Hamming 距离是 d）的 PK 中“牺牲”的。因为 Viterbi 算法选择具有最大似然函数值的路径，在 t 时刻首次事件错误概率为：

$$\begin{aligned} P_d &= \Pr\left\{M\left([\mathbf{r}|\mathbf{v}']_t\right) > M\left([\mathbf{r}|\mathbf{v}]_t\right)\right\} = \Pr\left\{[\mathbf{r}\cdot\mathbf{v}']_t > [\mathbf{r}\cdot\mathbf{v}]_t\right\} \\ &= \Pr\left\{[\mathbf{r}\cdot\mathbf{v}']_t - [\mathbf{r}\cdot\mathbf{v}]_t > 0\right\} = \Pr\left\{\sum_{l=0}^{t-1}\mathbf{r}_l\cdot\mathbf{v}'_l - \sum_{l=0}^{t-1}\mathbf{r}_l\cdot\mathbf{v}_l > 0\right\} \\ &= \Pr\left\{\sum_{l=0}^{nt-1}r_l\cdot v'_l - \sum_{l=0}^{nt-1}r_l\cdot v_l > 0\right\} \end{aligned} \qquad (5.33)$$

其中 $[\mathbf{r}\cdot\mathbf{v}]_t$ 表示 $\mathbf{r}$ 和 $\mathbf{v}$ 在前 t 个分支的内积。很显然，上式中只有 d 个位置不为 0，假设 $l = 1,2,\cdots,d$ 表示这些位置，因为 $v'_l \neq v_l$，$v'_l = +1$，$v_l = -1$，式（5.33）可表示为：

$$P_d = \Pr\left\{\sum_{l=1}^{d}(+r_l) - \sum_{l=1}^{d}(-r_l) > 0\right\} = \Pr\left\{2\sum_{l=1}^{d}r_l > 0\right\} = \Pr\left\{\sum_{l=1}^{d}r_l > 0\right\} \qquad (5.34)$$

上式表示如果接收信元的值在 $v'_l \neq v_l$ 的 d 个位置上的累积和为正数，就会在 t 时刻发生首次错误事件。

由于信道是无记忆的，发送码字 $\mathbf{v} = (-1,-1,\cdots,-1)$，$\rho \triangleq \sum_{l=1}^{d}r_l$ 是 d 个独立高斯随机变量的和，每个 r_l 是均值为－1、方差为 $N_0/2E_s$ 的高斯随机变量（见式 5.12），这样 ρ 是均

值为－d、方差为 $N_0/2E_s$ 的高斯随机变量，则式（5.34）可写为：

$$P_d = \Pr\{\rho > 0\} = \left(\sqrt{\frac{E_s}{\pi d N_0}}\right)\int_0^{\infty} e^{-\frac{E_s(\rho+d)^2}{dN_0}} d\rho \qquad (5.35)$$

进行变量代换 $y = (\rho + d)\sqrt{\frac{2E_s}{dN_0}}$，上式变为：

$$\begin{aligned} P_d &= \frac{1}{\sqrt{2\pi}}\int_{+d\sqrt{2E_s/dN_0}}^{\infty} e^{-\frac{y^2}{2}} dy \quad = \frac{1}{\sqrt{2\pi}}\int_{+\sqrt{2dE_s/N_0}}^{\infty} e^{-\frac{y^2}{2}} dy \\ &= Q\left(\sqrt{\frac{2dE_s}{N_0}}\right) \quad = Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right) \end{aligned} \qquad (5.36)$$

将式（5.36）代入式（5.22）和式（5.25）的界，得到二值输入、连续输出 AWGN 信道下的事件错误概率和比特错误概率的界，为：

$$P(E) < \sum_{d=d_{free}}^{\infty} A_d Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right) \qquad (5.37a)$$

$$P_b(E) < \sum_{d=d_{free}}^{\infty} B_d Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right) \qquad (5.37b)$$

如果用更松的界 $Q(x) \le \frac{1}{2}e^{-x^2/2} < e^{-x^2/2}$，　$x \ge 0$，可得：

$$P(E) < \sum_{d=d_{free}}^{\infty} A_d e^{-\frac{dRE_b}{N_0}} = A(X)\Big|_{X=\exp(-RE_b/N_0)} \qquad (5.38a)$$

$$P_b(E) < \sum_{d=d_{free}}^{\infty} B_d e^{-\frac{dRE_b}{N_0}} = B(X)\Big|_{X=\exp(-RE_b/N_0)} \qquad (5.38b)$$

比较式（5.38）和式（5.32），我们可知，对于连续输出 AWGN 信道，Bhattacharyya 参数 $D_0 = \exp(-RE_b/N_0)$。

当 E_b/N_0 较大时，在比特 WEF 中的第一项决定了式（5.38b）的界，$P_b(E)$ 近似为：

$$P_b(E) \approx B_{d_{free}} e^{-Rd_{free}E_b/N_0} \qquad (5.39)$$

此时渐近编码增益（软判决）为：

$$\gamma \triangleq 10\log_{10}\left(Rd_{free}\right)\ dB \qquad (5.40)$$

比较式（5.39）、（5.40）和式（5.29）、（5.31），我们可看到软判决情况比硬判决情况多出 3dB 增益，这也就意味着为了得到相同的错误概率，在 BSC 信道上传输的发射机的发射功率是 AWGN 信道下发射功率的 2 倍。（注：前面的分析是基于对特定码字的性能界，且只当 E_b/N_0 较大时才有效，当用随机码进行分析且 SNR 较小时，编码增益只有 2dB 左右，所

以软判决相对硬判决有 2～3dB 的增益，代价是译码的复杂度增加）。

这个界通过下面的不等式还可以更紧，

$$Q\left(\sqrt{y+z}\right)\le Q\left(\sqrt{y}\right)e^{-\frac{z}{2}},\quad \left(y>0,z\ge 0\right) \tag{5.41}$$

设 $y=2d_{free}RE_b/N_0$， $z=2(d-d_{free})RE_b/N_0$，式（5.36）可写为：

$$\begin{aligned} P_d &= Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right)=Q\left(\sqrt{y+z}\right)\\ &\le Q\left(\sqrt{\frac{2d_{free}RE_b}{N_0}}\right)e^{-\frac{(d-d_{free})RE_b}{N_0}} \qquad (5.42)\\ &= Q\left(\sqrt{\frac{2d_{free}RE_b}{N_0}}\right)e^{\frac{d_{free}RE_b}{N_0}}e^{-\frac{dRE_b}{N_0}} \end{aligned}$$

现在定义函数

$$f(x)=Q\left(\sqrt{2x}\right)e^{x} \tag{5.43}$$

式（5.42）可写为：

$$P_d\le f\left(\frac{d_{free}RE_b}{N_0}\right)e^{-\frac{dRE_b}{N_0}} \tag{5.44}$$

最后，将式（5.44）代入式（5.22）和式（5.25）的界，得到二值输入、连续输出 AWGN 信道下的事件错误概率和比特错误概率的界，为：

$$\begin{aligned} P(E) &< \sum_{d=d_{free}}^{\infty}A_d f\left(\frac{d_{free}RE_b}{N_0}\right)e^{-\frac{dRE_b}{N_0}}\\ &= f\left(\frac{d_{free}RE_b}{N_0}\right)A(X)\Big|_{X=\exp(-RE_b/N_0)} \end{aligned} \tag{5.45a}$$

$$\begin{aligned} P_b(E) &< \sum_{d=d_{free}}^{\infty}B_d f\left(\frac{d_{free}RE_b}{N_0}\right)e^{-\frac{dRE_b}{N_0}}\\ &= f\left(\frac{d_{free}RE_b}{N_0}\right)B(X)\Big|_{X=\exp(-RE_b/N_0)} \end{aligned} \tag{5.45b}$$

我们从式（5.45）表示的紧界和式（5.38）表示的界可知，紧界有一个因子 $f\left(\frac{d_{free}RE_b}{N_0}\right)$，该因子只依赖于码字的自由距。

==========

例 5.5：计算 AWGN 信道的误比特率界

考虑式（5.1）所示（3，1，2）编码器，其 IOWEF A(W, X, L)见式（5.14），其比特 WEF 可计算为：

$$B(X)=\frac{1}{k}\cdot\left.\frac{\partial A(W,X)}{\partial W}\right|_{W=1}=\left.\frac{\partial\left[X^7W/\left(1-XW-X^3W\right)\right]}{\partial W}\right|_{W=1} \qquad (5.46)$$

$$=\frac{X^7}{1-2X+X^2-2X^3+2X^4+X^6}=X^7+2X^8+3X^9+6X^{10}+\cdots$$

可知，该码的自由距 $d_{free}=7$，利用上式可直接得到（5.38b）和（5.45b）所示的界，如图 5.8 所示。

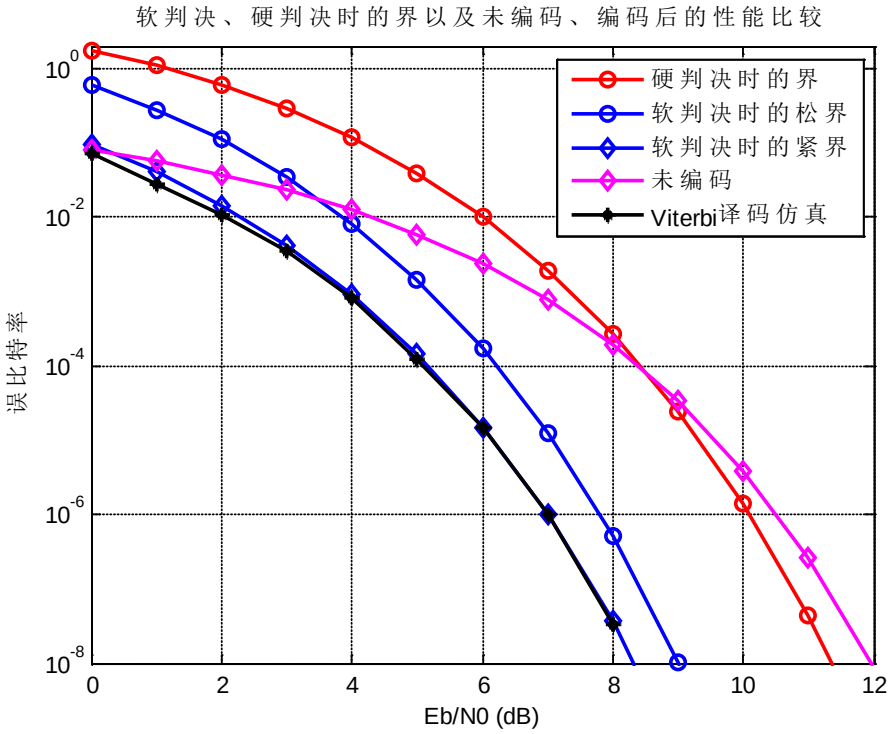


图 5.8 卷积码的性能界

从上图我们可看出，式（5.45b）表示的紧界和 viterbi 译码仿真非常拟合（SNR>4dB），而式（5.38b）表示的界一直追随着仿真曲线，但不是很靠近。

该码的渐近编码增益（软判决）为：

$$\gamma=10\log_{10}\left(Rd_{free}\right)=10\log_{10}\left(7/3\right)=3.68\,dB \qquad (5.47)$$

这是与未编码 BPSK 系统性能相比的增益，从上图我们可以看出这一点。

从上面的讨论我们可知，P(E)和 $P_b(E)$随 d_{free} 的增加指数下降，随 $A_{d_{free}}$ 和 $B_{d_{free}}$ 的增加线性增加，因此首要准则是最大化自由距离 d_{free}，第二个准则是最小化 $A_{d_{free}}$ 和 $B_{d_{free}}$。通常来说，在高 SNR 时，d_{free} 起决定作用，在低 SNR 时，$A_{d_{free}}$ 和 $B_{d_{free}}$ 起决定作用。

