Decodificación Soft-Input Soft-Output: algoritmo BCJR

Procesamiento de señal en sistemas de comunicaciones y audiovisuales

Máster Universitario en Ingeniería de Telecomunicación

ETSIT-UPV

Práctica 3

Índice

1. Objetivos del trabajo

Con este trabajo se pretende que los alumnos asimilen la teoría estudiada en clase sobre el decodificador BCJR. Para ello se pretende que el alumno programe un decodificador basado en este algoritmo y verifique el correcto el funcionamiento del mismo. Se proporciona una estructura de una función la cual el alumno debe completar.

2. Ecuaciones

Se considera un codificador convolucional definido por una estructura de enrejado y una secuencia de N $n-$bits codificados a su salida ($\mathbf{x} = x_1, x_2, \cdots, x_N$), donde x_k es el símbolo de n-bits generado por el codificador en la etapa k. Suponemos además, que la secuencia codificada $\mathbf{x}$ se transmite a través de un ruido blanco aditivo gaussiano (AWGN), recibiendo la secuencia $\mathbf{y} = (y_1, y_2, \cdots, y_N)$.

Durante la práctica usaremos un código convolucional de tasa 1/2, n = 2, , con M = 4 estados S = {0, 1, 2, 3} y un enrejado como el que muestra la Figura 1. En esta figura, una línea punteada significa que la rama fue generada por un bit de mensaje 1 y una línea continua significa que fue generada por un bit 0. Además, cada rama está etiquetada con la palabra código asociada de dos bits x_k. Suponemos que estamos en la etapa k, el estado correspondiente en esta etapa es denotado como S_k, el estado previo es denotado como S_{k-1} y el simbolo recibido de n-bits correspondiente a esta etapa es denotado como $\mathbf{y}_k$.

El decodificador a programar (Algoritmo BCJR Max-log) emplea las ecuaciones estudiadas en teoría, las cuales están resumidas a continuación.

$$\alpha_k(S_k) = \max_{(S_{k-1},S_k)\in S} \{\alpha_{k-1}(S_{k-1}) + \gamma_i(S_{k-1}, S_k)\} \tag{1}$$

$$\beta_{k-1}(S_{k-1}) = \max_{(S_{k-1},S_k)\in S} \{\beta_k(S_k) + \beta_k(S_{k-1}, S_k)\} \tag{2}$$

Para el caso de las gamma (γ_k) al considerar un canal AWGN, utilizaremos la siguiente ecuación:

$$\gamma_k(S_{k-1}, S_k) = x_{k0}\frac{La(x_k)}{2} + \frac{Lc}{2}\sum_{l=0}^{n-1} y_{kl}x_{kl} \tag{3}$$

donde:

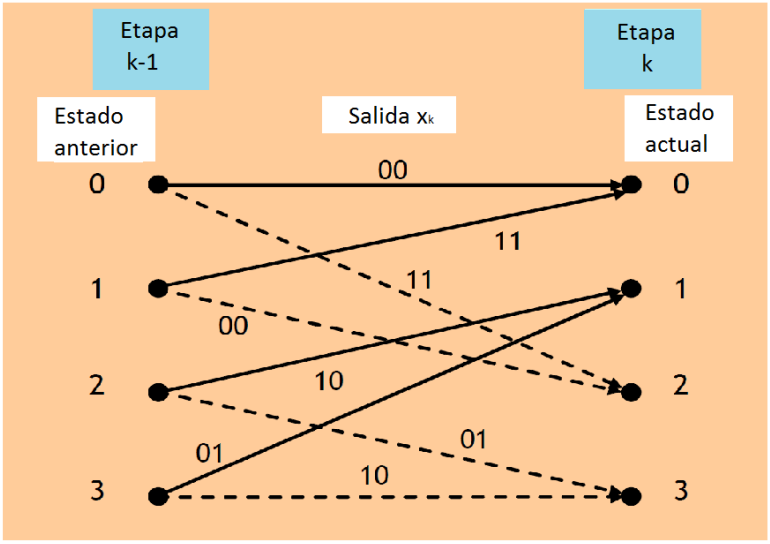

Figura 1: Estructura de trellis del algoritmos convolucional usado en la práctica.

x_{kl}: elemento l en el simbolo de salida x_k de n bits.
$La(x_k)$: información a priori del bit correspondiente a la etapa k.
Lc: información del ruido.
y_{kl}: elemento l del simbolo recibido $\mathbf{y}_k$.

