Universidad de Granada

Ingeniería Informática

Computación y Sistemas Inteligentes

Relacion Tema 4

Autor: José Antonio Ruiz Millán

Asignatura: Teoria de la Información y de la Codificación

12 de diciembre de 2018

1. **¿Qué es la distancia de un código? Exponga un ejemplo con un código inventado con k=5 palabras, donde cada palabra tenga n=8 bits.**

   Definimos distancia de un código como **distancia de Hamming de un código**, que se define como la mínima distancia de Hamming existente entre dos palabras de ese código. La distancia de Hamming es el número de símbolos que son distintos entre dos palabras x e y, esto nos permite detectar errores en la transmisión de códigos.

   - **A** $\rightarrow$ 00000000
   - **B** $\rightarrow$ 00000001
   - **C** $\rightarrow$ 00000010
   - **D** $\rightarrow$ 00000011
   - **E** $\rightarrow$ 00000100

   Este código sería un ejemplo de código de $k = 5$ palabras donde cada palabra está codificada con $n = 8$ bits.

2. **Describa el Teorema de Hamming para detección de errores. Para el código expuesto en el ejercicio anterior, indicar cuántos errores de un bit se asegura poder detectar en una palabra mal recibida.**

   El teorema de Hamming nos dice que para códigos de distancia de Hamming d, se pueden detectar $d-1$ errores. Por lo que ahora voy a caclular las distancias de Hamming del ejemplo anterior para comprobar cuantos errores de un bit puede detectar.

   - **D(A,B)** $= 1$

   En este caso, tenemos en el primer caso que $d = 1$, esto ya nos dice sin tener que calcular el resto que podemos detectar $d - 1 = 0$ errores, por lo que éste código no sería nada bueno para la detección de errores.

3. **Supongamos un canal binario simétrico por el que se pueden transmitir bits con una probabilidad de error de p=0.25. ¿Cuál sería la capacidad del canal? ¿Cuántas unidades de tiempo necesitaríamos para enviar 1 bit de información?**

$$C(p) = 1 + p \cdot log_2(p) + (1 - p) \cdot log_2(1 - p)$$

   Por lo que sabiendo esto, tenemos que:

$$C(0{,}25) = 1 + 0{,}25 \cdot log_2(0{,}25) + (1 - 0{,}25) \cdot log_2(1 - 0{,}25) \approx 0{,}189$$

   Con una probabilidad de error de 0,25, por cada unidad de tiempo no podremos enviar más de 0,189 bits.

   Como mínimo necesitaremos 6 unidades de tiempo (código de longitud n=6) para poder transmitir 1 BIT de información por un canal simétrico binario, si la probabilidad de error $p = 0{,}25$.

4. **Suponiendo que codificamos un código uniforme con 4 símbolos (00, 01, 10, 11), ¿sería posible transmitir este código tal cual por el canal binario simétrico con probabilidad de error = 0.1, según el Teorema de Shannon, y poder recuperar la información en el destino?**

Para ello, tenemos primero que comprobar si $C(0{,}1) > 0$ para comprobar que no es un canal inútil. Seguidamente despejaremos los valores necesarios para el cálculo.

$$C(0{,}1) \approx 0{,}531$$

Por lo que con esto sabemos que el canal no es inútil. Sabemos por el *Teorema de Shannon* que se tienen que cumplir las siguientes restricciones:

- $M <= 2^k$

  Donde M es el número de palabras.

- $k/n < C(p)$

Por lo que sabiendo esto, utilizaremos estas fórmulas para despejar y obtener las variables que necesitemos y comprobar si se cumplen.

$$M <= 2^k \Rightarrow$$

$$4 <= 2^k \Rightarrow$$

$$k >= 2$$

Una vez tenemos el valor de k, comprobamos si se cumple la última condición.

$$k/n < C(p) \Rightarrow$$

$$2/2 < 0{,}531 \Rightarrow$$

$$1 \not< 0{,}531$$

Por lo que podemos decir que el código no podría transmitirse por el canal y ser detectado o corregido en el destino.

5. **Genere un código binario uniforme para codificar un alfabeto de la fuente de 4 símbolos, con palabras de longitud n (a escoger) que, según el Teorema de Shannon, pueda utilizarse para recibir el mensaje en el destino correctamente, en un canal binario simétrico con probabilidad de error =0.01, y que asegure detectar 2 errores.**

   Dado el siguiente código:

   - **A** $\rightarrow$ 00001
   - **B** $\rightarrow$ 00110
   - **C** $\rightarrow$ 11100
   - **D** $\rightarrow$ 11011

   Tenemos un código donde $d = 3$ por lo que por el *Teorema de Shannon* sabemos que es capad de detectar $d - 1 = 2$ errores.

   Ahora vamos a comprobar que cumple todas las restricciones:

$$C(0{,}01) \approx 0{,}919 > 0$$

$$M <= 2^k \Rightarrow k >= 2$$

$$k/n < C(p) \Rightarrow 0{,}4 < 0{,}919$$

Por lo que este código cumple todas las restricciones del *Teorema de Shannon* como las restricciones impuestas por el ejercicio.