表 5.1 列出了一些最优码：

表 5.1(a) R＝1/4 的卷积码

v	$G^{(0)}$	$g^{(1)}$	$g^{(2)}$	$g^{(3)}$	d_{free}	$A_{d_{free}}$	$\gamma(dB)$
1	1	1	3	3	6	1	1.76
2	5	5	7	7	10	1	3.98
3	13	13	15	17	13	2	5.12
4	25	27	33	37	16	4	6.02
5	45	53	67	77	18	3	6.53
6	117	127	155	171	20	2	6.99
7	257	311	337	355	22	1	7.40
8	533	575	647	711	24	1	7.78
9	1173	1325	1467	1751	27	3	8.29

表 5.1(b) R＝1/3 的卷积码

v	$G^{(0)}$	$g^{(1)}$	$g^{(2)}$	d_{free}	$A_{d_{free}}$	$\gamma(dB)$
1	1	3	3	5	1	2.22
2	5	7	7	8	2	4.26
3	13	15	17	10	3	5.22
4	25	33	37	12	5	6.02
5	47	53	75	13	1	6.36
6	117	127	155	15	3	6.99
7	225	331	367	16	1	7.27
8	575	623	727	18	1	7.78
9	1167	1375	1545	20	3	8.23
10	2325	2731	3747	22	7	8.65
11	5745	6471	7553	24	13	9.03
12	2371	13725	14733	24	5	9.03

表 5.1(c) R＝1/2 的卷积码

v	$G^{(0)}$	$g^{(1)}$	d_{free}	$A_{d_{free}}$	$\gamma(dB)$
1	3	1	3	1	1.76
2	5	7	5	1	3.98
3	13	17	6	1	4.77
4	27	31	7	2	5.44
5	53	75	8	1	6.02
6	117	155	10	11	6.99
7	247	371	10	1	6.99
8	561	753	12	11	7.78
9	1131	1537	12	1	7.78
10	2473	3217	14	14	8.45
11	4325	6747	15	14	8.75
12	10627	16765	16	14	9.03
13	27251	37363	16	1	9.03

5.3 软输出 Viterbi 算法（SOVA）

5.3.1 SOVA 基本原理

对于卷积码，Viterbi 算法是最优的最大似然译码方法，译码输出为卷积码的最优估计序列。但对于属于级联卷积码的 Turbo 码而言，传统的 Viterbi 算法存在两个缺陷：首先，一个分量译码器输出中存在的突发错误会影响另一个分量译码器的译码性能，从而使级联码的性能下降。其次，无论是软判决 Viterbi 算法还是硬判决 Viterbi 算法，其译码输出均为硬判决信息，若一个分量码采用 Viterbi 算法译码，则另一个分量译码器只能以硬判决结果作为输入，无法实现软判决译码，从而性能会有所下降。因此，如果 Viterbi 译码器能够提供软信息输出，则可以弥补上述两个缺陷，并且可以通过在分量译码器之间软信息的交换使级联码的性能大大提高。为此，需要在传统的 Viterbi 算法上进行修正，使之提供软信息输出，相应的算法就称为软输出 Viterbi 算法，记做 SOVA（Soft Output Viterbi Algorithm）。

下面通过具体的实例来说明软输出 Viterbi 算法的软信息的生成。图 5.9 给出了在一个 4 状态网格图实现 SOVA 译码的计算输出软信息的示意图。

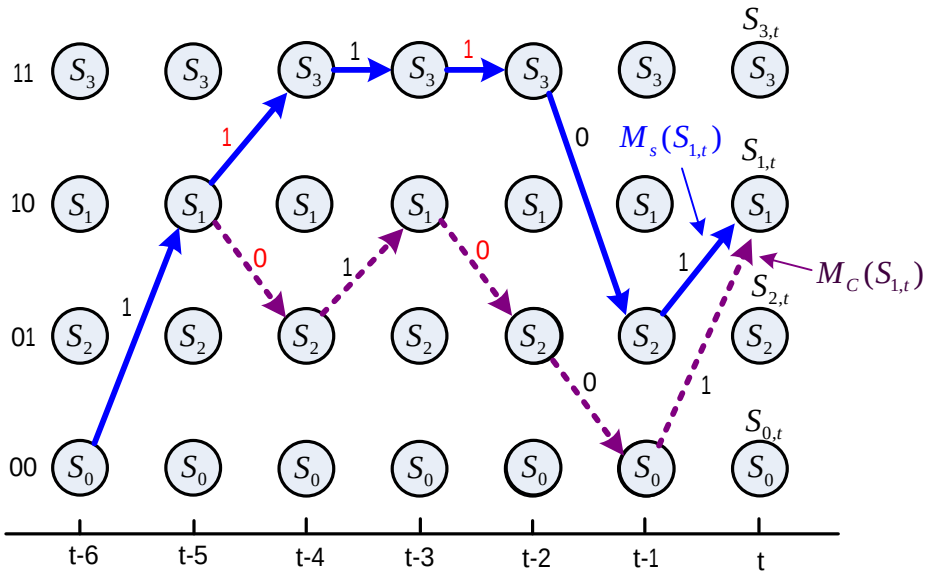


图 5.9 实现竞争路径和幸存路径可靠性估计的格图

图 5.9 中的实线表示 t 时刻状态结点 $S_{1,t}$ 处的幸存路径(这里假设为最终的最大似然路径的一部分)，虚线表示 t 时刻状态结点 $S_{1,t}$ 处的竞争路径。为使图看上去清楚一些，就没有画出其他结点的幸存路径和竞争路径。标记 $S_{1,t}$ 代表 t 时刻的状态 1，标记(0, 1)表示每个分支路径对应的二元判决值，当前结点时刻的幸存路径对应的累计度量值记为 $M_S(S_{1,t})$，当前结点时刻的竞争路径对应的累计度量值记为 $M_C(S_{1,t})$。指定结点 $S_{1,t}$ 的幸存路径的可靠性度量值 $\Delta_t(S_1)$ 为上述两个累计度量值之差的绝对值，即 $\Delta_t(S_1)=\left|M_S(S_{1,t})-M_C(S_{1,t})\right|$，这个值越大，幸存路径是最大似然路径的可能性就越大，判决也越可靠。这里假设幸存路径累计度量总是优于竞争路径累计度量。后面就将 $\Delta_t(S_1)$ 称为 t 时刻的 SOVA 译码输出的可靠性值。

显然，根据 $\Delta_t(S_1)=\left|M_S(S_{1,t})-M_C(S_{1,t})\right|$ 计算得到的值可以作为先验信息，辅助下一个分量译码器进行译码判决。

此外，为降低复杂性，只需要计算最大似然幸存路径的可靠性值(这里假设已知)。而其他幸存路径没有必要计算可靠性值，随着译码的进行，这些路径逐个被丢弃了。

为说明可靠性的概念，下面给出两个例子。在这些例子中，Viterbi 算法选择累计度量值小的路径作为幸存路径。

在第一个例子中，状态结点 $S_{1,t}$ 处幸存路径累计度量值为 $M_S(S_{1,t})=50$，竞争路径累计度量值 $M_C(S_{1,t})=100$，选择这个幸存路径时相应的可靠性值为

$$\Delta_t(S_1)=\left|M_S(S_{1,t})-M_C(S_{1,t})\right|=\left|50-100\right|=50$$

在第二个例子中，状态结点 $S_{1,t}$ 处幸存路径累计度量值 $M_S(S_{1,t})=50$；竞争路径累计度量值 $M_C(S_{1,t})=75$，选择这个幸存路径时相应的可靠性值为

$$\Delta_t(S_1)=\left|M_S(S_{1,t})-M_C(S_{1,t})\right|=\left|50-75\right|=25$$

尽管在这两个例子中幸存路径的累计度量值相同，但幸存路径的可靠程度并不同。根据可靠性度量值的大小可以判定：第一个例子中提供的幸存路径比第二个例子中提供的幸存路径更可信一些。

图 5.10 说明了使用幸存路径累计度量值与竞争路径累计度量值之差的绝对值作为可靠性时的示意图。

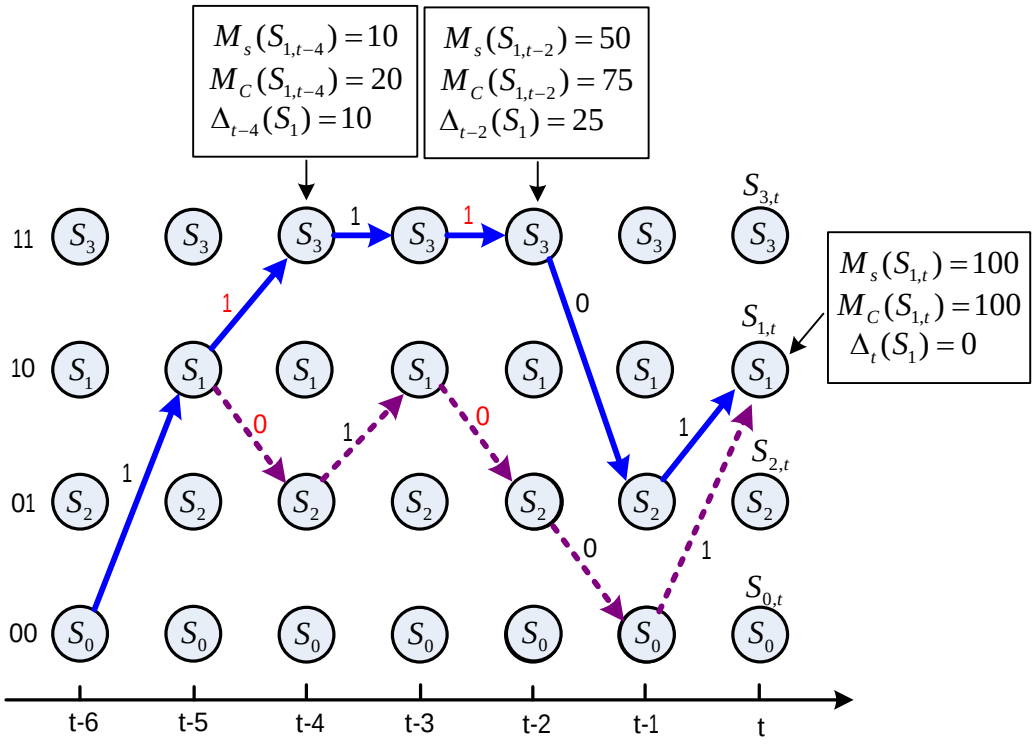


图 5.10 不同幸存路径和竞争路径外部信息的比较

在图 5.10 中，状态的幸存路径和竞争路径在 t－5 时刻的状态结点 $S_{1,t-5}$ 处开始分离，在 t－2 和 t－4 时刻幸存路径和竞争路径得到的比特判决值完全相反，如图中红色所示。为方便说明，假设在状态结点 $S_{1,t}$ 处竞争路径累计度量值和幸存路径累计度量值相等，且 $M_S(S_{1,t}) = M_C(S_{1,t}) = 100$。这意味着幸存路径和竞争路径成为最终的最大似然路径的概率是相等的。此外，类似于图 5.9，还假设在 t－2 和 t－4 时刻幸存路径累计度量值优于竞争路径的累计度量值。为使图看上去清楚一些，这些时刻的竞争路径没有画出。

