Códigos Lineales 2da Clase

Daniel Penazzi

9 de junio de 2021

Tabla de Contenidos

- La clase pasada definimos códigos lineales como aquellos que son subespacios vectoriales de $\{0,1\}^n$.
- Y vimos que una forma útil de pensar en ellos es como la imagen de una transformación lineal $T : \{0,1\}^k \mapsto \{0,1\}^n$ donde k es la dimensión del código.
- Y esta forma de pensarlos daba origen a las matrices generadoras del código, que justamente permitia generar fácilmente las palabras del código.
- Pero estamos interesados en corregir errores, y para eso hay otra forma de pensarlos y otras matrices asociadas que son mas útiles para esa función.

- Asi como todo espacio vectorial puede pensarse como la imagen de una cierta transformación lineal, tambien puede pensarse como el nucleo de alguna otra transformación lineal.
- Y al pensarlo de esta forma, se pueden detectar errores fácilmente, y tambien, en algunos casos, corregirlos.
- La parte de detectar es bastante obvia: si $C = Nu(L)$, entonces las palabras de C son exactamente aquellas v para las cuales $L(v) = 0$, asi que es fácil detectar si hubo o no errores de transmisión.
- Como antes, en realidad conviene hacerlo a través de una matriz

Matriz de chequeo

Definición

H es una matriz de chequeo(de paridad) de un código C de longitud n si $C = Nu(H) = \{x \in \{0,1\}^n : Hx^t = 0\}$

- Si bien la definición de arriba no lo requiere, si tenemos mas filas que la dimensión del espacio fila, esas filas extras serán redundantes para el chequeo, pues serán linealmente dependientes de otras filas, asi que de ahora en mas supondremos que la dimensión del espacio fila de H es igual a su número de filas, es decir, que las filas de H son LI.

Matriz de chequeo

- Como dije, veremos que la matriz de chequeo es muy útil para detectar y corregir errores.
- Tambien sirve para definir códigos lineales: dada una matriz H, simplemente definimos $C = Nu(H)$ y C será lineal.
- Viceversa, como todo subespacio vectorial es el nucleo de alguna transformación lineal, y asociada a toda transformación lineal tenemos una matriz, todo código lineal tiene asociada al menos una matriz de chequeo.

Matriz de chequeo:dimensiones

- ¿cuales son, en términos de k y n, las dimensiones de una matriz de chequeo? (en realidad la utilidad de esto es que nos va a permitir calcular n y k en términos del número de filas y columnas de H).
- Como C debe ser $\{x \in \{0,1\}^n : Hx^t = 0\}$, entonces la cantidad de columnas de H debe ser igual a la longitud n de las palabras de C, pues si no no podemos hacer Hx^t.
- Para calcular el número de filas, denotemos por r la cantidad de filas de H (recordemos que estamos suponiendo que las filas de H son LI, de lo contrario el número de filas de H podria ser arbitrariamente grande).

Matriz de chequeo:dimensiones

- Necesitamos un teorema de Álgebra lineal que dice que el rango fila de una matriz es igual al rango columna de esa matriz (y ese número común se llama “rango” de la matriz)
- Es decir, que la dimensión del espacio generado por las filas de una matriz es igual a la dimensión del espacio generado por las columnas de la matriz.
- Como las filas de H son LI, entonces el rango fila de H es igual a r, el número de filas, asi que por el teorema sabemos entonces que r es tambien el rango columna.

Matriz de chequeo:dimensiones

- En realidad no necesitamos el teorema en su forma mas general, pues en el caso particular de que el rango fila es igual al número de filas se deduce fácilmente:
  - reduciendo por operaciones de fila a H, como su rango fila es igual al número de filas, entonces podemos llegar a una matriz equivalente con la identidad a la izquierda.
  - Como esa matriz tiene evidentemente rango columna igual a r y las operaciones por fila no cambian el rango columna, entonces el rango columna de H tambien es r.

Matriz de chequeo:dimensiones

- Ahora que sabemos que $r =$ rango columna, continuemos.
- Como Hx^t es un vector columna con r filas, entonces $(Hx^t)^t = xH^t$ es un vector en $\{0,1\}^r$ con nuestra convención de escribir en forma horizontal los vectores.
- Sea $L : \{0,1\}^n \mapsto \{0,1\}^r$ dada por $L(x) = xH^t$.
- Entonces $Nu(L) = Nu(H) = C$.
- Necesitaremos otro teorema de Álgebra lineal, este mas difícil:
- el teorema del rango-nulidad dice que si $L : V \mapsto W$ es lineal, entonces dim $V =$dim $Im(L)+$dim$Nu(L)$.

