第七章 编织码

2017/5/2

本章内容

- 第七章 编织码
  - 7.1 编织码编码基本原理
  - 7.2 编织码的译码

第七章 编织码

- 编织码（Woven Code）是在编织卷积码（1997年由三位欧洲学者提出）的基础上发展起来的一种组合码。最早的组合编码是级联码，由Forney提出，其目的是找到一类纠错码及其译码算法，在R<C的情况下使得差错概率随码长呈指数下降，而译码复杂度呈代数关系增加，从而获得高的编码增益。由于使用了两个或更多相对简单的成员码来构成级联码，因此，其纠错性能和许多非级联码的长码相当。常见结构形式是内码采用约束长度短的卷积码和最大似然译码算法，外码采用高码率的多进制RS码和代数译码结构。
- Turbo码的出现提供了一个性能优越的编码方法，其思想精髓被推广到其他的编码方案中。1998年，Benedntto提出了以卷积码为成员码的串行级联卷积码（SCCC），该码保持了Turbo码在低信噪比时的最大似然译码性能，同时消除了Turbo码在高信噪比时的“误码平台”效应。除了以卷积码为分量码外，还可以用分组码作分量码，Pyndiah提出了分组Turbo码。于是，在这样的思路引导下，Host提出了编织卷积码（WCC），又推广到更一般的编织码，其系统结构可完全包容传统分组码、卷积码以及各类Turbo码。

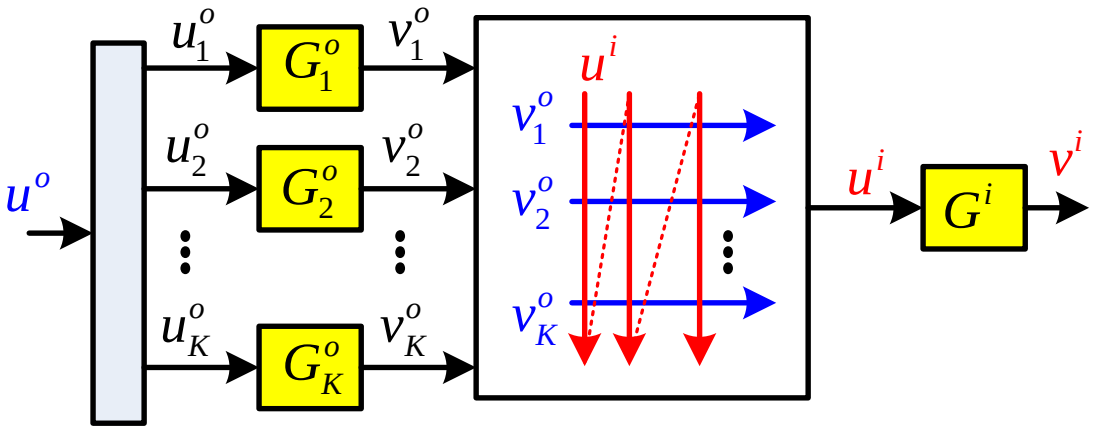
7.1 编织码编码基本原理

- 编织码的基本原理类似于纺织业中的织布原理，其编码结构主要有三种：外经（**Outer Warp**）、内经（**Inner Warp**）和斜纹（**Twill**）结构。它们是指外编码器的输出码字比特在缓冲器中以按列读出（外经）或按行读出（内经）输入到各个内编码器。

1. Host's Ph.D thesis：On Woven Convolutional Codes. 1999
2. Host, etc. Woven Convolutional Codes I: Encoder Properties. IEEE Trans. On Info. Theory. Vol.48, No.1. 2002. p149—161.
3. Jordan, etc. Woven Convolutional Codes II: Decoding Aspects. IEEE Trans. On Info. Theory. Vol.50, No.10. 2004. p2522—2529.