从图 5.10 可以看出，t 时刻幸存路径相应的可靠性值为 $\Delta_t(S_1) = 0$。这意味着选择该路径作为幸存路径没有任何可靠性。在 t－2 和 t－4 时刻，幸存路径相应的可靠性值均大于 0（$\Delta_{t-2}(S_1) = 25$，$\Delta_{t-4}(S_1) = 10$），这也意味着幸存路径的累计度量值较好。但在 t 时刻，由于幸存路径和竞争路径具有相同的度量，因此竞争路径也可以是幸存路径。这样，在不降低幸存路径的相关可靠性情况下，在 t－2 和 t－4 时刻存在完全相反的二元判决。

为改善幸存路径的可靠性值，Hagenauer 等人提出了通过后向跟踪更新可靠性值的方法。这个更新过程嵌入到 Viterbi 算法中，得到的算法描述如下。

对于格图中的状态结点 $S_{k,t}$（相应于 t 时刻的状态 k）：

①存储可靠性 $\Delta_t(S_k) = \left| M_S(S_{k,t}) - M_C(S_{k,t}) \right|$，如果存在不止一条竞争路径对应的可靠性值，就将最小的一个存储到 $\Delta_t(S_k)$。

②初始化状态 $S_{k,t}$ 的可靠性值为+∞。

③比较状态 $S_{k,t}$ 的幸存路径和竞争路径，并存储反映两个路径的二元判决不同的相对时刻值到 MEM，即与当前时刻的差值；

④按照下面的步骤更新这些 MEM 处的可靠性值。

- 找到可靠性值没有更新过、满足 MEM＞0 且值最小的 MEM．记做 MEM_{low}。
- 用 MEM＝0 和 MEM＝MEM_{low} 之间最小的可靠性值来更新 MEM_{low} 的可靠性值。

继续前面的例子，状态结点 $S_{1,t}$ 处幸存路径和竞争路径之间比特判决相反的位置存储为 MEM＝{2，4}，根据这个 MEM 信息，在图 5.11 和图 5.12 中给出了更新可靠性值的过程。图 5.11 中给出了第一个可靠性度量值的更新过程。MEM＞0 且未更新可靠性值的最小 MEM_{low}＝2，在 MEM＝0 和 MEM＝MEM_{low}＝2 之间最小的可靠性值为 $\Delta_t(S_1) = 0$，这样相应的可靠性值就由 $\Delta_{t-2}(S_1) = 25$ 更新为 $\Delta_{t-2}(S_1) = \Delta_t(S_1) = 0$。下一个 MEM＞0 且未更新可靠性值的最小 MEM_{low}＝4，在 MEM＝0 和 MEM＝MEM_{low}＝4 之间最小的可靠性值为 $\Delta_t(S_1) = 0$，这样相应的可靠性值就更新为 $\Delta_{t-4}(S_1) = \Delta_t(S_1) = 0$。图 5.12 给出了第二个可靠性值更新的过程。

研究表明，最后的可靠性值在送入下一个分量译码器之前应该经过归一化或者对数化处理，以减小由于可靠性值更新带来的影响。

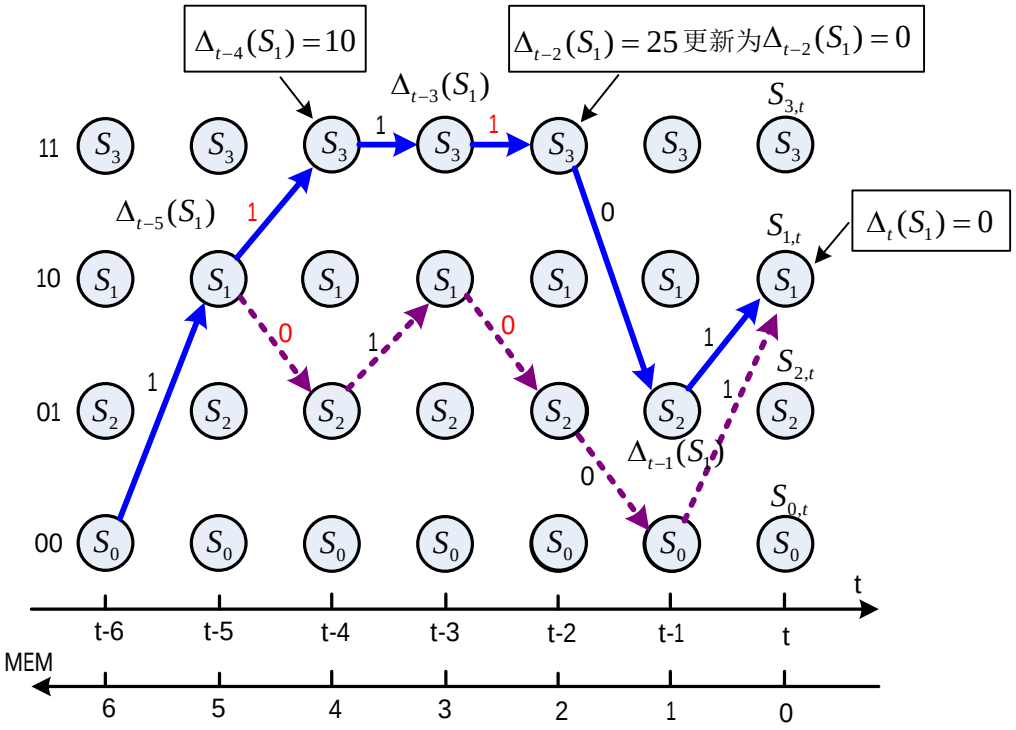


图 5.11 t－2 时刻路径度量更新过程

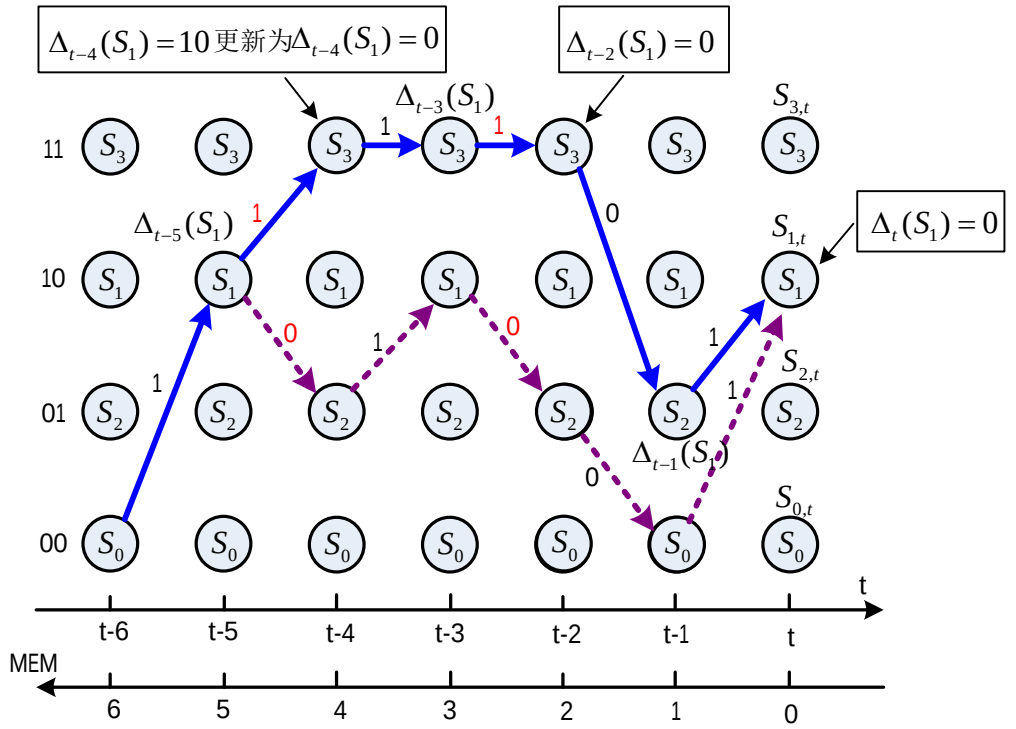


图 5.12 t－4 时刻路径度量更新过程

5.3.2 SOVA

在级联译码系统，通常一个译码器会将其译码输出的可靠度信息（软输出）传递给第二个译码器，这样第二个译码器就可以使用软判决译码。能够处理从信道（或另一个译码器）

传来的软输入值，并将软输入值传递给另一个译码器，象这种译码器称为软输入软输出（SISO，Soft-In, Soft-Out）译码器。

软输出 Viterbi 算法（SOVA）是 Hagenauer 和 Hoeher 在 1989 年提出的，我们主要讲解在二值输入、连续输出 AWGN 信道下编码速率为 R＝1/n 的卷积码的 SOVA。SOVA 的基本运算与 Viterbi 算法相同，唯一的区别在于对每个信息比特的硬判决输出附加一个可靠度的指示，这两者（硬判决输出和可靠度指示）组合在一起就是软输出。假设信息比特的先验概率为 $p(u_l),\ l=0,1,\cdots,h-1$，在 t 时刻，部分接收序列为 $[\mathbf{r}]_t=(\mathbf{r}_0,\mathbf{r}_1,\cdots\mathbf{r}_{t-1})=\left(r_0^{(0)},r_0^{(1)},\cdots r_0^{(n-1)};r_1^{(0)},r_1^{(1)},\cdots r_1^{(n-1)};r_{t-1}^{(0)},r_{t-1}^{(1)},\cdots r_{t-1}^{(n-1)}\right)$，部分路径度量必须最大化（应用 Viterbi 算法）：

$$M\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)=\ln\left\{p\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)p\left([\mathbf{v}]_t\right)\right\} \qquad (5.48)$$

式（5.48）和式（5.13）的区别就在于包含了先验路径概率 $p\left([\mathbf{v}]_t\right)$【因为当信息比特的先验概率不是等概时，先验路径概率也是不等概的】，注意：先验路径概率 $p\left([\mathbf{v}]_t\right)$就是相关的信息序列$[\mathbf{u}]_t$ 的先验概率，我们可以将部分路径度量按照 $l=t$ 时刻和 $l<t$ 时刻分开：

$$\begin{aligned}M\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)&=\ln\left\{\left[\prod_{l=0}^{t-1}p(\mathbf{r}_l\,|\,\mathbf{v}_l)p(u_l)\right]p(\mathbf{r}_t\,|\,\mathbf{v}_t)p(u_t)\right\}\\&=\ln\left\{\prod_{l=0}^{t-1}p(\mathbf{r}_l\,|\,\mathbf{v}_l)p(u_l)\right\}+\ln\left\{\prod_{j=0}^{n-1}p(r_t^{(j)}\,|\,v_t^{(j)})p(u_t)\right\} \qquad (5.49)\\&=\ln\left\{\prod_{l=0}^{t-1}p(\mathbf{r}_l\,|\,\mathbf{v}_l)p(u_l)\right\}+\left\{\sum_{j=0}^{n-1}\ln\left[p(r_t^{(j)}\,|\,v_t^{(j)})\right]+\ln\left[p(u_t)\right]\right\}\end{aligned}$$

对上式进行修正：将和式中的每一项乘以 2，并引入常数 $C_r^{(j)}$ 和 C_u，为：

$$\left\{\sum_{j=0}^{n-1}\left[2\ln\left[p(r_t^{(j)}\,|\,v_t^{(j)})\right]-C_r^{(j)}\right]+\left[2\ln\left[p(u_t)\right]-C_u\right]\right\} \qquad (5.50)$$

