Electrónica Digital

Ingeniería Informática – FICH, UNL
Leonardo Giovanini

Códigos

Codificación de canal

o

Corrección de errores

Organización de la clase

En esta clase se estudiarán los siguientes temas:

- Definición;
- Tipos;
- Códigos lineales;
- Detección de errores;
- Corrección de errores;
- Código de Hamming.

Códigos de canal – *Definición*

La detección y corrección de errores son técnicas que permiten mantener la integridad de los datos en medios de almacenamiento y canales de comunicación poco confiables.

La comunicación y el almacenamiento de datos produce continuamente un movimiento a través de canales no diseñados para este propósito e introducen errores en los datos recibidos.

Por lo tanto, debemos asegurarnos que si aparecen errores, éstos puedan al menos ser detectados y corregidos.

El método para detectar y corregir errores es incluir en los bloques de datos adicionales denominados redundancia.

Hay dos estrategias básicas para manejar los errores:

- **Corrección de errores** – incluir ***suficiente información redundante*** para que se puedan ***detectar y corregir*** los erróres;
- **Detección de errores** – incluir ***sólo la información redundante*** necesaria para ***detectar*** los errores.

Códigos de canal – *Definición*

Los esquemas de detección y corrección pueden ser:

- **No sistemático** – el mensaje original se transforma en un mensaje codificado que lleva la misma información y que tiene al menos la misma longitud del mensaje original.
- **Sistemático** – el transmisor envía los datos originales y adjunta información de verificación que se deriva de los datos mediante algún algoritmo determinista. Si solo se requiere la detección de errores, un receptor puede simplemente aplicar el mismo algoritmo a los datos recibidos y comparar su salida con los datos de verificación recibidos y determinar si se produjo un error.

Un buen rendimiento de control de errores requiere que el esquema se seleccione en función de las características de los errores a resolver. Los modelos más comunes son

- ***Modelos sin memoria*** – los errores ocurren al azar y con una cierta probabilidad; y
- ***Modelos dinámicos*** – los errores ocurren principalmente en ráfagas.

Algunos códigos están especializados para errores aleatorios, otros están especializados para errores en ráfaga y otros pueden ser adecuados para una combinación de errores aleatorios y errores de ráfaga.

Si las características de los errores no se pueden determinar, o son muy variables, se combinan un esquema de *"detección de errores"* con un *"sistema de retransmisión de datos"* y se lo conoce como ***solicitud de repetición automática*** (ARQ), que se utiliza principalmente en Internet. Un enfoque alternativo es la ***solicitud de repetición automática híbrida*** (HARQ), que es una combinación de ARQ y codificación de corrección de errores.

Códigos de canal – *Tipos*

Existen tres tipos principales de corrección de errores

- **Solicitud de repetición automática** (ARQ) – utiliza códigos de detección de errores, mensajes de confirmación (positivos y/o negativos) y tiempos de espera para lograr una transmisión de datos confiable. El receptor envia un ***acuse de recibo*** para indicar que ha ***recibido correctamente*** los datos. Cuando el transmisor no recibe el acuse de recibo antes de que se agote el tiempo de espera, retransmite la trama hasta que se recibe correctamente o el error persiste más allá de un número predeterminado de retransmisiones.

- **Adelanto de información** (FEC) – es un proceso de agregar datos redundantes para recuperar la información aún cuando hay varios errores (hasta la capacidad del código utilizado), ya sea durante la transmisión o el almacenamiento. Los códigos de adelanto de información se utilizan con frecuencia en la capa física de los sistemas de comunicación y el almacenamiento confiable en CD, DVD, discos duros y RAM. Este tipo de códigos de corrección se clasifican en códigos convolucionales y códigos de bloque.