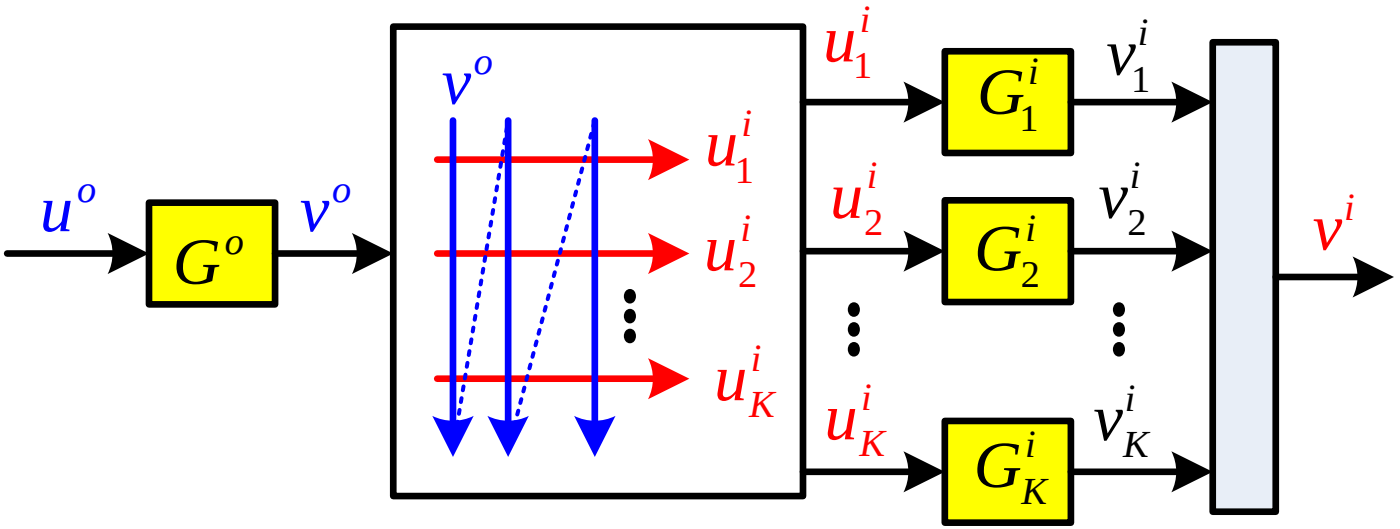
外经结构编码器

- 外经编码器结构如下图所示，假定外编码器由K个普通二进制编码器并行级联组成，所有成员编码器生成矩阵 G_k^o（k＝1, 2, …, K）可以相同，也可以不同。内编码器由一个普通二进制编码器构成，其生成矩阵为G^i。假定外编码器的每个子编码器的码率为$R_o=b_o/c_o$，即输入信息比特数为b_o，输出序列长为c_o。u被分为K个b_o长度的数据块，即图中的 $u=\left(u^{(1)}u^{(2)}\cdots u^{(b_o)}\right)$，每个 $u^{(i)}=\left(u_1^{(i)}u_2^{(i)}\cdots u_K^{(i)}\right),i=1,...,b_o$，第k个成员编码器的信息序列就为 $u_k^o=\left(u_k^{(1)}u_k^{(2)}\cdots u_k^{(b_o)}\right)$，这些信息子块再以并行的方式输入到K个并联的子编码器中，每个子编码器的输出序列（长度为c_o）以串行方式按行写入缓冲器（共K行），缓冲器的输出按列读出，作为内编码器（编码速率为$R_i=b_i/c_i$）的输入信息序列，此结构的总编码速率为$R=R_oR_i$。