Finalmente, para calcular los LLRs empleamos:

$$LLR(x_k|\mathbf{y}) = \max_{(S_{k-1},S_k)\in S^1}\{\alpha_{k-1}(S_{k-1}) + \gamma_i(S_{k-1},S_k) + \beta_k(S_k)\} - \quad (4)$$

$$\max_{(S_{k-1},S_k)\in S^0}\{\alpha_{k-1}(S_{k-1}) + \gamma_k(S_{k-1},S_k) + \beta_k(S_k)\} \quad (5)$$

3. Programación del BCJR.

El enrejado a considerar será el mostrado en la figura 1. Para ello definimos el enrejado tal y como muestra la ecuación (6).

$$
\begin{array}{rcccc}
EN = [& 1 & 1 & -1 & -1; \\
 & 1 & 3 & +1 & +1; \\
 & 2 & 1 & +1 & +1; \\
 & 2 & 3 & -1 & -1; \\
 & 3 & 2 & +1 & -1; \\
 & 3 & 4 & -1 & +1; \\
 & 4 & 4 & +1 & -1; \\
 & 4 & 2 & -1 & +1];
\end{array} \tag{6}
$$

En el enrejado definido en la ecuación (6), el número de filas es el doble del número de estados, debido a que por cada estado tenemos la información asociada a la transición provocada por un 0 (filas impares) y la información asociada a la transición provocada por un 1 (filas pares). Las columnas representan la siguiente información: estado anterior (S_{k-1}), estado actual (S_k), x_{k0}, x_{k1}. (en nuestro caso n=2, por tanto x_k tiene dos componentes).

Para implementar el decodificador, se proporciona una función que se debe completar. La estructura básica de la función es la siguiente:

```
function [LLR,Le]=bcjr(y,Lc,La,Enrejado)
%y = vector columna con la información del vector recibido.
%La = vector columna con la información a priori.
```

PASO 1: CALCULO DE γ

```
%Calcular Gamma: matriz de tamaño 2M × N.
    % Bucle for para k (Etapas).
    % Bucle for para m (Ramas).
```

PASO 2: CALCULO DE α

```
%Calcular Alfa: matriz de tamaño M × N.
    % Bucle for para k (Etapas).
    % Bucle for para m (Estados).
```

PASO 3: CALCULO DE β

```
%Calcular Beta: matriz de tamaño M × N.
    % Bucle for para k (Etapas).
    % Bucle for para m (Estados).
```

PASO 4: CALCULO DE LLR

```
%Calcular LLR y Le.
```

Para implementar el decodificador BCJR Max-log, debemos completar la función anterior, mediante los siguientes pasos:

3.1. PASO 1: Cálculo de γ

El primer paso para implementar el BCJR, es calcular todos los valores $\gamma_k(S_{k-1}, S_k)$ del enrejado. Para ello se debe emplear la ecuación (3). De este modo se debe calcular $\gamma_k(S_{k-1}, S_k)$ para las 2M ramas de las N etapas, construyendo la matriz Gamma correspondiente. En la Figura 2 se representan las 8 métricas a calcular en una de las etapas k.