其中常数

$$C_r^{(j)}\equiv\ln\left[p(r_t^{(j)}\,|\,v_t^{(j)}=+1)\right]+\ln\left[p(r_t^{(j)}\,|\,v_t^{(j)}=-1)\right],\ j=0,1,\cdots n-1 \qquad (5.51a)$$

$$C_u\equiv\ln\left[p(u_t=+1)\right]+\ln\left[p(u_t=-1)\right] \qquad (5.51b)$$

是独立于路径$[\mathbf{v}]_t$，这里的映射为$1\to+1$和$0\to-1$。类似地，对式（5.49）中 $l<t$ 时刻的每一项也进行修正，【注意：这种修正不会影响路径$[\mathbf{v}]_t$ 的最大化】，修正后的度量为：

$$M^*\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)=M^*\left([\mathbf{r}\,|\,\mathbf{v}]_{t-1}\right)+\sum_{j=0}^{n-1}\left\{2\ln\left[p(r_t^{(j)}\,|\,v_t^{(j)})\right]-C_r^{(j)}\right\}+\left\{2\ln\left[p(u_t)\right]-C_u\right\}$$

$$=M^*\left([\mathbf{r}\,|\,\mathbf{v}]_{t-1}\right)+\sum_{j=0}^{n-1}v_t^{(j)}\ln\left[\frac{p(r_t^{(j)}\,|\,v_t^{(j)}=+1)}{p(r_t^{(j)}\,|\,v_t^{(j)}=-1)}\right]+u_t\ln\left[\frac{p(u_t)=+1}{p(u_t)=-1}\right] \qquad (5.52)$$

我们可以通过定义在二进制输入 $v=\pm1$、未量化信道的输出为 r 的对数似然率（Log-likelihood ratio）或 L 值对上式进行简化。

$$L(r)=\ln\left[\frac{p(r\,|\,v=+1)}{p(r\,|\,v=-1)}\right] \qquad (5.53)$$

类似地，信息比特 u 的 L 值为：

$$L(u)=\ln\left[\frac{p(u=+1)}{p(u=-1)}\right] \qquad (5.54)$$

L 值可解释为对一个二进制随机变量的可靠度测量。例如，假设码比特 v 的先验（未传输前）概率是等概的，即 $p(v=+1)=p(v=-1)=1/2$，【在信息比特是等概的，且码是线性的情况下，这种假设是成立的】，利用 Bayes 准则，式（5.53）可写为：

$$L(r)=\ln\left[\frac{p(r\,|\,v=+1)}{p(r\,|\,v=-1)}\right]=\ln\left[\frac{p(v=+1\,|\,r)}{p(v=-1\,|\,r)}\right] \qquad (5.55)$$

从上式可以看出，给定接收信元 r，如果 $L(r)$是一个大的正值，表示 $v=+1$ 的可靠性较大，如果 $L(r)$是一个大的负值，表示 $v=-1$ 的可靠性较大，如果 $L(r)$是一个接近于 0 的值，表示只根据 r 来判决 v 值是非常不可靠的。类似地，$L(u)$也满足这个特性。

对 AWGN 信道，接收到的信元为 r'，二进制输入信元 $v'=\pm\sqrt{E_s}$，SNR 为 E_s/N_0，可得到：

$$L(r')=(4\sqrt{E_s}/N_0)r' \qquad (5.56a)$$

将上式重写为：

$$L(r)=(4E_s/N_0)r \qquad (5.56b)$$

其中归一化值 $r\equiv r'/\sqrt{E_s}$ 对应于输入是 $v=\pm1$ 时接收到的信元。式（5.56b）阐明了接收到的信元 r 的可靠度，它随信道 SNR E_s/N_0 线性增加。定义 $L_c\equiv4E_s/N_0$ 为信道可靠度因子，则 SOVA 译码算法中的修正度量可表示为：

$$M^*\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)=M^*\left([\mathbf{r}\,|\,\mathbf{v}]_{t-1}\right)+\sum_{j=0}^{n-1}L_c v_t^{(j)}r_t^{(j)}+u_tL(u_t) \qquad (5.57)$$

其中上式中的前 2 项就是式（5.13），而 $u_tL(u_t)$ 项在式（5.13）中并没有出现，这是因为前面假设信息比特序列是等概的。由式（5.57）可看出，SOVA 度量中结合了当前时刻之前的度量值、信道可靠度信息以及先验信息。

图 5.13 给出了 SOVA 度量计算中用到的先验信息。

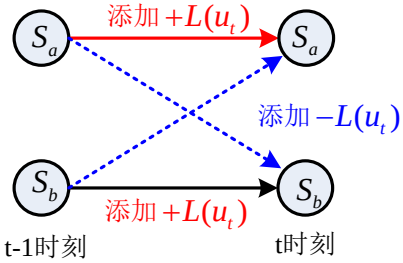


图 5.13 SOVA 度量中计算的信息值

在上图中包含了两个状态 S_a 和 S_b 在 t−1 时刻到 t 时刻的状态转移格图，红实线表示输入 $u_t=+1$ 时的状态转移分支路径，蓝虚线表示输入 $u_t=-1$ 时的状态转移分支路径。先验信息 $L(u_t)$ 可以是正值，也可以是负值，来自于另一个分量译码器，该先验信息结合在 SOVA 度量中以提供对估计信息比特更可靠的判决度量。

图 5.14 描述了 SOVA 度量的加权特性。

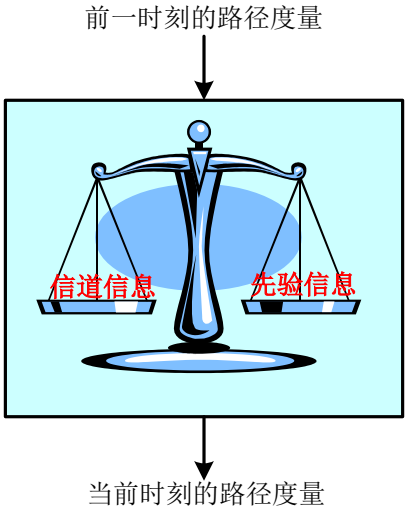


图 5.14 SOVA 度量加权示意图

信道可靠性值与先验信息之间的平衡对于 SOVA 度量非常重要，如果信道条件非常好，则 L_c 大于 $|L(u_t)|$，这时译码应该主要根据接收的信道值进行；但如果信道条件很差，译码应该主要根据先验信息 $L(u_t)$ 进行。

前面已经提到 SOVA 算法与 Viterbi 算法非常类似，只是对每个硬判决输出附加了一个可靠度测量因子。假设在状态 $S_i, i=0,1,\cdots 2^v-1$ 对最大似然路径 $[\mathbf{v}]_t$ 和一个错误路径 $[\mathbf{v}']_t$ 在 $l=t$ 时刻进行一次比较，定义度量差为：

$$\Delta_t(S_i)=\frac{1}{2}\left\{M^*\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)-M^*\left([\mathbf{r}\,|\,\mathbf{v}']_t\right)\right\} \qquad (5.58)$$

最大似然路径被正确选择的概率 P(C)为：

$$P(C)=\frac{p\left([\mathbf{v}\,|\,\mathbf{r}]_t\right)}{p\left([\mathbf{v}\,|\,\mathbf{r}]_t\right)+p\left([\mathbf{v}'\,|\,\mathbf{r}]_t\right)} \qquad (5.59)$$

其中

$$p\left([\mathbf{v}\,|\,\mathbf{r}]_t\right)=\frac{p\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)p\left([\mathbf{v}]_t\right)}{p(\mathbf{r})} \overset{\text{根据式（9.48}}{=\!=} \frac{e^{M\left([\mathbf{r}|\mathbf{v}]_t\right)}}{p(\mathbf{r})} \qquad (5.60a)$$

$$p\left([\mathbf{v}'\,|\,\mathbf{r}]_t\right)=\frac{p\left([\mathbf{r}\,|\,\mathbf{v}']_t\right)p\left([\mathbf{v}']_t\right)}{p(\mathbf{r})}=\frac{e^{M\left([\mathbf{r}|\mathbf{v}']_t\right)}}{p(\mathbf{r})} \qquad (5.60b)$$

根据式（5.50）中所用的修正方法，我们可写为：

$$M^*\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)=2M\left([\mathbf{r}\,|\,\mathbf{v}]_t\right)-c \qquad (5.61a)$$

$$M^*\left([\mathbf{r}\,|\,\mathbf{v}']_t\right)=2M\left([\mathbf{r}\,|\,\mathbf{v}']_t\right)-c \qquad (5.61b)$$

其中 c 是一个不依赖于$[\mathbf{v}]_t$和$[\mathbf{v}']_t$的常数。

利用式（5.60）和式（5.61），式 （5.59）可重写为：

$$P(C)=\frac{e^{M\left([\mathbf{r}|\mathbf{v}]_t\right)}}{e^{M\left([\mathbf{r}|\mathbf{v}]_t\right)}+e^{M\left([\mathbf{r}|\mathbf{v}']_t\right)}}=\frac{e^{\frac{M^*\left([\mathbf{r}|\mathbf{v}]_t\right)+c}{2}}}{e^{\frac{M^*\left([\mathbf{r}|\mathbf{v}]_t\right)+c}{2}}+e^{\frac{M^*\left([\mathbf{r}|\mathbf{v}']_t\right)+c}{2}}} \qquad (5.62)$$

$$=\frac{e^{M^*\left([\mathbf{r}|\mathbf{v}]_t\right)/2}}{e^{M^*\left([\mathbf{r}|\mathbf{v}]_t\right)/2}+e^{M^*\left([\mathbf{r}|\mathbf{v}']_t\right)/2}}=\frac{e^{\Delta_t(S_i)}}{1+e^{\Delta_t(S_i)}}$$

最后，这个路径判决的对数似然率（可靠度）为：

$$\ln\left(\frac{P(C)}{1-P(C)}\right)=\Delta_t(S_i) \qquad (5.63)$$

现在我们来看看一个路径的可靠度是如何与 Viterbi 译码器的硬判决输出这两者之间有效联系的。t 时刻某个给定结点的幸存路径可靠度为$\Delta_{t-MEM}(S_i)$，其中 MEM＝0，1，…，t。对于 t 时刻的这个状态结点，如果在 MEM＝k 处（相当于 t－MEM 时刻）幸存路径上对应的信息比特与同时刻竞争路径上的信息比特相同，则选择竞争路径也不会发生比特判决错误，因此，这个比特位置的可靠度保持不变；但如果在 MEM＝k 处幸存路径和竞争路径相应的信息比特不同，则会发生比特判决错误，这样就必须用前面讲述的更新过程来更新这个比特位置的可靠度，如图 5.15 所示，对于 MEM＝2 和 MEM＝4，需要更新可靠度。

可靠度更新过程用于改善译码器软输出值或者似然值，比特判决的软输出或者似然比值为：

$$L(\hat{u}_{t-MEM})=u_{t-MEM}\sum_{k=0}^{MEM}\Delta_{t-k}(S_i) \qquad (5.64)$$

为简化计算，上式可近似为：

$$L(\hat{\hat{u}}_{t-MEM}) \approx u_{t-MEM} \min_{k=0,1,...,MEM} \{\Delta_{t-k}(S_i)\} \qquad (5.65)$$