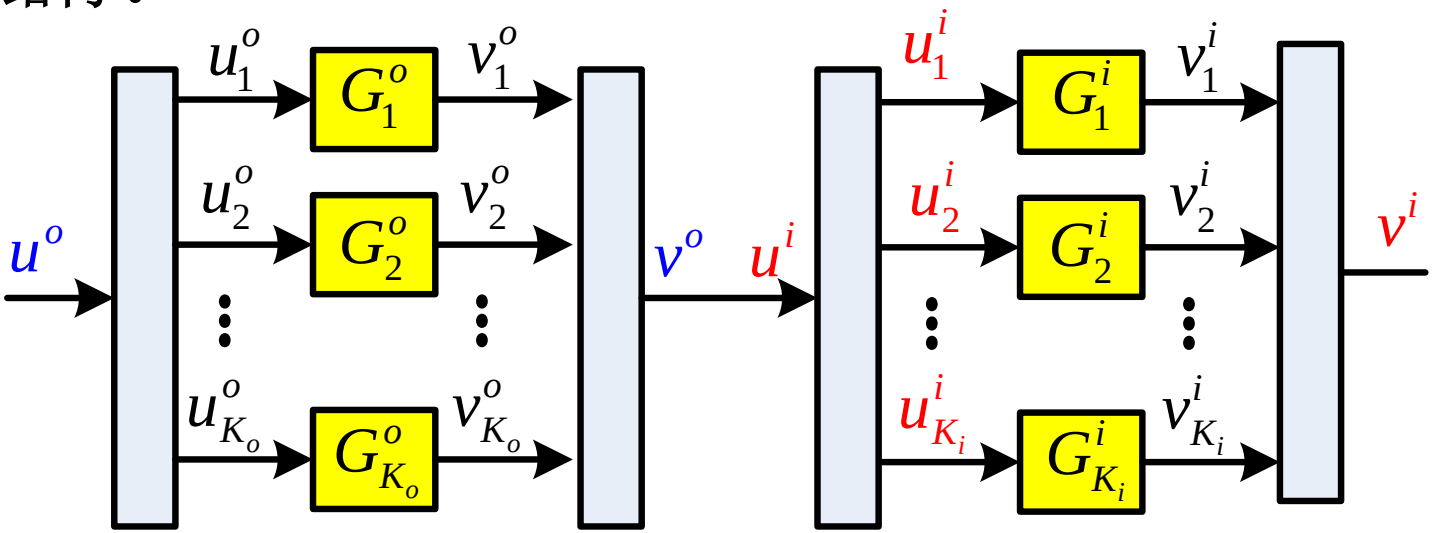
内经结构编码器

- 与外经结构相反，内经结构中是由一个二进制编码器作为外编码器，K个并联二进制编码器构成内编码器。外编码器的输出码序列按列写入缓冲器（共K行），作为K个并联内编码器的信息序列。外编码器的编码速率为$R_o=b_o/c_o$，内编码器的编码速率为$R_i=b_i/c_i$，则内经编码器的总的编码速率为$R=R_oR_i$。



