Códigos Convolucionales y Algoritmo Viterbi

Notas del curso Tecnología de Servicios Audiovisuales

Pablo Flores Guridi

21 de octubre de 2019

Índice

1. Códigos Convolucionales

1.1. Introducción

Introducidos por Elias [1] en 1955, los códigos convolucionales son un tipo de códigos correctores de errores caracterizados por la terna (n, m, k). Donde n es el número de entradas al codificador, m es el número de entradas y k la profundidad del código (las salidas serán definidas por la entrada actual y las $k-1$ entradas previas). En general n y m son enteros pequeños con $n < m$, pero la memoria k debe ser lo suficientemente larga para alcanzar bajas probabilidades de error, pero no tanto como para requerir demasiado procesamiento en el decodificador. Se denomina *tasa del código* a la relación entre número de entradas y número de salidas n/m; de alguna manera esta tasa nos indica qué proporción de los bits transmitidos serán efectivamente información y no tan sólo redundancia.

1.2. Representación de los Códigos Convolucionales

1.2.1. Diagramas de Conexiones

Más adelante veremos que los códigos convolucionales deben su nombre a que no son más que una convolución de cada una de las n palabras de entrada con ciertas *respuestas al impulso* que definirán cada una de las m salidas. Pero para llegar a comprender esto, primero veamos en la figura 1.1 el diagrama de conexiones de un código convolucional binario $(1, 2, 4)$.

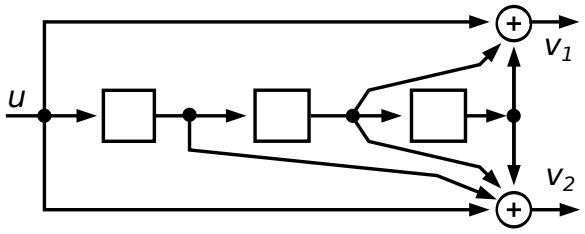


Figura 1.1: Diagrama de conexiones de un código convolucional binario $(1, 2, 4)$.

Véase que cuenta con una entrada **u**, tres registros de desplazamiento y dos sumadores mod-2 que definirán cada una de las dos salidas $\mathbf{v}_1$ y $\mathbf{v}_2$: se trata entonces de un código convolucional de tasa $\frac{1}{2}$. Los sumadores mod-2 son implementados con compuertas XOR. Otra forma comúnmente utilizada para representar al diagrama de conexión de la figura 1.1 es como en la figura 1.2.

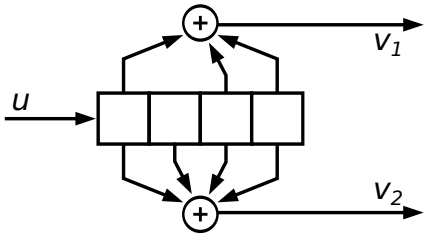


Figura 1.2: Otro diagrama de conexiones del mismo código convolucional binario $(1, 2, 4)$ de la figura 1.1.

Como la suma mod-2 es una operación lineal, estos códigos pueden ser vistos como sistemas lineales, es decir, podemos encontrar una matriz **G** tal que la salida **v** sea lograda a partir de la entrada **u** según

$$\mathbf{v} = \mathbf{uG}. \tag{1.1}$$

En los diagramas de conexión de las figuras 1.1 y 1.2, los bits de la secuencia **u** van entrando uno a uno y se van obteniendo las salidas $\mathbf{v}_1$ y $\mathbf{v}_2$. La primera salida se obtiene de la suma mod-2 del bit entrante con los bits en $t-2$ y $t-3$. La segunda salida se obtiene de la suma mod-2 del bit entrante con los bits en $t-1$, $t-2$ y $t-3$. Finalmente, $\mathbf{v}_1$ y $\mathbf{v}_2$ se multiplexan logrando un único flujo de salida.