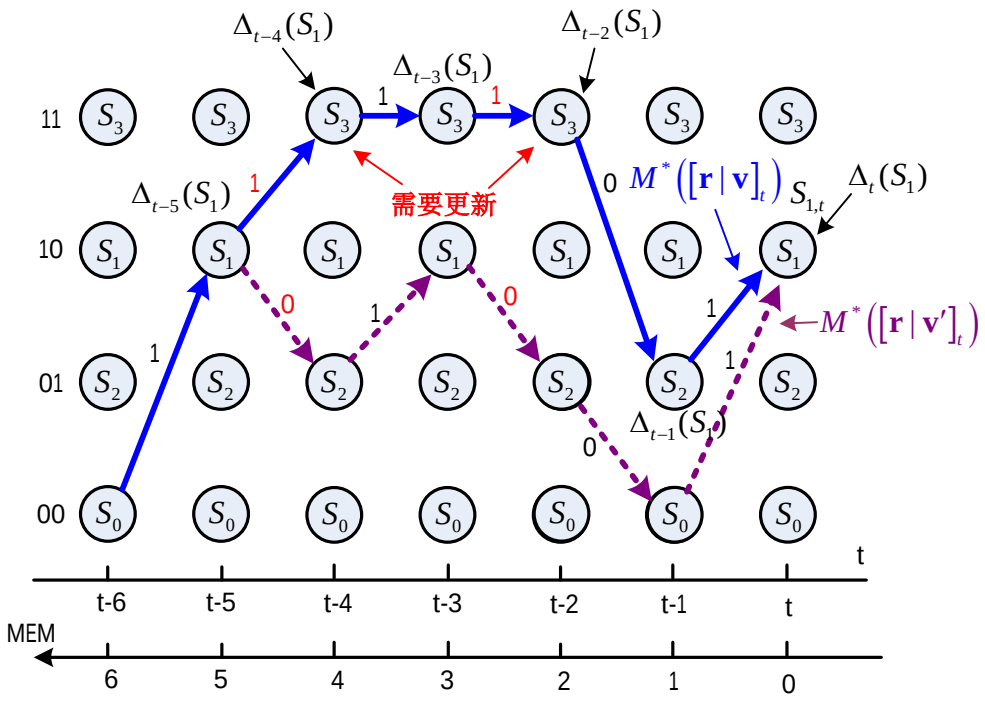


图 5.15 SOVA 的幸存路径和竞争路径度量

综上所述，递归 SOVA 译码器的译码过程（包括可靠度更新过程）如下：

（1）（a）初始化时刻 t＝0；

（b）对 0 状态初始化度量值为 0，其他状态初始化为∞（算法不同，也可为－∞）。

（2）（a）t←t＋1；

（b）对格图中的每个状态按照式（5.57）计算正确路径度量和错误路径度量；

（3）为每个状态搜索最小度量值（或最大度量值）。

（4）存储正确路径度量及相应的幸存比特和状态路径；

（5）按照式（5.58）计算度量差；

（6）比较 t 时刻每个状态处的幸存路径和竞争路径，并存储两条路径上的比特判决不同的相对时刻值 MEM。

（7）更新所有 MEM 对应的度量值。

（8）返回到步骤（2），直到接收到整个传输序列。

（9）输出估计的比特序列 $\hat{\mathbf{u}}$ 和相应的软输出值 $L(\hat{\mathbf{u}}) = \mathbf{u} \cdot \Delta$，其中 Δ 为最后更新的可靠度序列，作为先验信息用于下一轮译码。

5.4 BCJR 算法

给定一个接收序列 **r**，Viterbi 算法是寻找能够使对数似然函数最大化的码字 **v**。在 BSC 信道下，这等效为找到一个在 Hamming 距离上与（二进制）接收序列 **r** 最靠近的（二进制）码字 **v**，对于更一般的连续输出的 AWGN 信道，这等效为找到与（实值）接收序列 **r** 在

Euclidean 距离最靠近的（二进制）码字 $\mathbf{v}$，一旦 ML 码字 $\mathbf{v}$ 确定了，它所对应的信息序列 $\mathbf{u}$ 就是译码输出。

由于 Viterbi 算法是寻找最相似的码字，即最小化 WER $P_w(E)$，使译码（ML）后的码字 $\hat{\mathbf{v}}$ 不等于发送码字 $\mathbf{v}$ 的概率 $p(\hat{\mathbf{v}} \neq \mathbf{v} \mid \mathbf{r})$ 最小。但在很多情况下，我们希望最小化 BER $P_b(E)$，即在 l 时刻译码信息比特 $\hat{u}_l$ 不等于发送比特 u_l 的概率 $p(\hat{u}_l \neq u_l \mid \mathbf{r})$ 最小，$l = 0,1,\cdots,K^* - 1$。为了最小化 BER，则一个信息比特 u_l 被正确译码的后验概率 $p(\hat{u}_l = u_l \mid \mathbf{r})$ 必须最大化。最大化 $p(\hat{u}_l = u_l \mid \mathbf{r})$ 的算法称为最大后验概率（MAP，Maximum A Posteriori）译码器。当信息比特等概时，Viterbi 算法（ML）最大化 $p(\hat{\mathbf{v}} = \mathbf{v} \mid \mathbf{r})$，虽然它不能保证 $p(\hat{u}_l = u_l \mid \mathbf{r})$ 也最大化，但 $p(\hat{\mathbf{v}} = \mathbf{v} \mid \mathbf{r})$ 和 $p(\hat{u}_l = u_l \mid \mathbf{r})$ 密切相关，Viterbi 算法也可得到近最优的 BER 性能。

Viterbi 算法最大化 $p(\hat{\mathbf{v}} = \mathbf{v} \mid \mathbf{r}) = p(\mathbf{u} = \mathbf{u} \mid \mathbf{r})$，其中 $\hat{\mathbf{u}}$ 和 $\mathbf{u}$ 是信息序列，分别对应于码字 $\hat{\mathbf{v}}$ 和 $\mathbf{v}$。这等效为最小化 $p(\hat{\mathbf{u}} \neq \mathbf{u} \mid \mathbf{r})$，BER 为：

$$p(\hat{u_l} \neq u_l \mid \mathbf{r}) = \left[\frac{d(\hat{\mathbf{u}}, \mathbf{u})}{K^*}\right] p(\mathbf{u} \neq \mathbf{u} \mid \mathbf{r}),\ l = 0,1,\cdots,K^* - 1 \qquad (5.66)$$

从上式可看出，BER 依赖于 $\hat{\mathbf{u}}$ 和 $\mathbf{u}$ 序列之间的 Hamming 距离【这点与计算 $p(\hat{\mathbf{u}} \neq \mathbf{u} \mid \mathbf{r})$ 类似】，因此最小化 $p(\hat{u}_l \neq u_l \mid \mathbf{r})$ 也涉及到选择编码器具有以下特性：低重量码字对应于低重量信息序列。这样能够保证最大似然码字错误导致尽量少的比特错误，从而 BER 也就最小化了。

1974 年，Bahl, Cocke, Jelinek 及 Raviv 引入了 MAP 译码器，称为 BCJR 算法，它能够应用于任何线性分组码或卷积码，BCJR 的计算复杂度比 Viterbi 算法要高的多，因此在等概信息比特情况下一般选择 Viterbi 算法。但当信息比特不等概时，MAP 译码能够获得更好的性能，而且，当使用迭代译码时，每次迭代信息比特的先验概率都会发生改变，此时 MAP 译码器更适合。

我们主要描述编码速率 R＝1/n 的卷积码在二值输入、连续输出 AWGN 信道和 DMC 下的 BCJR 算法（仍然基于前面讲到的对数似然率或 L 值）。译码器输入是接收到的序列 $\mathbf{r}$，信息比特的先验 L 值为 $L_a(u_l),\ l = 0,1,\cdots,h-1$。象在 SOVA 一样，我们不假设信息比特是等概的。该算法计算后验 L 值为：

$$L(u_l) \equiv \ln\left[\frac{p(u_l = +1 \mid \mathbf{r})}{p(u_l = -1 \mid \mathbf{r})}\right] \qquad (5.67)$$

称为每个信息比特的 APP L 值，译码器输出为：

$$\hat{u}_l = \begin{cases} +1 & if\ L(u_l) > 0 \\ -1 & if\ L(u_l) < 0 \end{cases},\ l = 0,1,\cdots,h-1 \qquad (5.68)$$

在迭代译码中，APP L 值可作为译码器输出，形成 SISO 译码算法。
我们知道

$$p(u_l = +1 \mid \mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{p(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_l^+} p(\mathbf{r} \mid \mathbf{v}) p(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r} \mid \mathbf{v}) p(\mathbf{u})} \qquad (5.69)$$

其中 $\mathbf{U}_l^+$ 是 $u_l = +1$ 的所有信息序列集合，$\mathbf{v}$ 是发送的码字对应于信息序列 $\mathbf{u}$，$p(\mathbf{r} \mid \mathbf{v})$ 是给定 $\mathbf{v}$ 接收序列 $\mathbf{r}$ 的 pdf，同样我们可写出 $p(u_l = -1 \mid \mathbf{r})$ 的表达式，这样式（5.67）可写为：

$$L(u_l) \equiv \ln \left[\frac{\sum_{\mathbf{u} \in \mathbf{U}_l^+} p(\mathbf{r} \mid \mathbf{v}) p(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_l^-} p(\mathbf{r} \mid \mathbf{v}) p(\mathbf{u})} \right] \qquad (5.70)$$

其中 $\mathbf{U}_l^-$ 是 $u_l = -1$ 的所有信息序列集合。MAP 译码可通过式（5.70）计算 APP L 值 $L(u_l),\ l = 0,1,\cdots,h-1$，然后应用式（5.68）来实现。但这样的计算量非常大。对于具有网格结构和有限状态的码，如短约束长度的卷积码，利用基于网格图的递归计算能够大大简化处理过程。
首先，利用码的网格图结构，我们可以将式（5.69）重写为：

$$p(u_l = +1 \mid \mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{p(\mathbf{r})} = \frac{\sum_{(s',s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{p(\mathbf{r})} \qquad (5.71)$$

其中 Σ_l^+ 表示在 l 时刻状态为 $s_l = s'$，$l+1$ 时刻状态为 $s_{l+1} = s$，这种状态转移是由输入比特 $u_l = +1$ 所引起的，所有这些状态对的集合就构成 Σ_l^+。同样我们可写出 $p(u_l = -1 \mid \mathbf{r})$ 的表达式，这样式（5.67）可写为：

$$L(u_l) = \ln \left[\frac{\sum_{(s',s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{\sum_{(s',s) \in \Sigma_l^-} p(s_l = s', s_{l+1} = s, \mathbf{r})} \right] \qquad (5.72)$$

其中 Σ_l^- 表示状态转移对应于输入比特 $u_l = -1$ 的集合。

式（5.70）和式（5.72）对 APP L 值 $L(u_l)$ 是等效的，但式（5.70）中的加项涉及到 2^{h-1} 个信息序列的集合，而式（5.72）中的加项只涉及 2^v 个状态对的集合，因此，对较大的 block 长度 h，式（5.72）要简单的多。

现在我们通过递归的方法来推导式（5.72）中的联合 pdf $p(s', s, \mathbf{r})$：

$$p(s', s, \mathbf{r}) = p(s', s, \mathbf{r}_{t<l}, \mathbf{r}_l, \mathbf{r}_{t>l}) \qquad (5.73)$$

其中 $\mathbf{r}_{t<l}$ 表示在 l 时刻之前接收到的信息序列 $\mathbf{r}$，$\mathbf{r}_{t>l}$ 表示在 l 时刻之后接收到的信息序列 $\mathbf{r}$，