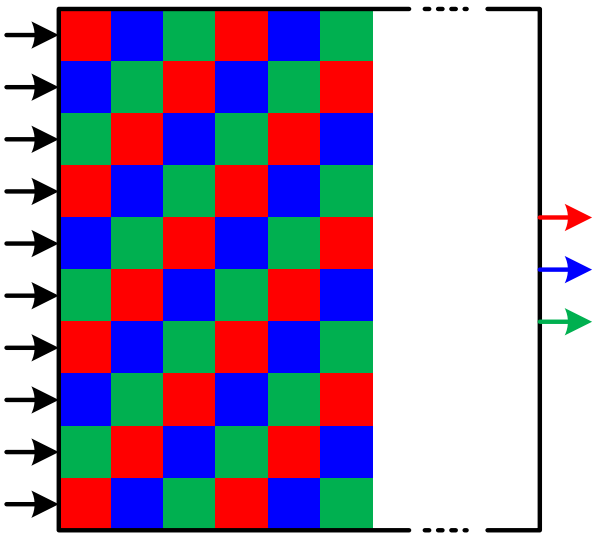
斜纹结构编码器

- 斜纹结构是前两种结构的综合，如下图所示。外编码器由K_o个码率为$R_o=b_o/c_o$的二进制编码器并联组成，内编码器由K_i个码率为$R_i=b_i/c_i$的二进制编码器并联组成。信息序列u^o被分为$K_oK_ib_o$长的子块，这些子块输入到K_o个并联的外编码器，输出按行写入缓冲器，按列读出作为内编码器的信息序列，输入到K_i个并联的内编码器。斜纹编码器的总的编码速率为$R=R_oR_i$。
- 显然，外经结构和内经结构都是斜纹结构的特殊情况。当$K_i=1$及$K_o>1$时为外经结构，当$K_i>1$及$K_o=1$时为内经结构，当$K_i=1$及$K_o=1$时为通常的级联码结构 。



- 在斜纹结构中，K_o和K_i的值应满足：$gcd(K_o,K_i)=1$，如不满足，则编织码的经纬不能构成斜纹。

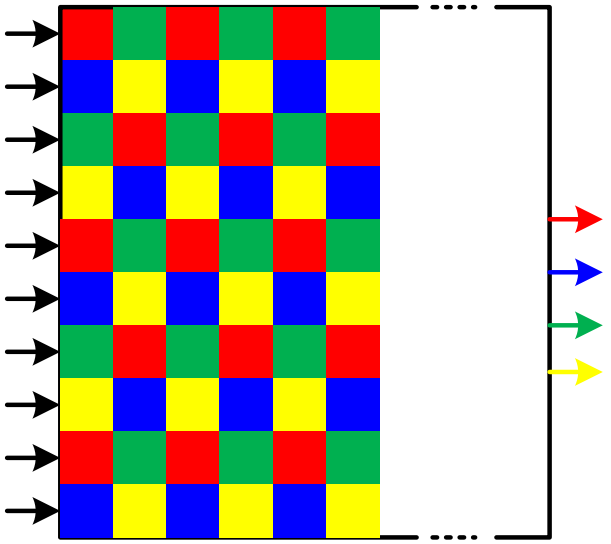
例如$K_o=10$，$K_i=3$时，内编码器和外编码器间缓冲器中信息分布如图（a）所示。



（a）

左边10个箭头表示外编码器，右边3个箭头表示内编码器，缓冲器中来自外编码器的码流按行写入，读出按列进行。即第一个码比特输入到第一个内编码器作为它的信息比特，第二个码比特输入到第二个内编码器，第三个码比特输入到第三个内编码器，第四个码比特输入到第一个内编码器，依次类推。可见，输入到相应内编码器的比特序列在缓冲器中组成图（a）中的斜纹结构。

例如$K_o=10$，$K_i=4$时，内编码器和外编码器间缓冲器中信息分布如图（b）所示。



（b）

形成不了斜纹结构！相当于两个$K_o=5$, $K_i=2$的斜纹编码器的并联。

编织码的分类

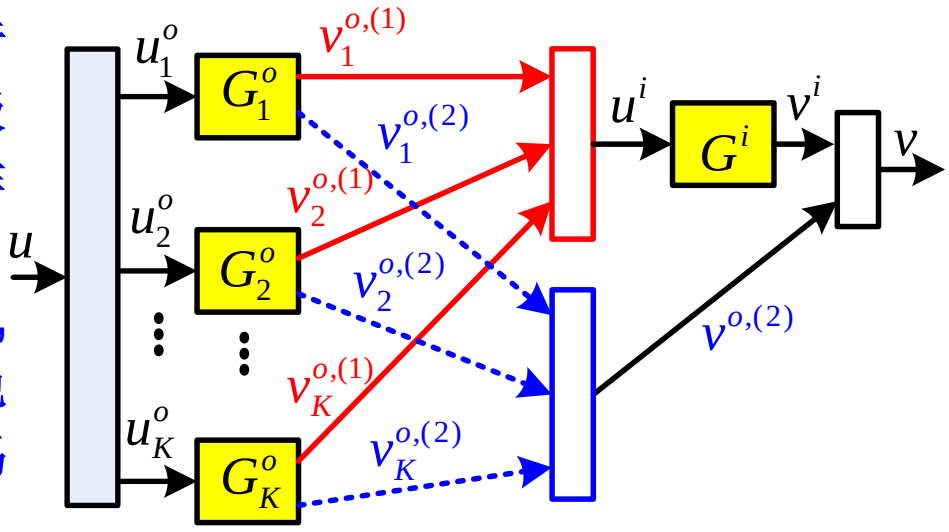
- 编织码的包容性很强，具体体现在分量码型上，按其分量码型的组成可分为两大类：一是其分量码为单一码型的，即内外码采用同样的码型；二是其分量码为混合码型，即内外码型不同。无论是哪一类分量码型，经过编织后所构成的新码型（编织码），其距离都得到了提高（即纠错能力提高）。不同的码型采用不同的交织器。