Ahora bien, si a la entrada colocamos una delta de Kronecker, i.e. $\mathbf{u} = (1, 0, 0, \dots)$, y observamos las salidas, obtendremos las respuestas al impulso $\mathbf{h}_1 = (1, 0, 1, 1)$ y $\mathbf{h}_2 = (1, 1, 1, 1)$ correspondientes a las salidas $\mathbf{v}_1$ y $\mathbf{v}_2$ respectivamente. Y como el sistema es lineal, se cumple que

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{u} * \mathbf{h}_1, \\ \mathbf{v}_2 &= \mathbf{u} * \mathbf{h}_2. \end{aligned} \tag{1.2}$$

Luego, no es difícil ver cómo para este código convolucional $(1, 2, 4)$ la matriz $\mathbf{G}$ de la ecuación (1.1) se logra según

$$\mathbf{G} = \begin{pmatrix} h_1[0] & h_2[0] & h_1[1] & h_2[1] & h_1[2] & h_2[2] & h_1[3] & h_2[3] & 0 & 0 & 0 & \dots \\ 0 & h_1[0] & h_2[0] & h_1[1] & h_2[1] & h_1[2] & h_2[2] & h_1[3] & h_2[3] & 0 & 0 & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \dots & 0 & 0 & h_1[0] & h_2[0] & h_1[1] & h_2[1] & h_1[2] & h_2[2] & h_1[3] & h_2[3] & 0 \\ \dots & 0 & 0 & 0 & h_1[0] & h_2[0] & h_1[1] & h_2[1] & h_1[2] & h_2[2] & h_1[3] & h_2[3] \end{pmatrix} \tag{1.3}$$

Finalmente, en cualquier sistema lineal, operaciones en el dominio del tiempo que involucran convoluciones pueden ser reemplazadas por operaciones en cierto espacio de transformación que involucran operaciones con polinomios. En particular, aplicando la Transformada-Z a la ecuación (1.2) obtenemos que

$$\begin{aligned} V_1(Z) = \quad & U(Z)H_1(Z) = \quad U(Z) \times (1 + Z^{-2} + Z^{-3}), \\ V_2(Z) = \quad & U(Z)H_2(Z) = \quad U(Z) \times (1 + Z^{-1} + Z^{-2} + Z^{-3}), \end{aligned} \tag{1.4}$$

donde todas las operaciones son mod-2 y

$$U(Z) = u_0 + u_1 Z^{-1} + u_2 Z^{-2} + \dots; \tag{1.5}$$

la multiplexación de ambas salidas en un único flujo se obtiene según

$$V(Z) = V_1(Z^2) + Z^{-1} V_2(Z^2). \tag{1.6}$$

Nótese que el evaluar tanto a V_1 como a V_2 en Z^2 y no en Z sugiere que al multiplexar ambas salidas lo que se está haciendo es un cambio en la tasa de bits, un sobre muestreo racional igual m/n. Finalmente, vale la pena aclarar que al trabajar con representaciones polinómicas de códigos convolucionales no se acostumbra hacerlo con la Transformada-Z, sino que se utiliza la llamada "Transmformada-D" que si bien es totalmente análoga, utiliza una "D" en vez de una Z^{-1} para denotar un retardo.

Todo lo presentado hasta este punto puede ser extendido a códigos con diferente número de entradas n, diferente número de salidas m, diferente cantidad de registros de memoria $k - 1$ y diferente diagrama de conexión. Pero como realizar una descripción exhaustiva para cada caso en particular no aporta grandes conceptos en general, la presente descripción continuará basándose en el código convolucional definido en las figuras 1.1 y 1.2. Para ver otros ejemplos y descripciones más detalladas se recomienda consultar [2].

Ejemplo 1.1. *Sea la secuencia a codificar* $\boldsymbol{u} = (10111)$. *Para el código convolucional de las figuras 1.1 y 1.2 se puede ver que la salida será* $\boldsymbol{v} = (1101000101010011)$. *Este resultado también puede obtenerse con la ecuaciones (1.1) y (1.2) para las respuestas al impulso* $\boldsymbol{h}_1 = (1, 0, 1, 1)$ *y* $\boldsymbol{h}_2 = (1, 1, 1, 1)$.

1.2.2. Diagramas de Estados

Como un codificador convolucional es un circuito secuencial, su operación puede ser descrita por un diagrama de estados. El estado del codificador es definido por el contenido de sus registros de desplazamiento; y en función del valor de la entrada, podremos saber el valor del próximo estado. El diagrama de estados del codificador convolucional de las figuras 1.1 y 1.2 se muestra en la figura 1.3.