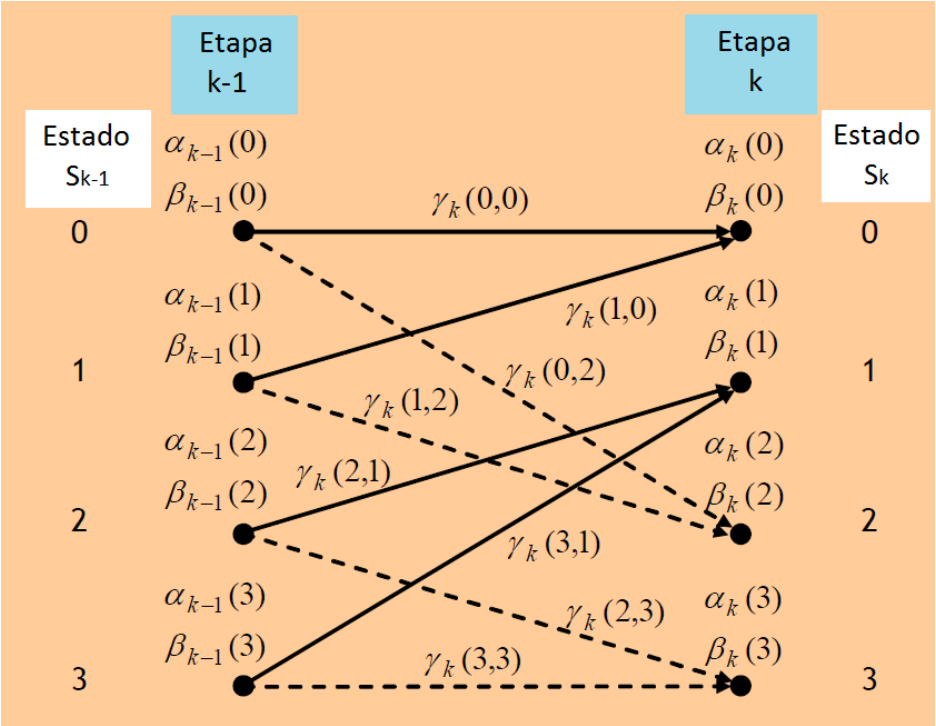

Figura 2: Etapa k del diagrama.

3.2. PASO 2: Cálculo de α

Para el caso de los valores de α se considera el máximo de todas las ramas que llegan al estado S_k desde el estado S_{k-1} empleando la ecuación (1). Por ejemplo, vemos en la Figura 3 que en la etapa k dos ramas llegan al estado $S_k = 2$, una proveniente del estado 0 y otra proveniente del estado 1, tal y como podemos ver en el Enrejado mostrado en la Figura 1. De tal manera que de las dos métricas proveniente de estas ramas, la métrica de mayor valor es la que se toma como valor para $\alpha_k(S_k)$. Para ello vamos a ir rellenando la matrix Alfa recorriendo los estados desde el inicio hasta el final del trellis, considerando para ello los valores iniciales $\alpha_0(S_0) = 0$ y $\alpha_N(S_e) = -\text{Inf}$ para e={1,2,3}.

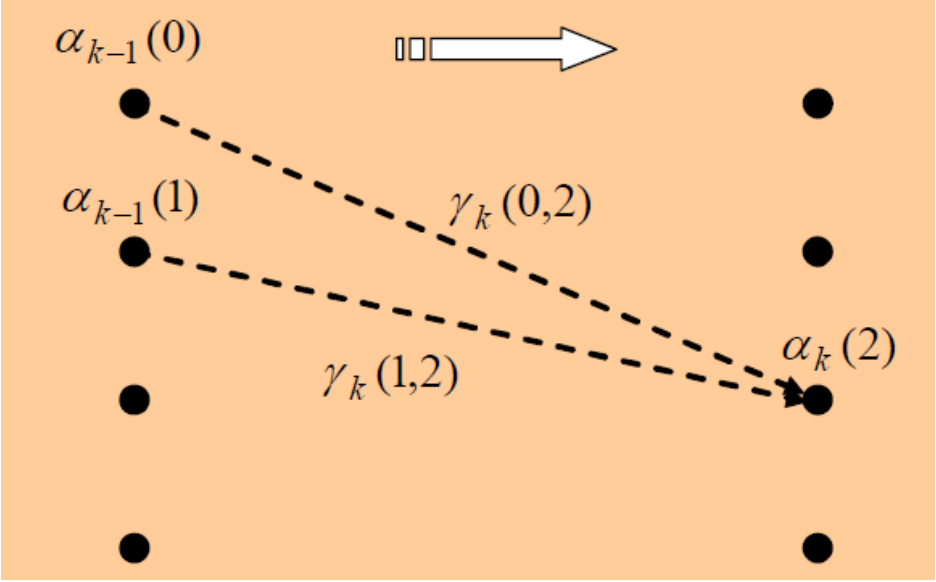

Figura 3: Computación de α.