（1）编织卷积码（WCC，Woven Convolutional Code）

WCC是串行级联卷级码，其内外编码器都是卷级码，总体上可视为一个卷级码，因此可以用卷级码的性质来分析和优化。（研究最为广泛）

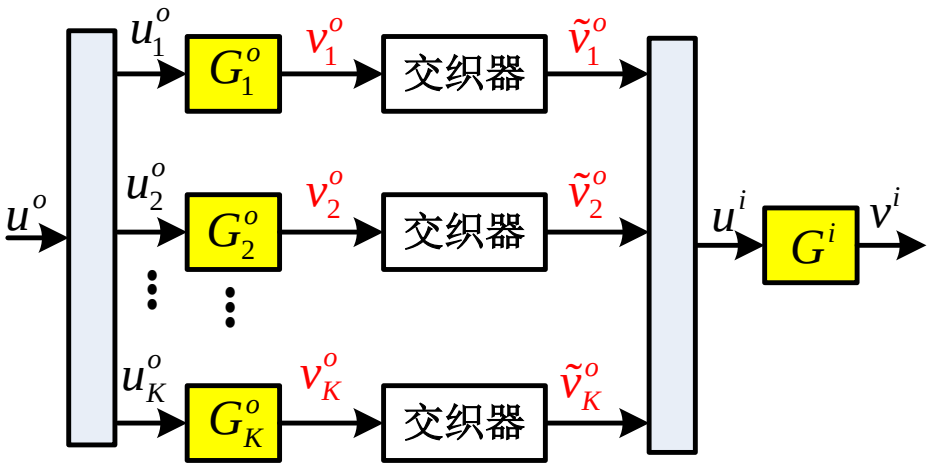
（2）编织Turbo码（WTC）

WTC的编码器结构如图所示，它由K个并行外编码器和一个内编码器组成（都是卷级码）。信息序列u被分为K个子块，分别作为K个码率为$R_o=b_o/c_o$的外编码器的输入，外编码器的部分输出序列 $v_k^{o,(1)}$ (k =1, 2, …,K)组成内编码器的输入序列u_i，其他输出序列 $v_k^{o,(2)}$ (k=1,2,…,K)直接与内编码器的输出vi一起组成最终的编织码v。



（3）编织分组码（WBC）

WBC编码器如下图所示，外码编码器由K个分组码编码器或者卷级码编码器组成，内码编码器由一个卷级码编码器构成，交织器按行交织。



（4）串行级联码（SCC, Serial Cascaded Code）

SCC编码器如下图所示，由内外码编码器及交织器级联组成，它可看成编织码的特例。

7.2 编织码的译码

- 以编织卷积码为例，讲解译码原理。WCC的译码与其编码特性有关，由于WCC为组合编码，其译码可以采用Turbo码的译码方式，将各个成员码分离出来分别进行译码，最后再将各译码器的值按一定的方式组合，就可得到最终的译码结果；也可采用内外译码器迭代的方式来提高译码的准确性。
- WCC的译码主要有三种：传统的软输入软输出Viterbi译码（SOVA）[1]、SW-BCJR译码算法[2]和pipeline译码算法[3]。