Los estados son etiquetados según $S_0, S_1, \dots, S_{2^{(k-1)n}-1}$. Nótese que si el número de entradas es $n = 1$, como sucede en este ejemplo, se tiene que los estados serán $S_0, S_1, \dots, S_{2^{k-1}-1}$; con $k-1$ el número de registros de desplazamiento. Las flechas que llevan de un estado a otro en la figura 1.3 dan información de la entrada necesaria para tomar ese camino dado el estado presente, y de los dos bits de salida una vez llevada a cabo la transición. Asumiendo que el estado inicial del diagrama es con todos sus registros en cero (S_0), es posible obtener la palabra de código correspondiente a cualquier secuencia de entrada $\mathbf{u}$ simplemente recorriendo el diagrama de estados.

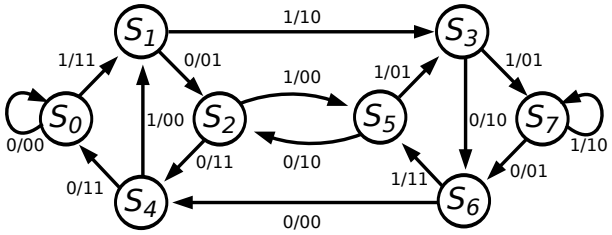


Figura 1.3: Diagrama de estados del código convolucional binario $(1, 2, 4)$ de las figuras 1.1 y 1.2.

Ejemplo 1.2. *Sea la secuencia a codificar* $\boldsymbol{u} = (1\,0\,1\,1\,1)$. *Para el código convolucional expresado en el diagrama de estados de la figura 1.3, se puede ver que la salida será* $\boldsymbol{v} = (1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$. *Ver que este resultado coincide con el del ejemplo 1.1.*

1.3. *Puncturing*

Puncturing es una técnica utilizada para lograr códigos de cualquier tasa n/m en transmisión, sin aumentar la complejidad de la decodificación en recepción. Como se verá más adelante, al aumentar el número de entradas n, se incrementa mucho la complejidad del algoritmo de decodificación Viterbi, que es el más comúnmente utilizado con códigos convolucionales.

Esta técnica codifica las palabras con un codificador de menor complejidad denominado "madre", que se caracteriza por contar con tan sólo una entrada y m_{madre} salidas; tendrá por lo tanto una tasa $1/m_{madre}$. Luego, se eliminan algunos de los bits codificados para lograr la tasa del código objetivo. Tanto el codificador como el decodificador saben perfectamente cuáles fueron los bits no transmitidos porque para eso se usa cierta *matriz de puncturing* que indica cuáles son los bits a remover. La información descartada es luego agregada en recepción en forma de bits *dummy* y posteriormente "corregida" por el algoritmo de decodificación. Lo interesante del *puncturing* es que la decodificación del nuevo código es idéntica a la del código madre y por lo tanto, mucho más simple.

A modo de ejemplo, las matrices presentadas en la tabla 1.1 son algunas de las más comúnmente utilizadas y en particular, son las definidas en el estándar de televisión digital terrestre ISDB-T [3]. Para cada una de las salidas v_1 y v_2, los ceros representan bits descartados y los unos bits transmitidos. En la columna de la derecha de la tabla se muestra la secuencia final transmitida.

Tasa del código	Matriz de *puncturing*	Secuencia de transmisión
1/2	$v_1 : 1$ $v_2 : 1$	$v_1[0], v_2[0]$
2/3	$v_1 : 10$ $v_2 : 11$	$v_1[0], v_2[0], v_2[1]$
3/4	$v_1 : 101$ $v_2 : 110$	$v_1[0], v_2[0], v_2[1], v_1[2]$
5/6	$v_1 : 10101$ $v_2 : 11010$	$v_1[0], v_2[0], v_2[1], v_1[2], v_2[3], v_1[4]$
7/8	$v_1 : 1000101$ $v_2 : 1111010$	$v_1[0], v_2[0], v_2[1], v_2[2], v_2[3], v_1[4], v_2[5], v_1[6]$

Tabla 1.1: Matrices de *puncturing* más comúnmente utilizadas, y definidas en el estándar de televisión digital terrestre ISDB-T [3].

2. Decodificador Viterbi

2.1. Introducción

Luego de introducidos los códigos convolucionales, algunos trabajos posteriores propusieron distintas maneras de decodificarlos. En 1967 Viterbi [4] propuso un algoritmo de decodificación relativamente fácil de implementar para códigos con órdenes de memoria relativamente bajos. Este algoritmo fue denominado Decodificación Viterbi.

Finalmente, Forney [5, 6] demostró que la Decodificación Viterbi se trataba de un algoritmo de máxima verosimilitud para códigos convolucionales. Esto quiere decir que la salida del decodificador siempre será la palabra de código que maximice la función log-similitud, que será definida en esta sección.

