Códigos catastróficos

Asignatura: Teoría de códigos y criptografía
Profesor: Lorenzo Javier Martín García

Álvaro Martínez Arbas
Ignacio Amaya de la Peña

2 de junio de 2015

Índice

1 Introducción

Los códigos catastróficos se encuentran englobados dentro de los códigos convolucionales. Se dice que un código es catastrófico si y sólo si su diagrama de estados contiene un bucle de peso cero en un estado diferente al inicial.

La transmisión de mensajes de forma fiable y libre de errores es un pilar básico de cualquier sistema de comunicaciones. Para conseguirlo se cuenta con dos estrategias:

- ARQ (*Automatic Repeat Request*): Se centra en la detección de errores, pero no los corrige. En caso de error sería necesario repetir la transmisión del mensaje.

- FEC (*Forward Error Correction*): Se basa en detectar los errores del mensaje recibido y corregir el mayor número de ellos posible.

En los dos casos anteriores es preciso añadir bits de información redundante al mensaje, pues en caso contrario sería imposible poder detectar o corregir los errores. Este proceso de añadir redundancia es denominado codificación de canal.

Las dos principales técnicas empleadas en la codificación de canal son los códigos de bloques y la codificación convolucional. Este último es una de las técnicas FEC más adecuadas en los canales donde la señal que se transmite se ve corrompida por ruido gaussiano blanco y aditivo (denominado AWGN). Muchos canales de radio pueden ser modelados como canales AWGN, pero hay otros muchos que presentan otros problemas y en los que esta técnica podría no ser la óptima.

Durante muchos años, la codificación convolucional ha sido la técnica FEC predominante en las comunicaciones espaciales, especialmente en redes VSAT.

2 Códigos convolucionales

Los códigos convolucionales presentan palabras de longitud variable y sus codificadores tienen memoria, por lo que la salida no sólo depende de la entrada, sino también de un número m de estados intermedios. Estos códigos son lineales e invariantes en el tiempo.

Los códigos convolucionales se representan por la tupla (n,k,m) donde n es el número de salidas, k el número de entradas y m el número de estados. Vamos a centrarnos en los que presentan sólo una entrada para reducir la complejidad.

Formalmente, un código convolucional (n,1,m) con generadores $\vec{g^1}, \vec{g^2}, ..., \vec{g^n} \in \mathbb{Z}_2^{m+1}$ es el código formado por todas las palabras de la forma:

$$\vec{v} = \left(\vec{u} * \vec{g^1}, \vec{u} * \vec{g^2}, ..., \vec{u} * \vec{g^n}\right)$$

donde $\vec{u} = (u_0, u_1, ...)$ representa una entrada cualquiera y $\vec{v} = (v_0, ..., v_n)$ la salida codificada.

Una de las mayores ventajas que presentan estos códigos es que se pueden implementar fácilmente usando solamente desplazamientos y sumadores binarios.

Si C es un código convolucional (n,1,m) con generadores $\vec{g^1}, \vec{g^2}, ..., \vec{g^n} \in \mathbb{Z}_2^{m+1}$ tales que

$$\forall i = 1, 2, ..., n \vec{g^i} = (g_0^i, g_1^i, ..., g_m^i)$$

entonces su matriz generadora es $G \in \mathcal{M}_{\ell x(m+\ell)n}$ donde ℓ es la longitud de la entrada. Estas matrices se construyen fácilmente a partir de los generadores.

Los $(m+1)n$ primeros elementos de la primera fila de G son los componentes de los vectores generadores en orden creciente, de forma que que primero estarían los n elementos de $\vec{g_0}$, luego los n elementos de $\vec{g_1}$ y así hasta llegar a los de $\vec{g_m}$. El resto de la fila se rellenaría con ceros. La siguiente fila se genera a partir de la anterior desplazándola n posiciones a la derecha y rellenando los huecos con ceros. Si repetimos este proceso al final tendremos los n elementos de $\vec{g_0}$ al final de la última fila.

Cabe destacar que todo lo expuesto aquí acerca de los códigos convolucionales (n,1,m) se puede extender fácilmente a los (n,k,m), pero se ha tomado la decisión de no incluirlo en el presente trabajo debido a que la notación es mucho más confusa, siendo los conceptos muy parecidos.