1. S. Benedetto, etc. A soft-input soft-output APP module for iterative decoding of concatenated codes. IEEE Communications letters. Vol.1, No.1, 1997. p22-24.
2. S. Benedetto, etc. Soft-output Decoding Algorithms in Iterative Decoding of Turbo Codes. TDA Progress Report. 1996. p63-87.
3. R. Jordan, etc. Pipeline Decoding of Woven convolutional codes. Proc. 3rd ITG Conf. Sour ce, Channel Coding. Munich, 2000.

BCJR算法

- 为了应用迭代译码，我们需要对成员码进行软输出译码（如**BCJR**和**SOVA**），我们主要讲解**BCJR**算法。
- 假设接收序列为r，信息序列或码字序列的先验概率为 $P\left(u_t^{(j)}\right)$或 $P\left(v_t^{(j)}\right)$，我们可以计算出对应的后验概率 $P\left(u_t^{(j)} \mid \mathrm{r}\right)$ 或 $P\left(v_t^{(j)} \mid \mathrm{r}\right)$，如下图所示。

定义信息序列为$u_{[0,T)}=(u_0,u_1,\ldots,u_{T-1})$，每个信元$u_t$是一个二进制**b-tuple**，**T**是帧长，对应的码字序列为$v_{[0,T)}=(v_0,v_1,\ldots,v_{T-1})$，其中$v_t$是一个二进制**c-tuple**，即编码效率为**b/c**；状态序列为$S_{[0,T]}=(\sigma_0, \sigma_1,\ldots, \sigma_T)$，状态$\sigma_t$是一个二进制矩阵，表示编码器在**t**时刻的物理状态，我们用$\mathbb{S}$来表示所有可能的状态集合；这样接收序列可表示成$r_{[0,T)}=(r_0,r_1,\ldots,r_{T-1})$，其中每个接收到的信元$r_t$都是一个**c-tuple**的信道信元。

Host's Ph.D thesis: On Woven Convolutional Codes. 1999

- 假设信道是无记忆的AWGN信道，给定码字序列的情况下，接收序列$r_{[0,T)}$的概率为：

$$P(\mathrm{r}_{[0,T)} \mid \mathrm{v}_{[0,T)}) = \prod_{t=0}^{T-1} P(\mathrm{r}_t \mid \mathrm{v}_t) = \prod_{t=0}^{T-1} \prod_{j=1}^{c} P\left(r_t^{(j)} \mid v_t^{(j)}\right)$$

其中 $P\left(r_t^{(j)} \mid v_t^{(j)}\right)$ 是信道转移概率。

已知信息或码序列的先验概率，很容易得到网格图中各分支的先验概率 $P(S_{t+1} = \sigma' \mid S_t = \sigma)$ 或 $P(S_t = \sigma \mid S_{t+1} = \sigma')$。考虑一个分支：t时刻在状态$\sigma$，t＋1时刻在状态 σ'。

初始条件$P(S_0=\sigma)=1/|\mathbb{S}|$，其中$|\mathbb{S}|$表示状态数，有$P(S_t=\sigma)=1/|\mathbb{S}|$，t=1,2,…，则有 $P(S_{t+1} = \sigma' \mid S_t = \sigma) = P(S_t = \sigma \mid S_{t+1} = \sigma')$

为了简化表示，引入4个变量：

$$\alpha_t(\sigma) \triangleq P\left(S_t = \sigma \mid r_{[0,t)}\right)$$

$$\beta_t(\sigma) \triangleq P\left(S_t = \sigma \mid r_{[t,T)}\right)$$

$$\gamma_t(\sigma,\sigma') \triangleq P\left(S_{t+1} = \sigma'; r_t \mid S_t = \sigma\right)$$

$$\delta_t(\sigma,\sigma') \triangleq P\left(S_{t+1} = \sigma'; S_t = \sigma \mid r_{[0,T)}\right)$$