应用 Bayes 准则，有

$$\begin{aligned} p(s',s,\mathbf{r}) &= p(\mathbf{r}_{t>l} \mid s',s,\mathbf{r}_{t<l},\mathbf{r}_l)\,p(s',s,\mathbf{r}_{t<l},\mathbf{r}_l) \\ &= p(\mathbf{r}_{t>l} \mid s',s,\mathbf{r}_{t<l},\mathbf{r}_l)\,p(s,\mathbf{r}_l \mid s',\mathbf{r}_{t<l})\,p(s',\mathbf{r}_{t<l}) \\ &= p(\mathbf{r}_{t>l} \mid s)\,p(s,\mathbf{r}_l \mid s')\,p(s',\mathbf{r}_{t<l}) \end{aligned} \qquad (5.74)$$

其中最后一个等式是因为在 l 时刻接收到的分支的概率只依赖于 l 时刻的状态和 l 时刻的输入。

定义：

$$\alpha_l(s') \equiv p(s',\mathbf{r}_{t<l}) \qquad (5.75a)$$

$$\gamma_l(s',s) \equiv p(s,\mathbf{r}_l \mid s') \qquad (5.75b)$$

$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{t>l} \mid s) \qquad (5.75c)$$

这样，式（5.74）可写为：

$$p(s',s,\mathbf{r}) = \beta_{l+1}(s)\gamma_l(s',s)\alpha_l(s') \qquad (5.76)$$

对于概率 $\alpha_{l+1}(s)$，我们可重新将表达式写为：

$$\begin{aligned} \alpha_{l+1}(s) = p(s,\mathbf{r}_{t<l+1}) &= \sum_{s'\in\sigma_l} p(s',s,\mathbf{r}_{t<l+1}) \\ &= \sum_{s'\in\sigma_l} p(s,\mathbf{r}_l \mid s',\mathbf{r}_{t<l})\,p(s',\mathbf{r}_{t<l}) \\ &= \sum_{s'\in\sigma_l} p(s,\mathbf{r}_l \mid s')\,p(s',\mathbf{r}_{t<l}) \\ &= \sum_{s'\in\sigma_l} \gamma_l(s,s')\alpha_l(s') \end{aligned} \qquad (5.77)$$

其中 σ_l 是在 l 时刻的所有状态的集合。因此，利用前向递归（5.77），我们可计算出在 l+1 时刻、每个状态 s 的前向度量 $\alpha_{l+1}(s)$。类似地，我们可写出概率 $\beta_l(s')$ 的表达式：

$$\beta_l(s') = \sum_{s\in\sigma_{l+1}} \gamma_l(s',s)\beta_{l+1}(s) \qquad (5.78)$$

其中 σ_{l+1} 式在 l+1 时刻的所有状态的集合。利用后向递归（5.78），我们可计算出在 l 时刻、每个状态 s' 的后向度量 $\beta_l(s')$。在 l=0 时刻开始的前向递归初始条件为：

$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases} \qquad (5.79)$$

因为编码器是在全 0 态 $S_0 = \mathbf{0}$ 开始的，这样可用式（5.77）来递归计算 $\alpha_{l+1}(s)$，

$l=0,1,\cdots,K-1$，其中 K＝h＋m 是输入序列长度。类似地，在 l＝K 时刻开始的后向递归，初始条件为：

$$\beta_K(s)=\begin{cases}1, & s=\mathbf{0}\\ 0, & s\neq\mathbf{0}\end{cases} \qquad (5.80)$$

因为编码器终止于全 0 态 $S_0=\mathbf{0}$，这样可用式（5.78）来递归计算 $\beta_l(s)$，$l=K-1,K-2,\cdots,0$。

分支度量 $\gamma_l(s',s)$ 可写为：

$$\begin{aligned}\gamma_l(s',s)&=p(s,\mathbf{r}_l\mid s')=\frac{p(s',s,\mathbf{r}_l)}{p(s')}\\&=\left[\frac{p(s',s)}{p(s')}\right]\left[\frac{p(s',s,\mathbf{r}_l)}{p(s',s)}\right] \qquad (5.81)\\&=p(s\mid s')p(\mathbf{r}_l\mid s',s)=p(u_l)p(\mathbf{r}_l\mid\mathbf{v}_l)\end{aligned}$$

其中 u_l 是输入比特，$\mathbf{v}_l$ 对应于 l 时刻状态转移 $s'\to s$ 的输出比特。对一个连续输出 AWGN 信道，如果 $s'\to s$ 是一个有效的状态转移，则

$$\gamma_l(s',s)=p(u_l)p(\mathbf{r}_l\mid\mathbf{v}_l)=p(u_l)\left(\sqrt{\frac{E_s}{\pi N_0}}\right)^n e^{-\frac{E_s}{N_0}\|\mathbf{r}_l-\mathbf{v}_l\|^2} \qquad (5.82)$$

其中 $\|\mathbf{r}_l-\mathbf{v}_l\|^2$ 表示（用 $\sqrt{E_s}$ 归一化的）接收分支 $\mathbf{r}_l$ 和 l 时刻的发送分支 $\mathbf{v}_l$ 之间的平方欧拉距离，但如果 $s'\to s$ 不是一个有效的状态转移，$p(s\mid s')$ 和 $\gamma_l(s',s)$ 都为 0。用式（5.72）、式（5.76）、以及式（5.77）～（5.80）和（5.82）所定义的度量来计算 APP L 值 $L(u_l)$ 的算法，就称为 MAP 算法。

对算法进行修改，可大大减少计算复杂度。首先，注意到式（5.82）中有常数项 $\left(\sqrt{\frac{E_s}{\pi N_0}}\right)^n$，从式（5.76）～（5.78）可知，在概率密度函数 $p(s',s,\mathbf{r})$ 的表达式中该常数项的指数部分增加了 h 倍，即在式（5.72）的分子和分母中的每一项都有 $\left(\sqrt{\frac{E_s}{\pi N_0}}\right)^{nh}$ 这个因子，其影响可忽略，因此，简化的分支度量为：

$$\gamma_l(s',s)=p(u_l)e^{-\frac{E_s}{N_0}\|\mathbf{r}_l-\mathbf{v}_l\|^2} \qquad (5.83)$$

其次，先验概率 $p(u_l=\pm1)$ 可写为：

$$p(u_l=\pm 1)=\frac{\left[p(u_l=+1)\big/p(u_l=-1)\right]^{\pm 1}}{\left\{1+\left[p(u_l=+1)\big/p(u_l=-1)\right]^{\pm 1}\right\}} \qquad (5.84)$$

$$=\frac{e^{\pm L_a(u_l)}}{1+e^{\pm L_a(u_l)}}=\frac{e^{-L_a(u_l)/2}}{1+e^{-L_a(u_l)}}e^{u_l L_a(u_l)/2}=A_l e^{u_l L_a(u_l)/2}$$

其中参数 A_l 独立于 u_l 的实际值。然后用式（5.84）来取代（5.83）中的 $p(u_l),\ l=0,1,\cdots h-1$。但对每个发送比特 $u_l, l=h,h+1,\cdots,h+m-1=K-1$，每个有效的状态转移都有 $p(u_l)=1$，$L_a(u_l)=\pm\infty$，我们可直接用式（5.83）。因此，我们可以将式（5.83）写为：

$$\gamma_l(s',s)=A_l e^{u_l L_a(u_l)/2}e^{-\frac{E_s}{N_0}\|\mathbf{r}_l-\mathbf{v}_l\|^2}$$

$$=A_l e^{u_l L_a(u_l)/2}e^{(2E_s/N_0)(\mathbf{r}_l\cdot\mathbf{v}_l)-(E_s/N_0)\|\mathbf{r}_l\|^2-(E_s/N_0)\|\mathbf{v}_l\|^2}$$

$$=A_l e^{u_l L_a(u_l)/2}e^{(2E_s/N_0)(\mathbf{r}_l\cdot\mathbf{v}_l)}e^{-(E_s/N_0)\|\mathbf{r}_l\|^2-(E_s/N_0)\|\mathbf{v}_l\|^2} \qquad (5.85a)$$

$$=A_l e^{u_l L_a(u_l)/2}e^{(L_c/2)(\mathbf{r}_l\cdot\mathbf{v}_l)}e^{-(E_s/N_0)\|\mathbf{r}_l\|^2-(E_s/N_0)n}$$

$$=A_l B_l e^{u_l L_a(u_l)/2}e^{(L_c/2)(\mathbf{r}_l\cdot\mathbf{v}_l)},\quad l=0,1,\cdots,h-1$$

$$\gamma_l(s',s)=p(u_l)e^{-\frac{E_s}{N_0}\|\mathbf{r}_l-\mathbf{v}_l\|^2} \qquad (5.85b)$$

$$=B_l e^{(L_c/2)(\mathbf{r}_l\cdot\mathbf{v}_l)},\quad l=h,h+1,\cdots,K-1$$

其中 $B_l=e^{-(E_s/N_0)\|\mathbf{r}_l\|^2-(E_s/N_0)n}$ 是一个独立于 $\mathbf{v}_l$ 的常数，$L_c=4E_s/N_0$ 是信道可靠度因子。

从式（5.76）～（5.80）和式（5.85）我们注意到，pdf $p(s',s,\mathbf{r})$ 含有因子 $\prod_{l=0}^{h-1}A_l$ 和 $\prod_{l=0}^{K-1}B_l$，这些因子在式（5.72）的分子和分母中的每一项都会出现，因此它们可消去，更简化的分支度量就变为：

$$\gamma_l(s',s)=e^{u_l L_a(u_l)/2}e^{(L_c/2)(\mathbf{r}_l\cdot\mathbf{v}_l)},\quad l=0,1,\cdots,h-1 \qquad (5.86a)$$

$$\gamma_l(s',s)=e^{(L_c/2)(\mathbf{r}_l\cdot\mathbf{v}_l)},\quad l=h,h+1,\cdots,K-1 \qquad (5.86b)$$

最后，从式（5.77）、（5.78）和（5.86）可看出，前向度量和反向度量是 $2^k=2$ 个指数项的和，每个对应于网格图中的一个有效状态转移。

利用下面的等式可简化计算：

$$\max{}^*(x,y)\equiv\ln(e^x+e^y)=\max(x,y)+\ln(1+e^{-|x-y|}) \qquad (5.87)$$

$\ln(e^x+e^y)$ 计算很复杂，用上式取代只有一个 max 函数和一个查表函数（对于

$\ln(1+e^{-|x-y|})$），有了这个铺垫，引入以下对数域度量：

$$\gamma_l^*(s',s)\equiv\ln\gamma_l(s',s)=\begin{cases}\dfrac{u_l L_a(u_l)}{2}+\dfrac{L_c}{2}\mathbf{r}_l\cdot\mathbf{v}_l, & l=0,1,\cdots,h-1\\ \dfrac{L_c}{2}\mathbf{r}_l\cdot\mathbf{v}_l, & l=h,h+1,\cdots,K-1\end{cases} \tag{5.88a}$$

$$\begin{aligned}\alpha_{l+1}^*(s)&\equiv\ln\alpha_{l+1}(s)=\ln\sum_{s'\in\sigma_l}\gamma_l(s',s)\alpha_l(s')\\&=\ln\sum_{s'\in\sigma_l}e^{\left[\gamma_l^*(s',s)+\alpha_l^*(s')\right]}\\&=\max{}^*_{s'\in\sigma_l}\left[\gamma_l^*(s',s)+\alpha_l^*(s')\right],\ l=0,1,\cdots,K-1\end{aligned} \tag{5.88b}$$

$$\alpha_0^*(s)\equiv\ln\alpha_0(s)=\begin{cases}0, & s=\mathbf{0}\\ -\infty, & s\neq\mathbf{0}\end{cases} \tag{5.88c}$$

$$\begin{aligned}\beta_l^*(s')&\equiv\ln\beta_l(s')=\ln\sum_{s\in\sigma_{l+1}}\gamma_l(s',s)\beta_{l+1}(s)\\&=\ln\sum_{s\in\sigma_{l+1}}e^{\left[\gamma_l^*(s',s)+\beta_{l+1}^*(s)\right]}\\&=\max{}^*_{s\in\sigma_{l+1}}\left[\gamma_l^*(s',s)+\beta_{l+1}^*(s)\right],\ l=K-1,K-2,\cdots,0\end{aligned} \tag{5.88d}$$

$$\beta_K^*(s)\equiv\ln\beta_K(s)=\begin{cases}0, & s=\mathbf{0}\\ -\infty, & s\neq\mathbf{0}\end{cases} \tag{5.88e}$$