Existen varios conceptos de distancias en los códigos convolucionales, siendo la más importante la denominada distancia libre, que se define como el mínimo de las distancias de Hamming entre dos palabras diferentes del código. Como las palabras de un código convolucional pueden tener distinto tamaño y para calcular la distancia necesitamos que estos sean iguales, se completan las que sean más pequeñas con ceros.

Si d es la distancia libre de un código convolucional, tenemos que su capacidad correctora t se calcula mediante la siguiente expresión:

$$t = E\left[\frac{d-1}{2}\right]$$

Un codificador de un código convolucional puede ser descrito mediante un diagrama de estados, de forma que dada una entrada, su salida queda determinada de forma unequívoca. Si desarrollamos en forma de grafo todas las posibles situaciones, obtenemos el diagrama de Trellis, que nos permite decodificar las palabras recibidas sin errores y corregir errores dentro de los límites de la capacidad correctora del código.

3 Códigos catastróficos

Un generador n-tuplo $g(D)$ es considerado catastrófico si existe una secuencia infinita de entrada $u(D)$ que genera una n-tupla finita como salida $y(D) = u(D)g(D)$. Cualquier ejecución de $g(D)$ deberá tener, por tanto, un ciclo en su diagrama de estados que no sea en el estado inicial.

Hay una correspondencia uno a uno entre el conjunto de todos los caminos posibles del diagrama de Trellis de un codificador convolucional y el conjunto

$u(D)$ de todas las posibles secuencias entrada. Sin embargo, la corresponencia entre el conjunto de todas las secuencias salida $u(D)g(D)$ podría no serlo. Por ejemplo, si el codificador es catastrófico, entonces existen al menos dos caminos que corresponden a la secuencia entrada formada sólo por ceros. Debido a la !!propiedad de grupo!! de un codificador habrá al menos dos caminos para cada una de las posiles secuencias salida. De hecho, La correspondencia entre los caminos del diagrama de Trellis y las secuencias de un código convolucional sólo son uno a uno en el caso en que $g(D)$ es no catastrófico.

TEOREMA.Podemos decir que un generador n-tuplo es catastrófico si y sólo si todos los numeradores $n_j(D)$ tienen un factor en común distinto de D.

Pongamos un ejemplo del teorema anterior. El codificador k=1 (ratio 1/2) $g(D) = (1 + D^2, 1 + D + D^2)$ es no catastrófico porque los polinomios $1 + D^2$ y $1 + D + D^2$ no tienen nigún factor en común. Sin embargo, $g(D) = (1+D^2, 1+D)$ es catastrófico, ya que $1+D$ y $1+D^2$ tienen en común el divisor $1 + D$. Por tanto la secuencia infinita de entrada $1/(1 + D) = 1 + D + D^2 + ...$ nos lleva a una salida finita $y(D) = (1 + D, 1)$ debido a un bucle en el estado 11 con salida 00.

Si $g(D)$ es catastrófico y tiene como máximo común múltiplo $mcdn_j(D)$, entonces podemos encontrar su generador no catastrófico equivalente dividiendo por ese valor. $g'(D) = g(D)/\text{mcd}(n_j(D))$. Volviendo al ejemplo anterior, podemos remplazar $(1 + D^2, 1 + D)$ por $(1 + D, 1)$.

Por tanto, se deben eliminar los denominadores y los factores comunes.

COROLARIO. Todo generador n-tuplo $g(D)$ es equivalente a un único (junto con sus múltiplos escalares) generador n-tuplo $g'(D)$:

$$g'(D) = \frac{mcm\,\{d_j(D)\}}{mcd\,\{n_j(D)\}} g(D)$$

donde n_j y d_j son los numeradores y denominadores.

Además, se sabe que un codificador convolucional es mínimo (esto significa que tiene el número mínimo de estados entre todos los codificadores para un mismo código) si y sólo si es no catastrófico, por lo que no puede haber bucles con salida cero en estados que no sean el inicial. Además se deberá cumplir que el generador $g(D) = (n_1(D), ...n_n(D))/d(D)$ no tenga n_j con factores en común, ni que $d(D)$ sea de grado mayor que el grado máximo de los numeradores.