Este algoritmo de decodificación de máxima verosimilitud hizo que los códigos convolucionales comenzaran a ser cada vez más populares.

2.2. El Diagrama de Trellis

Si reescribimos el diagrama de estados de la figura 1.3 evolucionando en el tiempo, obtenemos el diagrama de la figura 2.1. A este tipo de diagramas de estado expresados en el tiempo se los denomina *diagrama de trellis*.

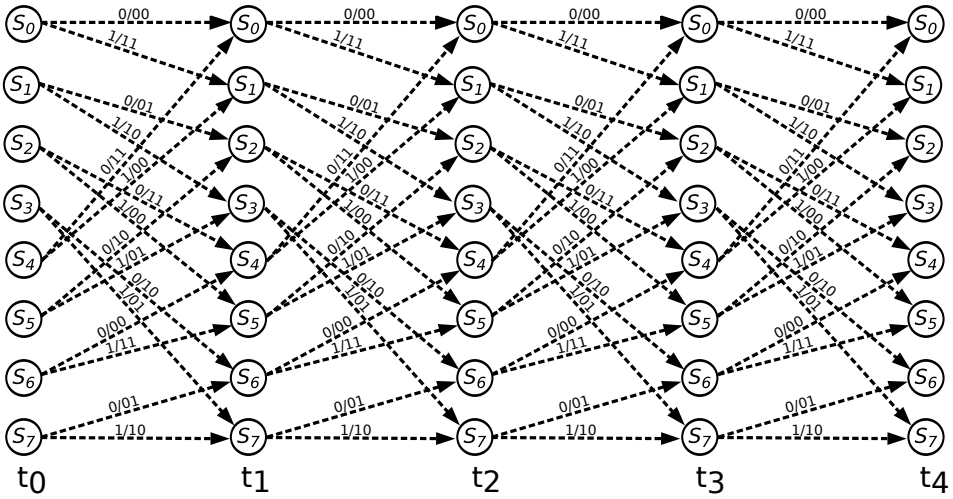


Figura 2.1: Diagrama de trellis del código convolucional binario $(1, 2, 4)$ de las figuras 1.1 y 1.2.

Véase cómo para cada nuevo instante de tiempo, dependiendo del valor de la entrada, el diagrama puede tomar uno u otro camino. Además, para cada camino se especifican la entrada y las respectivas salidas. Cada una de las palabras de código se representa con un único camino por el trellis. Los diagramas de trellis pueden ser realizados para la cantidad de estados, salidas y entradas que se desee. Claramente, conforme estas cantidades aumentan, también lo hace la complejidad del diagrama.

Ejemplo 2.1. *Sea la secuencia a codificar* $\boldsymbol{u} = (1\,0\,1\,1\,1)$*, como vimos en el ejemplo 1.1, para el código convolucional de las figuras 1.1 y 1.2 la salida será* $\boldsymbol{v} = (1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$*. Este resultado coincide con el camino definido en el diagrama de trellis de la figura 2.2.*

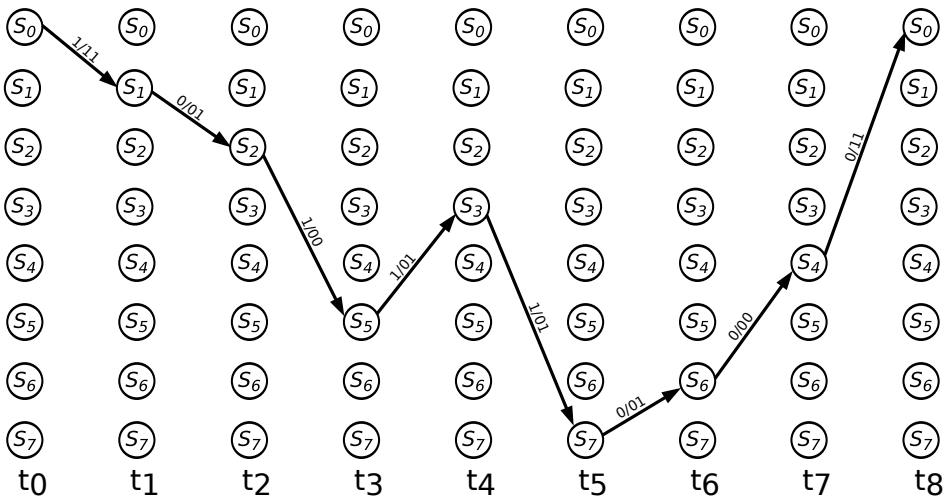


Figura 2.2: Diagrama de trellis con un camino que define la salida $\mathbf{v} = (1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$ para una entrada $\mathbf{u} = (1\,0\,1\,1\,1)$ para el código convolucional binario $(1, 2, 4)$ de las figuras 1.1 y 1.2.