3.3. PASO 3: Cálculo de β

En este paso los valores $\beta_{k-1}(S_{k-1})$ son calculados conociendo los valores $\beta_k(S_k)$, de forma similar a como se computan los valores $\alpha_k(S_k)$. Pero en este caso los valores se van calculando desde el final hasta el principio del trellis.

Por ejemplo en la Figura 4, vemos como dos ramas llegan al estado $S_{k-1} = 1$, una desde el estado $S_k = 0$ y otra desde el estado $S_k = 2$, tal y como muestra el Enrejado representado en la Figura 2. De tal manera que tenemos que inicializar $\beta_k(S_k)$ con el valor máximo de las métricas proporcionadas por estas ramas.

Para ello vamos a ir rellenando la matrix Beta recorriendo los estados desde el final hasta el principio del trellis, considerando para ello los valores iniciales $\beta_N(S_0) = 0$ y $\beta_N(S_e) = -\text{Inf}$ para e={1,2,3}.

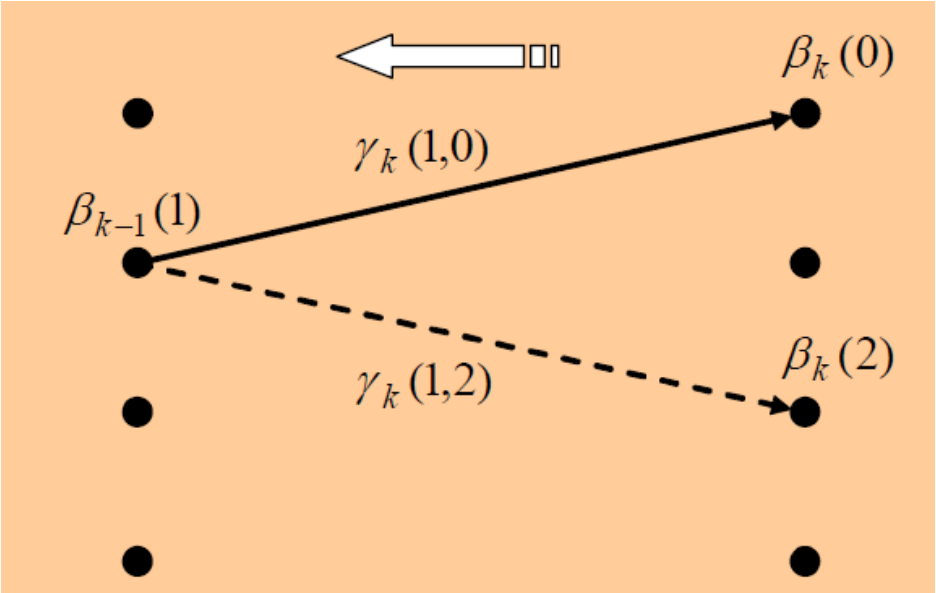

Figura 4: Computación de β.

3.4. PASO 4: Cálculo de LLR

Una vez los valores γ, α y β han sido calculados podemos calcular el valor de los LLRs empleando la ecuación (4).

Para comprobar el correcto funcionamiento de la función se debe comparar el resultado con el obtenido por la función `bcjr_comprobacion` que se proporciona (los parámetros de entrada son los mismos que la función a programar). Para ello se comprobará el resultado de los siguientes dos ejemplos:

- Caso 1:
  $Lc = 1$;

$y = [0{,}8; 0{,}1; 1{,}0; -0{,}5; -1{,}8; 1{,}1; 1{,}6; \quad -1{,}2; -0{,}5; 0{,}1];$
$La = zeros(5, 1);$

- Caso 2:
  $Lc = 1;$
  $y = [0{,}8; -1{,}2; -1{,}8; 1{,}2; 1; 0{,}2; 1{,}6; -1{,}1; -0{,}3; -0{,}6];$
  $La = [-0{,}9; 1{,}4; -0{,}9; -0{,}3; 0];$