Un codificador sistemático de ratio (1/n) es aquel en que la secuencia de entrada no cambia para alguna de las n secuencias salida. Un codificador sistemático se puede obtener a partir de $g(D) = (g_1(D), ..., g_n(D))$ multiplicando todos los generadores por $1/g_1(D)$. Por ejemplo, dado un generador

$g(D) = (1 + D^2, 1 + D + D^2)$ podemos obtener fácilmente su codificador sistemático.

$$g(D) = (1, \frac{1 + D + D^2}{1 + D^2})$$

4 Propiedades de los códigos no catastróficos

Como hemos explicado previamente, un codificador catastrófico es aquel donde al menos una palabra código con peso de Hamming finito(bits distintos de cero) es la salida de una entrada con peso de Hamming infinito (infinito número de unos). Esto quiere decir que para una entrada infinita tendremos una palabra código finita, por lo que habrá un número infinito de errores (ya que hay un número infinito de unos que se han codificado como ceros).

Como el conjunto de todas las palabras código es también el conjunto de todos los errores posibles, entonces lo anterior es equivalente a decir que un número finito de errores en la decodificación de una secuencia puede dar lugar a un número infinito de errores en la decodificación de la entrada.

Debido a que este tipo de códigos son poco deseables, se han desarrollado distintos métodos para detectarlos.

Para todo código siempre existe un codificador no catastrófico. De hecho, un codificador sistemático no puede ser catastrófico. Por esta razón la mayoría de los códigos convolucionales se suelen implementar usando los codificadores sistemáticos de menor complejidad posible.

Sea $G(D)$ un generador matricial de un código convolucional, entonces las siguientes condiciones son equivalentes. Si $G(D)$ satisface cualquiera de ellas, entonces el codificador será no catastrófico.

1. Un número finito de bits en la entrada produce un número finito de bits en la salida (equivalente a que todo número infinito de bits de entrada, produce un número infinito de bits en la salida).

2. El máximo común divisor de los determinantes de todas las submatrices $k_{\times}k$ de $G(D)$ es una potencia no negativa de D (es decir, equivale a D^{δ} para $\delta > 0$).

3. $G(D)$ tiene una inversa por la derecha $K(D)$ de peso finito, de forma que $G(D)K(D) = I_k$.

5 Programa realizado

Se ha decidido implementar una herramienta en Java que nos permitiese codificar y decodificar una ristra de bits. De esta manera se puede comprobar

con ejemplos prácticos la teoría expuesta en los apartados anteriores.

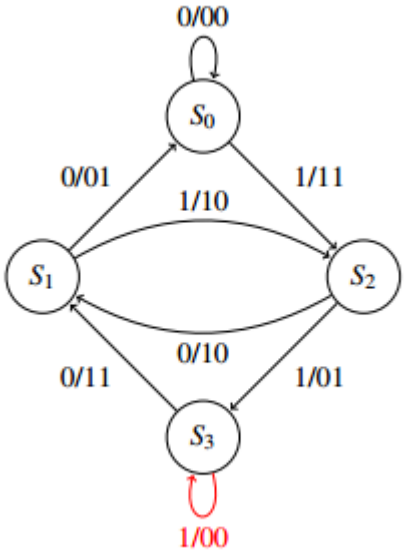
Se ha escogido Java como lenguaje de programación porque cuenta con la librería Swing para realizar interfaces gráficas, lo que resulta de utilidad a la hora de presentar nuestros resultados. De esta forma, basta con introducir en la interfaz los bits que queremos codificar o decodificar. Se podrá introducir el diagrama de estados de un codificador convolucional siguiendo un formato determinado.Se pueden especificar diagramas de estado para cualquier número de salidas y de estados (se le pasan los valores n y m), pero sólo para k=1 (no acepta más de una entrada). Para las pruebas se han utilizado dos codificadores: uno catastrófico y el otro no catastrófico. De esta forma, se puede comparar el comportamiento de ambos. A la hora de decodificar se corregirán los errores siempre y cuando su número no exceda la capacidad correctora del código. Cabe destacar que aunque haya dos correcciones posibles (lo que implica dos caminos distintos en el diagrama de Trellis al aplicar Viterbi) sólo se ha tenido en cuenta una de ellas. Además, para asegurarnos de que alcanza el estado final se le ha añadido a las entradas once ceros al final de forma automática, de forma que si introducimos la entrada 111, el programa la procesa como 11100000000000.