6. **Exponga un ejemplo de bits de paridad para codificar la palabra (00110101), indicando qué valor debería tomar el bit de paridad en caso de usar paridad par y cuál valor en caso de utilizar paridad impar.**

   - **Paridad par**

     Para este caso necesitamos que toda la cadena de bits sume 0, como la cadena tiene 4 "1" debemos añadir el bit de paridad con valor "0", quedando finalmente **000110101**

   - **Paridad impar**

     Al contrario del caso anterior, necesitamos que la cadena de bits sume 1, como la cadena tiene 4 "1" necesitamos añadir el bit de paridad con valor "1", quedando finalmente **100110101**

7. **¿En qué consiste la paridad bidimensional HVC? ¿Qué es un código P(m,k)? Exponga un ejemplo de codificación HVC con m=2 para transmitir el mensaje (11010110). Introduzca 2 errores aleatorios de un bit en el mensaje e indique el procedimiento para detectar los errores, utilizando dicho ejemplo.**

   La paridad bidemensional consiste en agrupar los bits de paridad agrupando el mensaje por bloques y aplicando bits de paridad horizontal, vertical y cruzada.

   Un **código P(m,k)** es un código de paridad de m bloques con k bits por bloque.

   Para **el ejemplo**, en este caso tenemos un ejemplo del tipo P(2,4), obteniendo la siguiente matriz. Como el enunciado no indica el tipo de paridad, asumo **paridad par**.

1	1	0	1	1
0	1	1	0	0
1	0	1	1	0

   Para este caso, transmitiría **110110110010110**.

   Para el caso de los **errores**, suponemos que el código que recibimos es *010110111010110* donde han sido modificados el primer y noveno bit empezando desde la izquierda.

   - **Primer bit (1 $\rightarrow$ 0)**:

     Para el caso del primer bit, sabemos que los bit de paridad que afectan a este bit son el 5 y el 11 empezando por la izquierda. Si cojemos la primera fila de la matriz, donde este elemento está metido, vemos que no se cumple la paridad. Al comprobar la primera columna, vemos que tampoco se cumple la paridad, como hemos comprobado que la paridad falla en la primera fila y la primera columna, sabemos que el bit que está mal es el bit perteneciente a la primera fila primera columna.

   - **Noveno bit (0 $\rightarrow$ 1)**

     Al igual que en el caso anterior, al comprobar la paridad de la segunda fila, nos daríamos cuenta que no se cumple y por lo tanto tenemos algún fallo en ella. Cuando comenzamos

a comprobar por columnas, cuando lleguemos a la cuarta columna, nos daremos cuenta que tampoco se cumple la paridad, por lo que como sabemos que nos encontramos en la segunda fila, y estamos comprobando la cuarta columna, tenemos que el bit que está mal es el bit que se encuentra en la segunda fila cuarta columna.

8. **¿Qué es un código CRC? ¿En qué se basan los códigos CRC? ¿Qué es un polinomio generador? ¿Cómo se decodifica y detectan errores en el receptor, usando un código CRC? Supongamos que deseamos enviar el mensaje 1101 codificado en un código CRC con polinomio generador $p(x) = x^4 + 1$. ¿Cuántos bits tendrá el código CRC que se enviará? ¿Cómo se codifica el mensaje en el código CRC? Calcule el código CRC que se enviaría. Introduzca un error de 1 bit en el código calculado. Explique y calcule cómo el receptor detectaría el error en el mensaje.**

   Los **CRC** son un tipo de códigos polinómicos. Los códigos polinómicos están basados en modelar el código con polinomios cuyos coeficientes se expresan en módulo 2.

   El **polinomio generador** no es más que un polinomio de grado k que se utiliza como divisor para añadir bits de control de errores. Este polinomio es conocido tanto por el emisor como por el receptor.

   Para **detectar errores** en el receptor se comprueba que es divisible por el polinomio generador (que lo conoce como he especificado anteriormente), si es divisible, el resto debe ser 0 y por lo tanto no hay errores. Si el resto es distinto de 0, entonces tenemos errores en la transmisión.

   Para la última parte, tenemos que seguir el siguiente procedimiento:

   - Añadimos al mensaje original k bits con el valor de "0" al final de la cadena.
   - Una vez tengamos eso, dividimos ésta cadena por el polinomio generador para añadir los bits de redundancia.
   - Sustituimos los últimos k bits que añadimos por el resto de la división.
   - Se envía el mensaje.

   Por lo que tenemos que dividir 11010000/10001 (division de polinomios binario), del cual obtengo como cociente 1101 y de resto 1101, por lo que finalmente el mensaje que se envía es **11011101** y hemos necesitado 8 bits.

   Podemos comprobar que este mensaje se ha creado correctamente ya que si dividmos el mensaje entre el polinomio generador obtenemos de resto 0.

   Para el error, he cambiado el tercer bit, obteniendo **11111101**, para comprobar si el código es correcto o no, tenemos que seguir los siguientes pasos:

   - Comprobar si el código es divisible entre el polinomio generador.
   - Si es así, el resto debe ser 0 y no hay errores.
   - Si el resto es distinto de 0, entonces hay errores.

   Haciendo los cálculos y dividiendo 11111101/10001 (division de polinomios binario) obtenemos de resto 10, por lo que sabemos que tenemos un error en este mensaje.