2.3. El algoritmo

Sea la secuencia $\mathbf{u} = (u_0, u_1, \ldots, u_{L-1})$, de largo L, codificada en una palabra de código $\mathbf{v} = (v_0, v_1, \ldots, v_{2(L+k-1)-1})$, de largo $2(L + k - 1)$ con $k - 1$ el número de registros de desplazamiento; y sea la secuencia $\mathbf{r} = (r_0, r_1, \ldots, r_{2(L+k-1)-1})$ la versión de $\mathbf{v}$ recibida a través de un canal discreto y sin memoria (DMC). Un decodificador deberá producir una estimación $\hat{\mathbf{v}}$ de la palabra de código $\mathbf{v}$ basada en la secuencia recibida $\mathbf{r}$. Un *decodificador de máxima verosimilitud* (MLD) para un DMC elegirá $\hat{\mathbf{v}}$ como la palabra de código que maximice la función log-similitud, $\log P(\mathbf{r}|\mathbf{v})$.

Sea $P(r_i|v_i)$ la probabilidad de transición[1] del canal DMC, se tiene que

$$P(\mathbf{r}|\mathbf{v}) = \prod_{i=0}^{2(L+k-1)-1} P(r_i|v_i), \tag{2.1}$$

de donde se concluye que

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_{i=0}^{2(L+k-1)-1} \log P(r_i|v_i) \tag{2.2}$$

A la función log-similitud de la ecuación (2.2) se la llama *métrica* asociada al camino[2] $\mathbf{v}$ y se denota como $M(\mathbf{r}|\mathbf{v})$. Los términos $\log P(r_i|v_i)$ son llamados *métrica de bit* y se denotan como $M(r_i|v_i)$. Ambas métricas se relacionan según

$$M(\mathbf{r}|\mathbf{v}) = \sum_{i=0}^{2(L+k-1)-1} M(r_i|v_i) \tag{2.3}$$

También se define una métrica parcial para las primeras j ramas del camino $\mathbf{v}$ como

$$M([\mathbf{r}|\mathbf{v}]_j) = \sum_{i=0}^{j-1} M(r_i|v_i) \tag{2.4}$$

En la lista 2.1 se presenta al Algoritmo Viterbi, que cuando se aplica a una secuencia $\mathbf{r}$ recibida a través de un canal DMC obtiene el camino en el diagrama de trellis de mayor métrica. Este algoritmo es iterativo

[1] Se denomina *probabilidad de transición* a la probabilidad de recibir r_i dado que se transmitió v_i.

[2] Se utiliza la palabra "camino" en referencia a lo visto en la sección 2.2.

y a cada paso compara las métricas de todos los caminos entrando a cada uno de los estados y guarda el de mayor métrica, llamado *el sobreviviente*, conjuntamente con su métrica.

Algorithm 2.1 Algoritmo Viterbi

1: Para el tiempo $j = k - 1$, calcular la métrica parcial para cada camino entrando a cada estado. Almacenar el camino sobreviviente y su métrica para cada estado.
2: $j = j+1$. Calcular la métrica parcial para cada camino entrando a cada estado sumándole a cada métrica almacenada en el paso anterior la métrica de bit de la nueva rama. Almacenar el camino sobreviviente y su métrica para cada estado.
3: Si $j < L + k - 1$, repetir el paso 2. De lo contrario, detenerse.

Véase que para palabras de longitud L finita, en el instante $L + k - 1$ hay un único estado posible que es el S_0 (todos los registros de desplazamiento del codificador convolucional en cero) y por lo tanto habrá un único camino sobreviviente. En cambio, para palabras de largo infinito, como por ejemplo en los sitemas de *broadcasting* en los que el transmisor está continuamente codificando los bits a transmitir, sólo a medida que evoluciona el diagrama de trellis se podrán ir eliminando los caminos de menor métrica. El camino sobreviviente se irá decodificando con cierto retardo cada vez que entre dos instantes de tiempo, $t_j \to t_{j+1}$, haya una única transición posible. A este retardo se le denomina "retardo de decodificación", del inglés *decoding delay*, y puede tomar valores de hasta $5 \times k$ [7].