- 得到前面的公式后，继而就可得到后验概率，为：

$$P\left(u_t^{(i)}=u \mid r_{[0,T)}\right)=\sum_{(\sigma,\sigma'):\sigma \xrightarrow{u} \sigma'} P(S_t=\sigma;S_{t+1}=\sigma' \mid r_{[0,T)})$$

$$=\sum_{(\sigma,\sigma'):\sigma \xrightarrow{u} \sigma'} \delta_t(\sigma,\sigma')$$

及 $$P\left(v_t^{(i)}=v \mid r_{[0,T)}\right)=\sum_{(\sigma,\sigma'):\sigma \xrightarrow{v} \sigma'} P(S_t=\sigma;S_{t+1}=\sigma' \mid r_{[0,T)})$$

$$=\sum_{(\sigma,\sigma'):\sigma \xrightarrow{v} \sigma'} \delta_t(\sigma,\sigma')$$

$(\sigma,\sigma'):\sigma \xrightarrow{u} \sigma'$表示 σ 和 σ' 取自状态对集合，并在网格图中存在一条分支：在t时刻状态为 σ ，当第*i*个信息信元为 $u_t^{(i)}=u$ 时，转移到状态 σ'（t+1时刻）。 $(\sigma,\sigma'):\sigma \xrightarrow{u} \sigma'$ 同理。

$\sigma:\sigma \to \sigma'$ 表示t时刻的状态集合，这些状态必须满足在t+1时刻转移到状态 σ' 这个条件。 $\sigma':\sigma \to \sigma'$ 表示t+1时刻的状态集合，这些状态必须满足在时刻t状态为 σ 的条件。

计算各度量值：

$$\gamma_t(\sigma,\sigma') = P(S_{t+1}=\sigma'; r_t \mid S_t=\sigma)$$
$$= P(r_t \mid S_t=\sigma; S_{t+1}=\sigma')P(S_{t+1}=\sigma' \mid S_t=\sigma)$$
$$= P\left(r_t \mid v(\sigma,\sigma')\right)P(S_{t+1}=\sigma' \mid S_t=\sigma)$$

分支度量！

其中 $v(\sigma,\sigma')$ 是t时刻状态为 σ 、t+1时刻状态为 σ' 这条分支对应的码字。

如果在时刻 t，$\alpha_t(\sigma)$ 的值已知，就可推导出 t+1 时刻的值，为：

$$\alpha_{t+1}(\sigma') = P(S_{t+1}=\sigma' \mid r_{[0,t+1)}) = \sum_{\sigma:\sigma\to\sigma'} P(S_{t+1}=\sigma'; S_t=\sigma \mid r_{[0,t)}; r_t)$$

$$= \frac{1}{P(r_t \mid r_{[0,t)})} \sum_{\sigma:\sigma\to\sigma'} P(S_{t+1}=\sigma'; S_t=\sigma; r_t \mid r_{[0,t)})$$

$$= k_{t+1}^{\alpha} \sum_{\sigma:\sigma\to\sigma'} P(S_t=\sigma \mid r_{[0,t)})P(S_{t+1}=\sigma'; r_t \mid S_t=\sigma) = k_{t+1}^{\alpha} \sum_{\sigma:\sigma\to\sigma'} \alpha_t(\sigma)\gamma_t(\sigma,\sigma')$$