El diagrama de estados debe introducirse en la aplicación de forma que cada línea indique un estado con sus correspondientes salidas con el cero y con el uno, ambas separadas por un espacio. El último estado deberá terminar en salto de línea. Uno de los diagramas de estados usados para las pruebas ha sido el siguiente:

0000 2111
0001 2110
1010 3101
1011 3100

Por ejemplo, fijémonos en el primer estado, que viene definido por: 0000 y 2111. El primer número nos indica el estado al que se avanza y el segundo el bit con el cual se avanza a ese estado (en el primer caso hay un bucle en el estado cero con el bit cero y se avanza al estado dos con el bit uno). Las salidas generadas se indican en los dos números siguientes y el número de estas dependerá del valor de n.

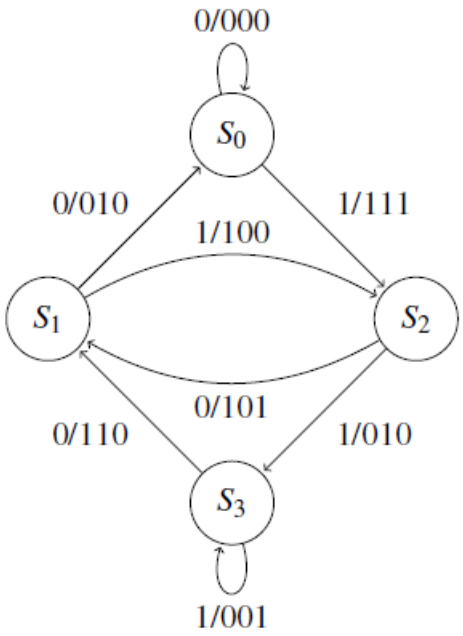
El ejemplo anterior se corresponde al codificador catastrófico con el siguiente diagrama de estados:



El otro diagrama de estados venía definido por:

00000 21111
00010 21100
10101 31010
10110 31001

Este es un codificador convolucional no catastrófico, cuyo diagrama se corresponde con:



La aplicación controla que las entradas sean del tipo correcto y que no se dejen en blanco para evitar errores. Aún así, en el diagrama de estados sólo se comprueba que las dimensiones sean las correctas (por lo que si se introducen letras o un autómata erróneo el programa dará un error en la ejecución). Asimismo sólo se aceptan entradas en binario.

Se incluyen dos ficheros de texto con los diagramas de estado usados para facilitar la labor de introducirlos en la aplicación.

Por defecto la aplicación se inicia con el diagrama catastrófico mostrado anteriormente.

Las pruebas que se han realizado han sido las siguientes (se pueden repetir en la aplicación para confirmarlas o también ejecutar el main de Java que se incluye junto con el código):

- Autómata catastrófico.

  Entrada original usada: 111111111111
  Entrada codificada: 110100000000000000000000

  Sin embargo al decodificar la entrada anterior codificada, tenemos lo siguiente:

Entrada: 00010000000000000000000
Corregida: 00000000000000000000000 , Errores = 1

Secuencia :: S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0→ S0

Matriz de distancias y estados Las filas son los instantes y las columnas los estados Los valores son el camino y la distancia

(S0,0)			
(S0,0)		(S0,2)	
(S0,1)	(S2,4)	(S0,1)	(S2,2)
(S0,1)	(S2,2)	(S0,3)	(S2,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)	(S0,3)	(S3,2)
(S0,1)	(S2,4)		
(S0,1)			

Se produce el peor error posible entre el mensaje original 111111... y el mensaje descodificado 00000...

- Autómata no catastrófico.

Primero pongamos dos ejemplos de codificación:

Entrada 1: 1101000111010011100100
Salida 1: 11101011010010101000011101000111010010101011101000111001011110101 0
Entrada 2: 1011000
Salida 2: 111101100010110010000

Ahora veamos qué obtenemos al decodificar la salida 2 anterior con dos errores intencionados para el ejemplo:

Entrada → 111111100110110010000
Corregida -→ 111101100010110010000, Errores = 2

Salida -→ 1011000
Secuencia :: S0→ S2→ S1→ S2→ S3→ S1→ S0→ S0

Matriz de distancias y estados
Las filas son los instantes y las columnas los estados
Los valores son el camino y la distancia

(S0,0)			
(S0,3)		(S0,0)	
(S0,6)	(S2,1)	(S0,3)	(S2,2)
(S1,3)	(S3,3)	(S1,1)	(S3,4)
(S1,4)	(S2,3)	(S0,4)	(S2,2)
(S1,4)	(S3,2)	(S1,4)	(S2,5)
(S1,2)	(S3,6)		
(S0,2)			