Matriz de chequeo:dimensiones

- En nuestro caso, tendriamos que $n =$dim$Im(L)+$dim$Nu(L)$.
- Pero $Nu(L) = Nu(H) = C$ asi que dim$Nu(L) =$dim $C = k$.
- y dim$Im(L) =$(dimensión del espacio columna de H)=(rango columna de H) que vimos que era igual a r.
- Por lo tanto $n = r + k$, y $r = n - k$.
- Conclusión: Si las filas de H son LI, pej, si H “tiene” a la identidad, entonces H es $(n - k) \times n$.
- Mas importante: la dimensión de C, en términos del número de filas y columnas de H es $k = n - r =$número de columnas - número de filas.

Códigos que corrigen un error

- Luego daremos el teorema general y la prueba, pero por ahora, veamos el teorema en la forma en la cual mas lo vamos a usar:

Teorema

Si H es una matriz que no tiene la columna cero ni tiene columnas repetidas, entonces $C = Nu(H)$ corrije al menos un error.
Si ademas alguna columna es suma de otras dos u otras tres columnas, entonces C corrije exactamente un error.

Códigos que corrigen un error

- Por lo tanto, construir códigos que corrigan un error, de la longitud que queramos, es fácil:
- Simplemente tomamos una matriz con n columnas que no tenga la columna cero ni columnas repetidas,
- y tendremos un codigo de longitud n que corregirá (al menos) un error.

Ejemplo

- Sea $C = Nu(H)$ donde H es la matriz:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

- Como H no tiene la columna 0 ni columnas repetidas, entonces C corrije al menos un error.
- Como la ultima columna es suma de las tres primeras, C corrije exactamente un error.
- Como H tiene 7 columnas, la longitud de C es 7.

Dimensión

- ¿Cual es la dimensión de C?
- Vimos que k =número de columnas $-$ número de filas.
- En este caso $k = 7 - 4 = 3$.
- Otra forma de ver esto, si H tiene la identidad adentro, es contar cuantas columnas NO forman parte de la identidad.
- Esto es al revés que con la matriz generadora, en la cual k es la cantidad de filas=cantidad de columnas que SI forman parte de la identidad.
- Resumiendo este ejemplo C es un código de longitud 7, que corrije exactamente un error, y con dimensión 3, por lo tanto tiene $2^3 = 8$ palabras.

Generando palabras

- Como son sólo 8 palabras podriamos pedirles que den todas las palabras.
- O bien, en gral, aún si son mas palabras, pedirles que den algunas palabras.
- Hay dos formas de hacer esto.
- Una bastane eficiente es usando un teorema que daremos luego, que nos da una matriz generadora asociada a la matriz de chequeo.
- Otra forma directamente de la definición de $C = Nu(H) = \{w = w_1 w_2 ... w_7 : Hw^t = 0\}$.

Generando palabras

- Mirando H

Generando palabras

- Mirando H, vemos que $w_1 w_2 \ldots w_7 \in C$ si y solo si:

$$\begin{cases} w_1 \oplus w_5 \oplus w_7 & = 0 \\ w_2 \oplus w_6 \oplus w_7 & = 0 \\ w_3 \oplus w_5 \oplus w_6 \oplus w_7 & = 0 \\ w_4 \oplus w_5 \oplus w_6 & = 0 \end{cases}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Generando palabras

- Mirando H, vemos que $w_1 w_2 ... w_7 \in C$ si y solo si:

$$\begin{cases} w_1 \oplus w_5 \oplus w_7 & = 0 \\ w_2 \oplus w_6 \oplus w_7 & = 0 \\ w_3 \oplus w_5 \oplus w_6 \oplus w_7 & = 0 \\ w_4 \oplus w_5 \oplus w_6 & = 0 \end{cases}$$

- Como vemos, tenemos 4 ecuaciones de paridad.
- Vemos que las variables $w_i, i = 1, 2, 3, 4$ aparecen cada una en sólo una ecuación.
- Esto es consecuencia de que son las variables asociadas a la identidad que aparece en H.