El camino sobreviviente será el de máxima verosimilitud y por lo tanto el Algoritmo Viterbi logra una estimación de máxima verosimilitud $\hat{\mathbf{v}}$ de la secuencia codificada $\mathbf{v}$. Esto se demuestra en el teorema 2.1.

Teorema 2.1. *El camino sobreviviente $\hat{\boldsymbol{v}}$ del Algoritmo Viterbi es el de máxima verosimilitud. Esto es*

$$M(\boldsymbol{r}|\hat{\boldsymbol{v}}) \geq M(\boldsymbol{r}|\boldsymbol{v}) \quad \forall \boldsymbol{v} \neq \hat{\boldsymbol{v}} \tag{2.5}$$

Demostración. Asumamos que el camino de máxima verosimilitud es eliminado por el algoritmo en el instante j. Esto implica que el camino parcial del sobreviviente hasta este punto tiene una métrica mayor que el de máxima verosimilitud. Luego, si el resto del camino de máxima verosimilitud (desde el punto j hasta el final) es anexado al camino sobreviviente hasta el punto j, entonces obtendríamos un camino de métrica mayor al de máxima verosimilitud. Y eso contradice la definición de camino de máxima verosimilitud como el que tiene mayor métrica. Finalmente, este camino nunca prodrá ser eliminado con el Algoritmo Viterbi. □

2.4. Algoritmo Viterbi para un Canal Binario Simétrico (BSC)

Es el ejemplo básico de un canal ruidoso de comunicaciones. Este canal tiene una entrada binaria, y su salida será igual a su entrada con una probabilidad $1-p$. Sin embargo, con probabilidad p un 0 será recibido como un 1 y un 1 será recibido como un 0 [8]. En este caso la capacidad del canal puede ser expresada como

$$C = 1 + p \log p + (1 - p) \log(1 - p) \quad \text{bits por uso del canal.} \tag{2.6}$$

La ecuación (2.2) aplicada a este canal puede ser expresada como

$$\begin{aligned} \log P(\mathbf{r}|\mathbf{v}) &= d(\mathbf{r}, \mathbf{v}) \log(p) + [2(L + k - 1) - d(\mathbf{r}, \mathbf{v})] \log(1 - p) \\ &= d(\mathbf{r}, \mathbf{v}) \log(\tfrac{p}{1-p}) + 2(L + k - 1) \log(1 - p) \end{aligned} \tag{2.7}$$

donde $d(\mathbf{r}, \mathbf{v})$ es la distancia de Hamming entre $\mathbf{r}$ y $\mathbf{v}$, esto es, el número de bits en los que $\mathbf{r}$ y $\mathbf{v}$ difieren; y $2(L + k - 1)$ el número de bits de las palabras codificada y transmitida.

Cuando la probabilidad de transición $p < \frac{1}{2}$,[3] se tiene que $\log[p/(1 - p)] < 0$ y que $2(L + k - 1) \log(1 - p)$ será constante para todo $\mathbf{v}$. De esta manera, un decodificador de máxima verosimilitud para un BSC eligirá $\hat{\mathbf{v}}$ de manera que minimice la distancia de Hamming

$$d(\mathbf{r}, \mathbf{v}) = \sum_{i=0}^{2(L+k-1)-1} d(r_i, v_i), \tag{2.8}$$

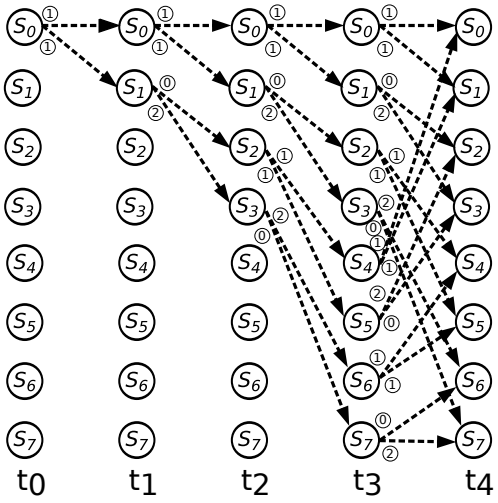
[3] Un BSC con $p > \frac{1}{2}$ siempre puede ser convertido a un BSC con $p < \frac{1}{2}$ simplemente denominando a las salidas de forma contraria.

o en otras palabras, el camino que sobreviva en el diagrama de trellis será el más cercano a **v** en términos de esta distancia. El resto del algoritmo se mantiene exactamente igual, excepto que la métrica será ahora calculada con la distancia de Hamming y no más con la función log-similitud.