前向状态度量！

如果在时刻 **t+1**，$\beta_{t+1}(\sigma')$ 的值已知，就可推导出 **t** 时刻的值，为：

$$\beta_t(\sigma) = P(S_t = \sigma \mid r_{[t,T)})$$

$$= \sum_{\sigma':\sigma\to\sigma'} P(S_t = \sigma; S_{t+1} = \sigma' \mid r_t; r_{[t+1,T)})$$

$$= \frac{1}{P(r_t \mid r_{[t+1,T)})} \sum_{\sigma':\sigma\to\sigma'} P(S_t = \sigma; S_{t+1} = \sigma'; r_t \mid r_{[t+1,T)})$$

$$= k_t^{\beta} \sum_{\sigma':\sigma\to\sigma'} P(S_{t+1} = \sigma' \mid r_{[t+1,T)}) P(S_{t+1} = \sigma'; r_t \mid S_t = \sigma)$$

$$= k_t^{\beta} \sum_{\sigma':\sigma\to\sigma'} \beta_{t+1}(\sigma')\gamma_t(\sigma,\sigma')$$

反向状态度量！

对于由状态 $(S_t, S_{t+1}) = (\sigma, \sigma')$ 定义的支路， $\alpha_t(\sigma)$表示到t时刻的序列的后验信息，$\beta_{t+1}(\sigma')$ 表示t+1时刻及之后的序列的后验信息，状态之间的支路信息在 $\gamma_t(\sigma, \sigma')$中。知道了这些信息，就可计算 $\delta_t(\sigma, \sigma')$ ，为：

$$\delta_t(\sigma, \sigma') = P(S_t = \sigma; S_{t+1} = \sigma' \mid r_{[0,T)})$$

$$= \frac{1}{P(r_{[0,T)})} P(S_t = \sigma; r_{[0,t)}) P(S_{t+1} = \sigma'; r_{[t,T)} \mid S_t = \sigma)$$

$$= \frac{P(r_{[0,t)})}{P(r_{[0,T)})} \alpha_t(\sigma) P(S_{t+1} = \sigma'; r_t \mid S_t = \sigma) P(r_{[t+1,T)} \mid S_{t+1} = \sigma')$$

$$= k_t^{\delta} \alpha_t(\sigma) \gamma_t(\sigma, \sigma') \beta_{t+1}(\sigma')$$

得到后验信息！

BCJR算法归纳

Initialization:

$$\alpha_0(\mathbf{0}) \leftarrow 1;\ \beta_T(\mathbf{0}) \leftarrow 1$$
$$\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \leftarrow P(\boldsymbol{r}_t | v(\boldsymbol{\sigma}, \boldsymbol{\sigma}'))P(S_{t+1} = \boldsymbol{\sigma}' | S_t = \boldsymbol{\sigma}),$$
$$0 \leqslant t < T;\ (\boldsymbol{\sigma}, \boldsymbol{\sigma}') \in \mathcal{S} \times \mathcal{S}$$

Forward recursion: t from 0 to $T-1$; $\boldsymbol{\sigma} \in \mathcal{S}$;

$$\alpha_{t+1}(\boldsymbol{\sigma}') \leftarrow k_{t+1}^{\alpha} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} \alpha_t(\boldsymbol{\sigma})\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$$

Backward recursion: t from $T-1$ to 0; $\boldsymbol{\sigma} \in \mathcal{S}$;

$$\beta_t(\boldsymbol{\sigma}) \leftarrow k_t^{\beta} \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} \beta_{t+1}(\boldsymbol{\sigma}')\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$$

Gather information: $0 \leqslant t < T;\ (\boldsymbol{\sigma}, \boldsymbol{\sigma}') \in \mathcal{S} \times \mathcal{S}$;

$$\delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \leftarrow k_t^{\sigma} \alpha_t(\boldsymbol{\sigma})\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')\beta_{t+1}(\boldsymbol{\sigma}')$$

Get *a posteriori* information $0 \leqslant t < T$

$$P(u_t^{(j)} = u | \boldsymbol{r}_{[0,T)}) \leftarrow \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{u}\boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}'),\ 1 \leqslant j \leqslant b$$
$$P(v_t^{(j)} = v | \boldsymbol{r}_{[0,T)}) \leftarrow \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{v}\boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}'),\ 1 \leqslant j \leqslant c$$