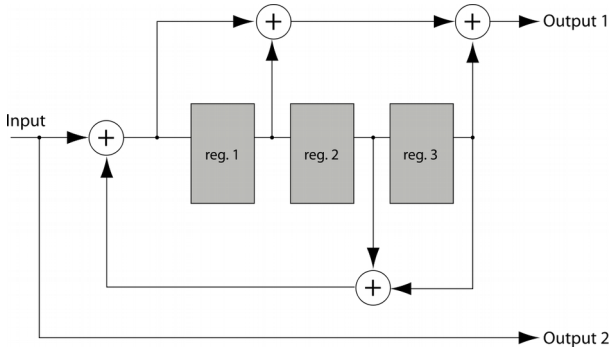
- **Esquemas híbridos** (HARQ) – es una combinación de ARQ y FEC. Hay dos enfoques básicos:
  - Los mensajes siempre se transmiten con ***información de detección/correccion*** – El receptor decodifica un mensaje utilizando la información redundante y solicita la retransmisión utilizando ARQ solo si la corrección no fué exitosa (verificación de integridad fallida).
  - Los mensajes se transmiten solo con ***información de detección*** – Si un receptor detecta un error, solicita información de corrección del transmisor utilizando ARQ y lo utiliza para reconstruir el mensaje original.

Códigos de canal – *Tipos*

Los códigos de corrección de errores generalmente se distinguen entre códigos convolucionales y códigos de bloque:

- **Códigos de bloque** – son códigos que procesan la información bloque a bloque de longitud fija. En particular los ***códigos de bloque lineales*** son códigos que tienen la ***propiedad de linealidad***: la suma de dos palabras de código también es una palabra de código.

- **Códigos de convolucionales** – son códigos en los que cada símbolo de palabra de código es la suma ponderada de los símbolos de mensaje de entrada de modo que la salida del codificador es la convolución de la entrada contra los estados almacenado en registros. Estos códigos tienen un desempeño similiar a un código de bloque equivalente pero su implementación es más sencilla.



Los códigos de bloque clásicos se decodifican con ***algoritmos de decisión dura***, esto significa que para cada símbolo se toma la decisión si corresponde o no a alguno de los símbolos de $\mathcal{T}$. En contraste, los códigos convolucionales se decodifican con ***algoritmos de decisión suave*** como, esto significa que para cada símbolo se calcula el grado de parecido con símbolos de $\mathcal{T}$ y luego se decide, como los algoritmos de Viterbi, que logran un ***mejor rendimiento*** que la decodificación de decisión dura.

La mayoría de los códigos de corrección solo ***corrigen cambios***, en esta configuración la ***distancia de Hamming*** es la forma adecuada de medir la tasa de error. Si uno quiere corregir inserciones o eliminaciones, la ***distancia de Levenshtein*** es más apropiada para medir la tasa de error cuando se usan dichos códigos.

Códigos de bloque lineales

Los códigos de bloque lineal se resumen por sus alfabetos y parámetros (n, m, d_{min}) donde

n es la longitud de la palabra de código, en símbolos;

m es el número de símbolos fuente que se usarán para codificar;

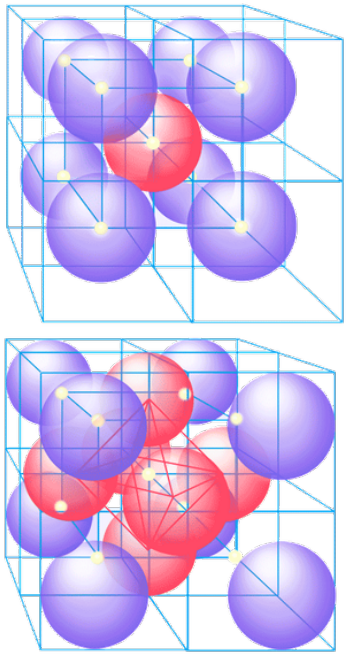
d_{min} es la distancia Hamming mínima del código.

Existen muchos tipos de códigos de bloque lineal, como por ejemplo

- **Códigos de repetición** – son códigos que repiten el mensaje varias veces con la probabilidad de que el canal corrompa una minoría de las repeticiones, de forma que el receptor detecte el error de transmisión;
- **Códigos cíclicos** – son códigos donde rotaciones de las palabras código dan otra palabra código. Su estructura está relacionada con el Cuerpo de los polinomios con coeficientes enteros (***Campos de Galois***) ya que los números binarios se representan como polinomios. Los algoritmos de codificación y decodificación son computacionalmente muy eficientes;
- **Códigos polinomiales** – son códigos cuyo conjunto de palabras de código válidas consiste en aquellos ***polinomios*** (generalmente de cierta longitud fija) que son divisibles por un polinomio fijo dado, de longitud más corta, llamado ***polinomio generador***;
- **Códigos BCH** – son una clase de códigos que se construyen utilizando polinomios sobre un campo finito. Sus características más importantes son: control preciso sobre el número de errores corregibles y la facilidad con la que se pueden decodificar, es decir, a través de un método algebraico conocido como ***decodificación de síndrome***;