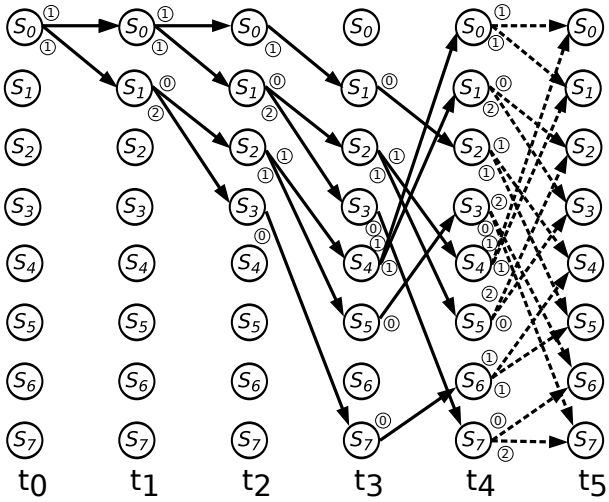
Ejemplo 2.2. *Sea la secuencia a codificar* $\boldsymbol{u} = (1\,0\,1\,1\,1)$*, como vimos en el ejemplo 1.1, para el código convolucional de las figuras 1.1 y 1.2 la salida será* $\boldsymbol{v} = (1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$*. Supóngase que luego de transmitir* $\boldsymbol{v}$ *a través de un BSC, obtenemos en recepción la secuencia* $\boldsymbol{r} = (1\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$*, con* b_1 *y* b_5 *cambiados.*

En las figuras 2.3 y 2.4 se muestra el proceso de la decodificación Viterbi. Se puede ver cómo al inicio de cada rama de cada camino se especifica la distancia de Hamming que esa rama agrega al resultado total. El camino sobreviviente será el que minimice la distancia de Hamming una vez que hayamos vuelto al estado S_0 *con todos los registros nuevamente en cero. Nótese que en la figura 2.4(b) se usó que en el último estado, al cabo de* $L + k - 1$ *instantes de tiempo, el codificador necesariamente vuelve al estado* S_0*.*

El resultado final es efectivamente la secuencia original, $\boldsymbol{v} = (1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1)$*. Y la distancia de Hamming entre esta secuencia y la secuencia recibida* $\boldsymbol{r}$ *es 2. Se puede ver cómo el camino de la figura 2.4(c) coincide con el camino de la figura 2.2.*

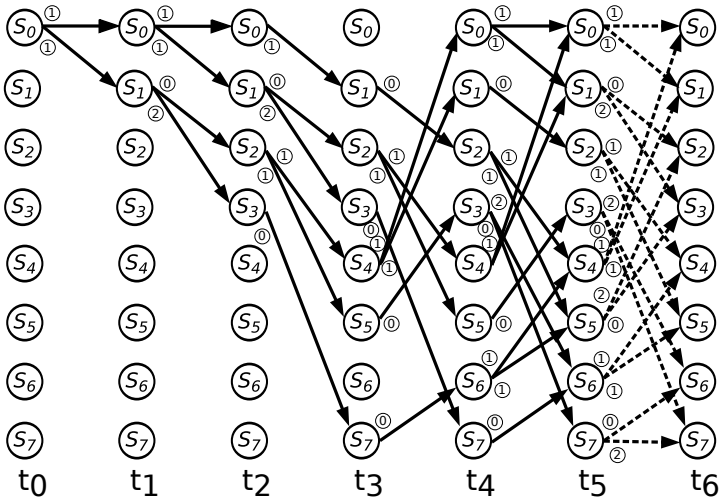


(a) Primero se calculan todos los caminos posibles con sus respectivas distancias.