这样，式（5.76）表示的 pdf $p(s',s,\mathbf{r})$ 和式（5.72）表示的 APP L 值 $L(u_l)$ 可写为：

$$p(s',s,\mathbf{r})=e^{\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')} \tag{5.89}$$

$$L(u_l)=\ln\left\{\sum_{(s',s)\in\Sigma_l^+}e^{\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')}\right\}-\ln\left\{\sum_{(s',s)\in\Sigma_l^-}e^{\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')}\right\} \tag{5.90}$$

我们看到，在式（5.90）中每一项都要计算 2^ν 个指数项和的对数，对应于网格图中的每个状态。应用式（5.87）定义的 $\max^*$ 函数，我们可以处理多个指数项和的情况，即

$$\max{}^*(x,y,z)\equiv\ln(e^x+e^y+e^z)=\max{}^*\left[\max{}^*(x,y),z)\right] \tag{5.91}$$

利用上式，式（5.90）写为：

$$\begin{aligned}L(u_l)=&\max{}^*_{(s',s)\in\Sigma_l^+}\left[\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')\right]\\&-\max{}^*_{(s',s)\in\Sigma_l^-}\left[\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')\right]\end{aligned} \tag{5.92}$$

基于式（5.88）定义的对数域度量、式（5.87）和（5.91）定义的$\max^*$函数，用式（5.92）计算 APP L 值$L(u_l)$的算法称为 log-MAP 算法，或对数域 BCJR 算法。由于 log-MAP 算法只是用了一个 max()函数和一次查表，因此比 MAP 算法要简单的多。

对数域 BCJR 算法步骤：

（1） 用（5.88c）和（5.88e）来初始化前向度量$\alpha_0^*(s)$和后向度量$\beta_K^*(s)$；

（2） 用（5.88a）计算分支度量$\gamma_l^*(s',s)$，$l=0,1,\cdots,K-1$；

（3） 用（5.88b）计算前向度量$\alpha_{l+1}^*(s)$，$l=0,1,\cdots,K-1$；

（4） 用（5.88d）计算后向度量$\beta_l^*(s')$，$l=K-1,K-2,\cdots,0$；

（5） 用式（5.92）计算 APP L 值$L(u_l)$，$l=0,1,\cdots,h-1$；

（6） （可选）用式（5.68）计算硬判决$\hat{u}_l$，$l=0,1,\cdots,h-1$；

对于步骤（3）、（4）、（5），每个的计算量都与 Viterbi 算法相同，因此 BCJR 算法的复杂度大致是 Viterbi 算法的 3 倍，而且，从式（5.88a）可以看出，BCJR 算法需要信道 SNR E_s/N_0（$L_c=4E_s/N_0$）信息来计算其分支度量，而 Viterbi 算法中的分支度量，只是$\mathbf{r}_l\cdot\mathbf{v}_l$的相关值，不需要信道信息。【说明：在式（5.13）中计算 Viterbi 算法中的分支度量中也有一个常数量 C_1（$=L_c/2$），它可以忽略，因为它不影响译码结果；而在式（5.88a）的 BCJR 算法中常数量 $L_c/2$ 不能忽略，因为它与先验因子$u_lL_a(u_l)/2$有关，会影响到译码结果。】当 BCJR 算法用于迭代译码算法时，步骤（6）的硬判决计算只在最后一次迭代时才涉及，而在前面的迭代中，步骤（5）的 APP L 值是译码器软输出。对数域 BCJR 算法的基本运算如图 5.16 所示。

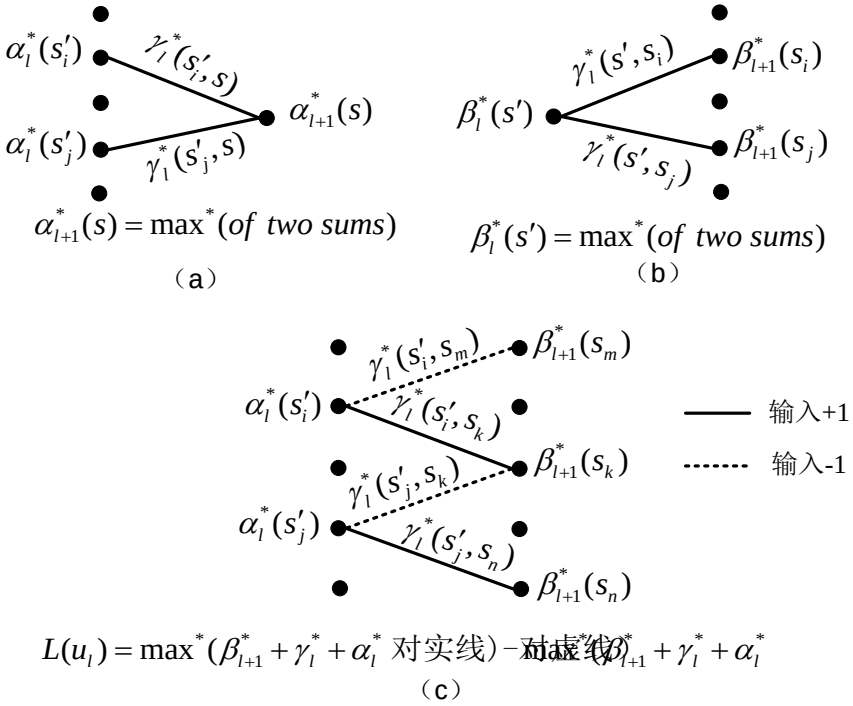


图 5.16 （a）式 5.88b 表示的前向递归 （b）式 5.88d 表示的后向递归
（c）式 5.92 表示的 APP L 值

如果在 max*函数中忽略校正项 $\ln(1+e^{-|x-y|})$，算法更简单，近似为：

$$\max^*(x, y) \approx \max(x, y) \qquad (5.93)$$

因为校正项的界为：

$$0 < \ln(1+e^{-|x-y|}) \le \ln(2) = 0.693 \qquad (5.94)$$

当 $|\max(x, y)| \ge 7$ 时，上述近似是非常合理的。在计算 APP L 值 $L(u_l)$ 时，如果用 max 函数代替 max*函数，则算法称为 Max-log-MAP 算法。由于 max 函数的作用与 Viterbi 算法中的比-选运算相同，Max-log-MAP 算法中的前向递归等效于一个前向 Viterbi 算法，后向递归等效于一个后向 Viterbi 算法。BCJR 算法和 Viterbi 算法的差异就在于校正项 $\ln(1+e^{-|x-y|})$，它保证了 BCJR 算法中的 BER 最优，而 Viterbi 算法是对 WER 最优的。

上述 3 个算法：MAP、log-MAP、Max-log-MAP，都属于前向-后向算法，因为都涉及前向递归和后向递归。同时，它们还属于 SISO 译码器，因为真实值 APP L 值 $L(u_l)$ 作为译码器软输出，而不是象式（5.68）那样进行译码器的硬判决输出。

==========

例 5.6：在 AWGN 信道对（2，1，1）系统递归卷积码的 BCJR 译码

（2，1，1）系统递归卷积码的生成矩阵为：

G(D)=[1 1/(1+D)] （5.95）

其编码器结构如图 5.17（a）所示，状态数为 2，映射 0→-1，1→+1，输入序列长度为 4，如图 5.17（b）所示。设 $\mathbf{u}=(u_0,u_1,u_2,u_3)$ 表示长度 K＝h＋m＝4 的输入向量，$\mathbf{v}=(\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3)$ 表示长度 N＝nK＝8 的码字，假设信道 SNR E_s/N_0＝1/4 (-6.02dB)，接收向量（用 $\sqrt{E_s}$ 归一化）为：

$$\begin{aligned}\mathbf{r}&=(\mathbf{r}_0,\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3)=(r_0^{(0)},r_0^{(1)};r_1^{(0)},r_1^{(1)};r_2^{(0)},r_2^{(1)};r_3^{(0)},r_3^{(1)})\\&=(+0.8,+0.1;+1.0,-0.5;-1.8,+1.1;+1.6,-1.6)\end{aligned}\qquad(5.96)$$

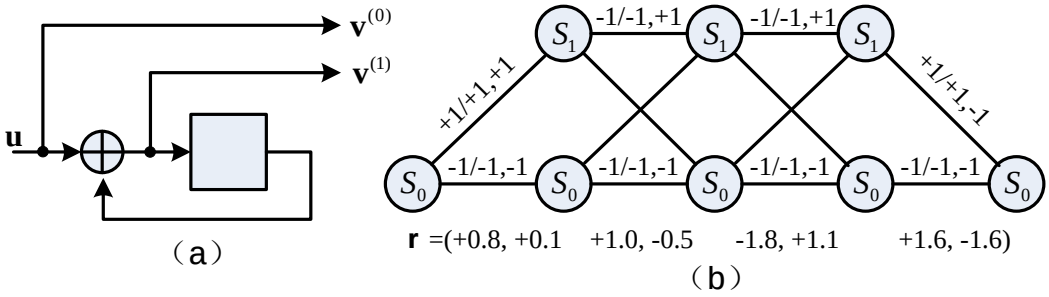


图 5.17 （a）（2，1，1）系统反馈编码器（b）网格图 K＝4，N＝8

【说明：（1）在网格图中，分支标识 $u_l/v_l^{(0)},v_l^{(1)}$ 指示了输入 u_l 和输出 $v_l^{(0)},v_l^{(1)}$；（2）对反馈编码器网格图来说，离开每个状态的上面分支并不必对应于 1（＋1）输入，下面的分支也不是必须对应于 0（－1）输入；（3）由于编码器是系统的，每个分支的第一个输出就等于输入；（4）对于结尾编码器，编码速率 R＝h/N＝3/8，E_s/N_0＝1/4(-6.02dB)，对应于 E_b/N_0＝E_s/RN_0＝2/3(-1.76dB)。】