Códigos de bloque lineales

Los códigos de bloque están vinculados con el ***problema de empaquetamiento de esferas en las esquinas de un cubo,*** con las esferas definidas por la distancia de Hamming d : si las esferas tienen radio d entonces sus centros son la solución de un código de corrección de errores.
En códigos binarios las ***dimensiones*** refieren a la ***longitud de la palabra de código.***



A medida que las dimensiones se hacen más grandes, el porcentaje de espacio ocupado por ls esferas aumenta. Pero en ciertas dimensiones, el empaquetado utiliza todo el espacio, es decir el código aprovecha todas las palabras código disponibles, y estos códigos son los llamados códigos "*perfectos*".

Formalmente, esta idea está conectada con la ***cota de Hamming*** dada por

$$A_q(n,d) \leq \frac{q^n}{\sum_{k=0}^{t} \binom{n}{k}(q-1)^k} \qquad \text{donde} \qquad t = \left\lfloor \frac{d-1}{2} \right\rfloor .$$

La cual establece una limitación sobre la eficiencia con la que cualquier código de corrección de errores utiliza el espacio en el que embebe sus palabras de código.
Si un código alcanza el límite de Hamming se dice que es un código perfecto.

Cuando las ***dimensiones aumentan***, el número de ***vecinos cercanos aumenta*** y con ello la probabilidad de error también crece. Puede ser que la ***probabilidad de error con un solo vecino*** baje, pero el número de vecinos puede ser lo suficientemente grande como para que la ***probabilidad de error total*** aumente.

La detección de errores se realiza utilizando una ***función hash*** adecuada que agrega una ***etiqueta de longitud fija*** al mensaje, lo que permite a los receptores verificar el mensaje entregado recalculando la etiqueta y comparándola con la proporcionada.
Existe una gran variedad de diferentes diseños de funciones hash. Sin embargo, algunos son de uso particularmente extendido debido a su simplicidad o su idoneidad para detectar ciertos tipos de errores.

Bit de paridad – es un bit que se agrega al mensaje para garantizar que el número de bits con valor 1) en el resultado sea par o impar. Es un esquema muy simple que se puede utilizar para detectar un número impar (es decir, uno, tres, cinco, etc.) de errores. Un número par de bits invertidos hará que el bit de paridad parezca correcto aunque los datos sean erróneos. Las extensiones y variaciones en el mecanismo de bits de paridad son verificaciones de redundancia longitudinal, verificaciones de redundancia transversal y técnicas similares de agrupación de bits.

Checksum – es una suma aritmética modular de palabras de código de mensaje de una longitud de palabra fija (por ejemplo, valores de bytes). La suma puede ser negada antes de la transmisión para detectar mensajes no intencionales de cero. Los esquemas de suma de verificación incluyen bits de paridad, dígitos de verificación y verificaciones de redundancia longitudinal.

Cheque de Redundancia y Cíclica (CRC) – es una función de hash diseñada para detectar cambios accidentales en datos digitales. No es adecuado para detectar errores introducidos maliciosamente. Se caracteriza por la especificación de un polinomio generador, que se utiliza como divisor en una división polinómica sobre un campo finito, tomando los datos de entrada como dividendo. El resto se convierte en el resultado. El bit de paridad se puede ver como un CRC de 1 bit para casos especiales.

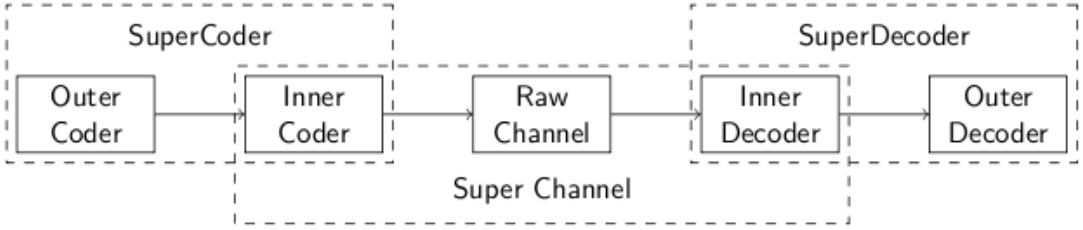
Un código con ***distancia mínima de Hamming*** d ***puede detectar hasta*** $d-1$ errores.