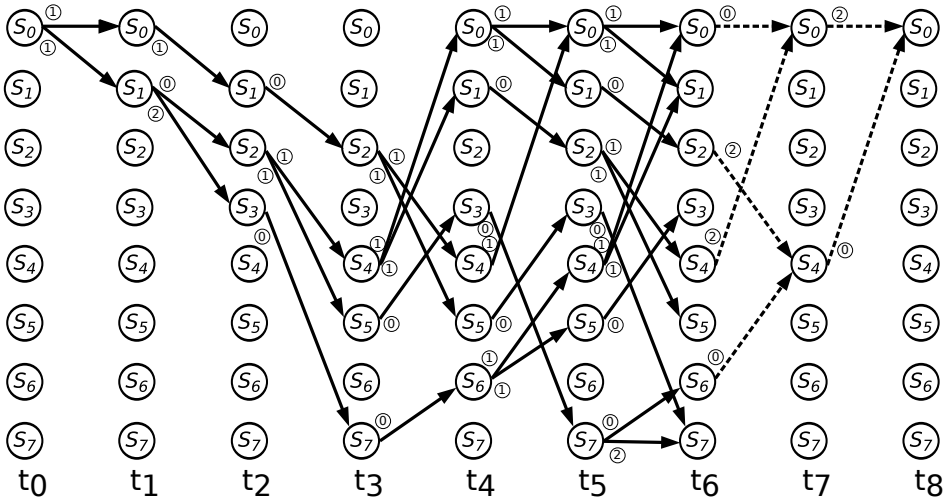


(b) Cuando más de un camino lleva al sistema al mismo estado, sobrevive el de menor distancia.

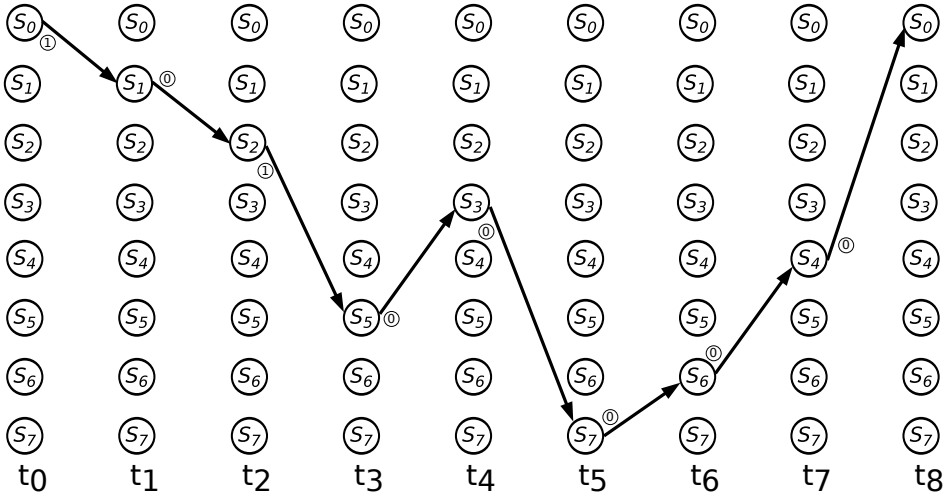
Figura 2.3: Decodificación Viterbi del Ejemplo 2.2. Pasos 1 y 2.



(a) Se continúa expandiendo el diagrama de trellis y escribiendo las respectivas distancias de cada rama.



(b) Se agrega la información de que al cabo de $L + k - 1$ instantes, el codificador debe volver al estado S_0.



(c) Camino sobreviviente.

Figura 2.4: Decodificación Viterbi del Ejemplo 2.2. Pasos 3, 4 y 5.

Referencias

[1] P. Elias, *Coding for Noisy Channels*, IRE Conv. Rec., Part 4, pp. 37-47,1955.

[2] Lin, Shu and Jr., Daniel J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983.

[3] Associação Brasileira de Normas Técnicas, *Televisão digital terrestre - Sistema de transmissão*, Norma Brasileira, 2008.

[4] A.J. Viterbi, *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, IEEE Trans. Inf. Theory., IT-13, pp. 260-269, April 1967.

[5] G. D. Forney, Jr., *The Viterbi Algorithm*, Proc. IEEE, 61, pp. 268-278, marzo de 1973.

[6] G. D. Forney, Jr., *Convolutional Codes II: Maximum Likelihood Decoding*, Inf. Control, 25, pp. 222-266, julio de 1974.

[7] Benigno Rodríguez Díaz, *Differential STBC for OFDM Based Wireless Systems*, Ph.D. Thesis, Technischen Universität Hamburg-Harburg, 2007.

[8] Cover, Thomas M. and Thomas, Joy A., *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience, 2006.