假设信息比特的先验概率是等概的，即 $L_a(u_l)=0,\ l=0,1,2$，用式（5.88a）计算对数域的分支度量（注意：L_c＝$4E_s/N_0$＝1）为：

$$\gamma_0^*(S_0,S_0)=\frac{u_0L_a(u_0)}{2}+\frac{L_c}{2}\mathbf{r}_0\cdot\mathbf{v}_0=\frac{-1}{2}L_a(u_0)+\frac{1}{2}\mathbf{r}_0\cdot\mathbf{v}_0$$

$$=\frac{1}{2}(-0.8-0.1)=-0.45$$

$$\gamma_0^*(S_0,S_1)=\frac{+1}{2}L_a(u_0)+\frac{1}{2}\mathbf{r}_0\cdot\mathbf{v}_0=\frac{1}{2}(0.8+0.1)=0.45$$

$$\gamma_1^*(S_0,S_0)=\frac{-1}{2}L_a(u_1)+\frac{1}{2}\mathbf{r}_1\cdot\mathbf{v}_1=\frac{1}{2}(-1.0+0.5)=-0.25$$

$$\gamma_1^*(S_0,S_1)=\frac{+1}{2}L_a(u_1)+\frac{1}{2}\mathbf{r}_1\cdot\mathbf{v}_1=\frac{1}{2}(1.0-0.5)=0.25$$

$$\gamma_1^*(S_1,S_1)=\frac{-1}{2}L_a(u_1)+\frac{1}{2}\mathbf{r}_1\cdot\mathbf{v}_1=\frac{1}{2}(-1.0-0.5)=-0.75$$

$$\gamma_1^*(S_1,S_0)=\frac{+1}{2}L_a(u_1)+\frac{1}{2}\mathbf{r}_1\cdot\mathbf{v}_1=\frac{1}{2}(1.0+0.5)=0.75$$

$$\gamma_2^*(S_0,S_0)=\frac{1}{2}\mathbf{r}_2\cdot\mathbf{v}_2=\frac{1}{2}(1.8-1.1)=0.35$$

$$\gamma_2^*(S_0,S_1)=\frac{1}{2}\mathbf{r}_2\cdot\mathbf{v}_2=\frac{1}{2}(-1.8+1.1)=-0.35$$

$$\gamma_2^*(S_1,S_1)=\frac{1}{2}\mathbf{r}_2\cdot\mathbf{v}_2=\frac{1}{2}(1.8+1.1)=1.45$$

$$\gamma_2^*(S_1,S_0)=\frac{1}{2}\mathbf{r}_2\cdot\mathbf{v}_2=\frac{1}{2}(-1.8-1.1)=-1.45$$

$$\gamma_3^*(S_0,S_0)=\frac{1}{2}\mathbf{r}_3\cdot\mathbf{v}_3=\frac{1}{2}(-1.6+1.6)=0$$

$$\gamma_3^*(S_1,S_0)=\frac{1}{2}\mathbf{r}_3\cdot\mathbf{v}_3=\frac{1}{2}(1.6+1.6)=1.6$$

然后利用式（5.88b）计算对数域前向度量，为：

$$\left[\begin{array}{l}\alpha_{l+1}^*(s)\equiv\ln\alpha_{l+1}(s)=\ln\sum_{s'\in\sigma_l}\gamma_l(s',s)\alpha_l(s')\\ \qquad=\ln\sum_{s'\in\sigma_l}e^{\left[\gamma_l^*(s',s)+\alpha_l^*(s')\right]}\\ \qquad=\max_{s'\in\sigma_l}^*\left[\gamma_l^*(s',s)+\alpha_l^*(s')\right],\ l=0,1,\cdots,K-1\end{array}\right.\qquad(5.88b)$$

$$\alpha_1^*(S_0)=\left[\gamma_0^*(S_0,S_0)+\alpha_0^*(S_0)\right]=-0.45+0=-0.45$$

$$\alpha_1^*(S_1)=\left[\gamma_0^*(S_0,S_1)+\alpha_0^*(S_0)\right]=0.45+0=0.45$$

$$\alpha_2^*(S_0)=\max^*\left\{\left[\gamma_1^*(S_0,S_0)+\alpha_1^*(S_0)\right],\left[\gamma_1^*(S_1,S_0)+\alpha_1^*(S_1)\right]\right\}$$

$$=\max^*\left\{[-0.25-0.45],[0.75+0.45]\right\}$$

$$=\max^*\{-0.7,1.2\}=1.2+\ln(1+e^{-|-1.9|})=1.34$$

$$\alpha_2^*(S_1)=\max^*\left\{\left[\gamma_1^*(S_0,S_1)+\alpha_1^*(S_0)\right],\left[\gamma_1^*(S_1,S_1)+\alpha_1^*(S_1)\right]\right\}$$

$$=\max^*\{-0.2,-0.3\}=-0.2+\ln(1+e^{-|0.1|})=0.44$$

利用式（5.88d）计算对数域后向度量，为：

$$\left[\begin{array}{l}\beta_l^*(s')\equiv\ln\beta_l(s')=\ln\sum_{s\in\sigma_{l+1}}\gamma_l(s',s)\beta_{l+1}(s)\\ \qquad=\ln\sum_{s\in\sigma_{l+1}}e^{\left[\gamma_l^*(s',s)+\beta_{l+1}^*(s)\right]}\\ \qquad=\max_{s\in\sigma_{l+1}}^*\left[\gamma_l^*(s',s)+\beta_{l+1}^*(s)\right],\ l=K-1,K-2,\cdots,0\end{array}\right.\qquad(5.88d)$$

$$\beta_3^*(S_0)=\left[\gamma_3^*(S_0,S_0)+\beta_4^*(S_0)\right]=0+0=0$$

$$\beta_3^*(S_1)=\left[\gamma_3^*(S_1,S_0)+\beta_4^*(S_0)\right]=1.6+0=1.6$$

$$\beta_2^*(S_0)=\max^*\left\{\left[\gamma_2^*(S_0,S_0)+\beta_3^*(S_0)\right],\left[\gamma_2^*(S_0,S_1)+\beta_3^*(S_1)\right]\right\}$$

$$=\max^*\left\{[0.35+0],[-0.35+1.6]\right\}$$

$$=\max^*(0.35,1.25)=1.25+\ln(1+e^{-|-0.9|})=1.59$$

$$\beta_2^*(S_1)=\max^*\left\{\left[\gamma_2^*(S_1,S_0)+\beta_3^*(S_0)\right],\left[\gamma_2^*(S_1,S_1)+\beta_3^*(S_1)\right]\right\}$$

$$=\max^*(-1.45,3.05)=3.05+\ln(1+e^{-|-4.5|})=3.06$$

$$\beta_1^*(S_0)=\max^*\left\{\left[\gamma_1^*(S_0,S_0)+\beta_2^*(S_0)\right],\left[\gamma_1^*(S_0,S_1)+\beta_2^*(S_1)\right]\right\}$$

$$=\max^*(1.34,3.31)=3.44$$

$$\beta_1^*(S_1)=\max^*\left\{\left[\gamma_1^*(S_1,S_0)+\beta_2^*(S_0)\right],\left[\gamma_1^*(S_1,S_1)+\beta_2^*(S_1)\right]\right\}$$

$$=\max^*(2.34,2.31)=3.02$$

最后，利用式（5.92）计算 APP L 值，为：

$$\left[\begin{array}{l}L(u_l)=\max^*_{(s',s)\in\Sigma_l^+}\left[\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')\right]\\ \qquad\qquad -\max^*_{(s',s)\in\Sigma_l^-}\left[\beta_{l+1}^*(s)+\gamma_l^*(s',s)+\alpha_l^*(s')\right]\end{array}\right. \qquad (5.92)$$

$$L(u_0)=\left[\beta_1^*(S_1)+\gamma_0^*(S_0,S_1)+\alpha_0^*(S_0)\right]-\left[\beta_1^*(S_0)+\gamma_0^*(S_0,S_0)+\alpha_0^*(S_0)\right]$$

$$=3.47-2.99=+0.48$$

$$L(u_1)=\max^*\left\{\left[\beta_2^*(S_0)+\gamma_1^*(S_1,S_0)+\alpha_1^*(S_1)\right],\left[\beta_2^*(S_1)+\gamma_1^*(S_0,S_1)+\alpha_1^*(S_0)\right]\right\}$$

$$-\max^*\left\{\left[\beta_2^*(S_0)+\gamma_1^*(S_0,S_0)+\alpha_1^*(S_0)\right],\left[\beta_2^*(S_1)+\gamma_1^*(S_1,S_1)+\alpha_1^*(S_1)\right]\right\}$$

$$=\max^*(2.79,2.86)-\max^*(0.89,2.76)=+0.62$$

$$L(u_2)=\max^*\left\{\left[\beta_3^*(S_0)+\gamma_2^*(S_1,S_0)+\alpha_2^*(S_1)\right],\left[\beta_3^*(S_1)+\gamma_2^*(S_0,S_1)+\alpha_2^*(S_0)\right]\right\}$$

$$-\max^*\left\{\left[\beta_3^*(S_0)+\gamma_2^*(S_0,S_0)+\alpha_2^*(S_0)\right],\left[\beta_3^*(S_1)+\gamma_2^*(S_1,S_1)+\alpha_2^*(S_1)\right]\right\}$$

$$=\max^*(-1.01,2.59)-\max^*(1.69,3.49)=-1.02$$

利用式（5.68），可得到 BCJR 译码器的硬判决输出：

$$\hat{\mathbf{u}} = (+1 \quad +1 \quad -1)$$

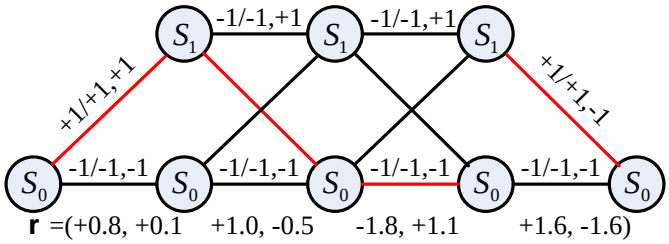
虽然结尾比特 u_3 不是一个信息比特，但我们也可按照前述相同的步骤得到其 APP　L 值。

$$L(u_3) = \max^*\left\{\left[\beta_4^*(S_0)+\gamma_3^*(S_1,S_0)+\alpha_3^*(S_1)\right],\left[\beta_4^*(S_1)+\gamma_3^*(S_0,S_1)+\alpha_3^*(S_0)\right]\right\}$$

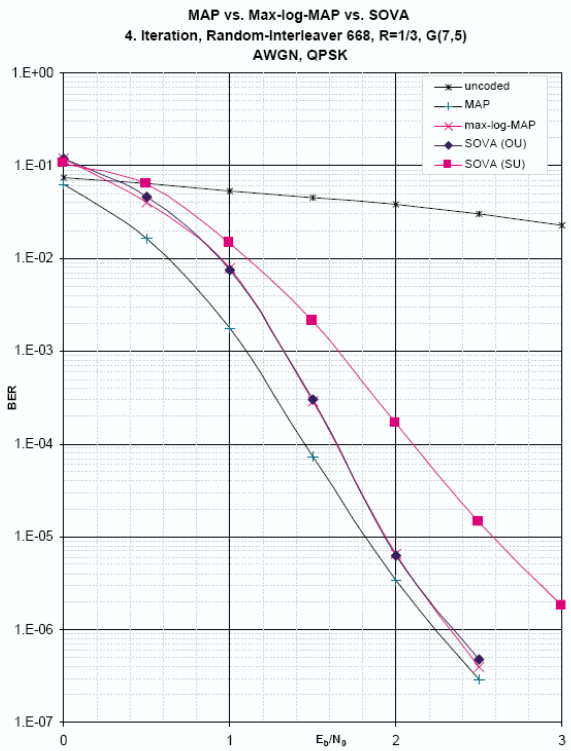
$$-\max^*\left\{\left[\beta_4^*(S_0)+\gamma_3^*(S_0,S_0)+\alpha_3^*(S_0)\right],\left[\beta_4^*(S_1)+\gamma_3^*(S_1,S_1)+\alpha_3^*(S_1)\right]\right\}$$

$$= \max^*(1.6,0)-\max^*(0,0) = +1.09$$

这样，u_3 就判决为＋1，根据网格图我们知道这是不可能的，显然中间传输过程中出错了。



这是因为译码器工作在 E_b/N_0＝-1.76dB，而编码速率 R＝3/8 的 Shannon 限为 E_b/N_0＝-0.33dB。



J. Vogt, K. Koora, A. Finger and G. Fettweis. COMPARISON OF DIFFERENT TURBO DECODER REALIZATIONS FOR IMT-2000. Globecom'99，p2704～2708.