Los códigos con distancia mínima de Hamming $d=2$ son casos degenerados de códigos de corrección y se pueden utilizar para detectar un error.

Hay muchos tipos de códigos de bloque, los códigos Reed-Solomon se destacan por su uso en unidades de almacenamiento masivo como DVD y discos duro. Otros ejemplos de códigos de bloque incluyen los códigos Golay y BCH, utilizados en los satélites, y los códigos de Hamming que se usan para corregir errores en circuitos digitales como los discos de estado sólido y las memorias RAM

Los errores generalmente ocurren en ráfagas en lugar de ser independientes. Para aliviar este problema ***se mezclan*** símbolos de origen en varias palabras de código ***intercalandolos*** (***interleaving***). De esta manera se crea una distribución más uniforme de errores, mejorando el desempeño de los códigos de corrección que han sido diseñados para errores aleatorios.

Para ***mejorar el desempeño*** de los códigos correctores se los puede combinar para obtener probabilidades de error que disminuyan al aumentar la longitud del bloque, dando lugar a la clase de los ***códigos concatenados***.



El código interno suele ser un ***código convolucional con decisión blanda y corto*** mientras que el código externo suele ser un ***código de bloque de decisión dura largo***. El tamaño de símbolo grande hace que el código externo sea robusto para las ráfagas de error que pueden ocurrir debido a degradaciones del canal y también porque la salida errónea del código convolucional en sí es ráfaga.

Los códigos de Hamming son una ***familia de códigos lineales*** correctores capaces de detectar errores de hasta dos bits o corregir errores de un bit. Los códigos de Hamming son ***códigos perfectos***, es decir logran la tasa más alta posible para códigos con su longitud de bloque, y una ***distancia mínima de tres***.

Para cada $r \geq 2$ hay un código con longitud de bloque $n = 2r - 1$ y longitud de mensaje $k = 2^r - r - 1$ tal que su tasa es

$$R = kn^{-1} = (1 - r)(2r - 1)^{-1},$$

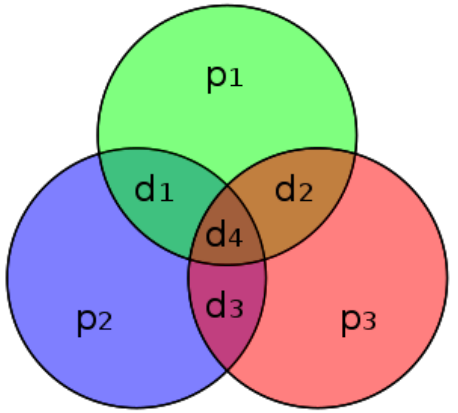
que es el más alto posible para códigos con una distancia mínima de tres (es decir, el número mínimo de cambios de bits necesarios para pasar de una palabra de código a cualquier otra es tres) y la longitud de bloque $2r - 1$.

Debido a la redundancia limitada que los códigos de Hamming agregan a los datos, solo pueden detectar y corregir errores cuando la tasa de error es baja. Este es el caso de la memoria de la computadora (memoria ECC), donde los errores de bits son extremadamente raros y los códigos de Hamming son ampliamente utilizados.

En este contexto, a menudo se usa un código Hamming extendido que tiene un bit de paridad adicional. Los códigos de Hamming extendidos alcanzan una distancia de Hamming de cuatro, lo que permite al decodificador distinguir cuándo se produce un error de un bit como máximo y cuándo se producen errores de dos bits. En este sentido, los códigos extendidos de Hamming son corrección de un solo error y detección de doble error, abreviados como SECDED

Los pasos que implementan el algoritmo para el diseño de un código Hamming son:

1. Numere los bits a partir de 1 (1, 2, 3, 4, 5, 6, 7, etc);
2. Escriba los números de bits en binario (1, 10, 11, 100, 101, 110, 111, etc);
3. Todas las posiciones de bits que son ***potencias de dos son bits de paridad*** (1, 2, 4, 8, etc.);
4. ***Todas las demás posiciones*** de bits, con dos o más bits en la forma binaria de su posición, son ***bits de datos***;
5. ***Cada bit de datos*** se incluye en ***un conjunto único de bits de paridad***, según lo determinado por la forma binaria de su posición de bit:
    1. El bit de paridad 1 cubre todas las posiciones de bit que tienen el conjunto de bits menos significativo: 3, 5, 7, 9, etc.
    2. El bit de paridad 2 cubre todas las posiciones de bit que tienen el segundo conjunto de bits menos significativo: 3, 6, 7, 10, 11, etc.
    3. El bit de paridad 4 cubre todas las posiciones de bit que tienen el tercer conjunto de bits menos significativo: bits 4–7, 12–15, 20–23, etc.
    4. El bit de paridad 8 cubre todas las posiciones de bit que tienen el cuarto conjunto de bits menos significativo: bits 8–15, 24–31, 40–47, etc.
    5. En general, cada ***bit de paridad cubre todos los bits donde el AND bit a bit de la posición de paridad*** y la posición de bit no es cero.



Visualmente

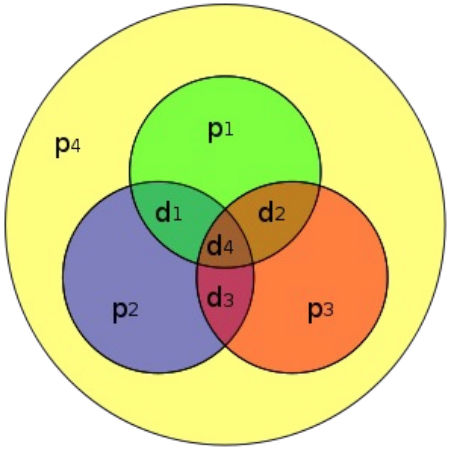
Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✗		✗		✗		✗		✗		✗		✗		✗		✗		✗		...
	p2		✗	✗			✗	✗			✗	✗			✗	✗			✗	✗		
	p4				✗	✗	✗	✗					✗	✗	✗	✗					✗	
	p8								✗	✗	✗	✗	✗	✗	✗	✗						
	p16																✗	✗	✗	✗	✗	

Como m varía, obtenemos todos los códigos de Hamming posibles:

Paridad	Total	Datos	Nombre	Tasa
2	3	1	Hamming(3,1)	1/3 ≈ 0.333
3	7	4	Hamming(7,4)	4/7 ≈ 0.571
4	15	11	Hamming(15,11)	11/15 ≈ 0.733
5	31	26	Hamming(31,26)	26/31 ≈ 0.839
6	63	57	Hamming(63,57)	57/63 ≈ 0.905
7	127	120	Hamming(127,120)	120/127 ≈ 0.945
8	255	247	Hamming(255,247)	247/255 ≈ 0.969
...				
m	$n = 2^m - 1$	$k = 2^m - m - 1$	Hamming$(2^m - 1, 2^m - m - 1)$	$(2^m - m - 1)/(2^m - 1)$

Los códigos de Hamming tienen una distancia mínima de 3, lo que significa que el decodificador puede detectar y corregir un solo error, pero no puede distinguir un error de doble bit de alguna palabra de código de un error de un solo bit de una palabra de código diferente. Por lo tanto, algunos errores de doble bit se decodificarán incorrectamente como si fueran errores de un solo bit y, por lo tanto, no se detectarán, a menos que no se intente corregirlos.

Para remediar esta deficiencia, los códigos de Hamming se pueden extender con un bit de paridad adicional. [Cita requerida] De esta manera, es posible aumentar la distancia mínima del código de Hamming a 4, lo que permite al decodificador distinguir entre errores de un solo bit y dos -bit errores. Así, el decodificador puede detectar y corregir un solo error y al mismo tiempo detectar (pero no corregir) un doble error.



Si el decodificador no intenta corregir errores, puede detectar de manera confiable errores de triple bit. Si el decodificador corrige errores, algunos errores triples se confundirán con errores individuales y se "corregirán" al valor incorrecto. La corrección de errores es, por lo tanto, una compensación entre la certeza (la capacidad de detectar de manera confiable los errores de triple bit) y la resistencia (la capacidad de seguir funcionando ante los errores de un solo bit).