Ejemplo (cont)

- Entonces, para generar el código podemos hacer que las variables dependientes sean w_1, w_2, w_3, w_4, dependiendo de las variables independientes w_5, w_6, w_7
- Volvemos a ver que la dimension del codigo es 3, pues hay 3 variables independientes.
- Las ecuaciones quedan, pasando de miembro las independientes:

Ejemplo (cont)

$$\begin{cases} w_1 = & w_5 \oplus w_7 \\ w_2 = & w_6 \oplus w_7 \\ w_3 = & w_5 \oplus w_6 \oplus w_7 \\ w_4 = & w_5 \oplus w_6 \end{cases}$$

Dandoles todos los valores posibles a w_5, w_6, w_7, generamos el código: (en azul las variables independientes)

$$C = \begin{Bmatrix} 0000000 & 1110001 & 0111010 & 1001011 \\ 1011100 & 0101101 & 1100110 & 0010111 \end{Bmatrix}$$

Corrección de errores

- La matriz H es importante para corregir un error eficientemente.
- Supongamos que queremos detectar y corregir un posible error en la transmisión de v, con el receptor recibiendo w.
- Es decir, vamos a asumir que $d_H(v, w) \leq 1$.
- Si no hubo errores, entonces $w = v$, y como v está en el código, entonces $Hw^t = Hv^t = 0$.
- Supongamos ahora que hubo un error, es decir $d_H(v, w) = 1$
- Entonces v y w difieren en exactamente un bit, digamos el bit j.

Corrección de errores

- Sea e_j el vector que tiene un 1 en el lugar j y 0 en los otros lugares.
- Estamos diciendo entonces que $w = v + e_j$. (suma modulo 2)
- Por lo tanto $Hw^t = H(v^t + e_j^t) = Hv^t + He_j^t$.
- Como $v \in Nu(H)$, entonces $Hv^t = 0$, asi que tenemos:
- $Hw^t = He_j^t$.
- Pero He_j^t es la j-ésima columna de H.

Corrección de errores

- Conclusión: si se recibe una palabra w que está a distancia menor o igual de 1 de una palabra del código, y se calcula Hw^t, entonces:
  1. Si w es una palabra del código, se obtendrá la columna 0.
  2. Si w difiere en el bit j con una palabra del código, se obtendrá la columna j de H
- El receptor no sabe de antemando cual de 1) o 2) va a ser correcta, pero puede calcular Hw^t y ver qué obtiene, dando lugar al siguiente algoritmo para corregir un error:

Corrección de errores

Algoritmo de Corrección de un error

Al recibir la palabra w:

- Calcular Hw^t.
- Si el resultado es la columna 0, aceptar w como la palabra mandada.
- Si el resultado es una columna de H, digamos la columna j, cambiar el bit j de w para recuperar la palabra mandada.
- Si el resultado no es la columna 0 ni una columna de H, hubo mas de un error y no se puede corregir el error de esta forma.

Corrección de errores: Ejemplo

- En el ejemplo que dimos, supongamos que llega la palabra $w = 0101111$.
- Si hacemos Hw^t obtenemos:

$$Hw^t = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Que es la penúltima columna de H.
- Asi que cambiamos ese bit de w y recuperamos la palabra enviada: 0101111

Corrección de errores: Ejemplo

- En el ejemplo que dimos, supongamos que llega la palabra $w = 0101111$.
- Si hacemos Hw^t obtenemos:

$$Hw^t = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Que es la penúltima columna de H.
- Asi que cambiamos ese bit de w y recuperamos la palabra enviada: 0101101

Volviendo al ejemplo

- Recordemos que en el ejemplo, habiamos concluido que

$$C = \begin{Bmatrix} 0000000 & 1110001 & 0111010 & 1001011 \\ 1011100 & 0101101 & 1100110 & 0010111 \end{Bmatrix}$$

- Claramente los vectores que tienen los 3 bits de la parte derecha igual a $100, 010, 001$ son LI y generan a los otros.
- Asi que una matriz generadora es:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Generadora y chequeo

- Si denotamos por I_t a la identidad $t \times t$, vemos que G es de la forma $[A|I_3]$ mientras que H es de la forma $[I_4|B]$.
- Pero si miramos con atención las matrices A y B, notaremos que $A^t = B$!

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Ejemplo

- Esto vale en general, y es la otra forma de generar un código a partir de la matriz de chequeo:
- Le aplico lo que dice el teorema que sigue para encontrar una generadora, y genero el código usando la generadora.

Teorema

Si A es una matriz $r \times k$ entonces $[I_r|A]$ es una matriz de chequeo de un código C si y solo si $[A^t|I_k]$ es una matriz generadora de C. Similarmente, $[A|I_r]$ es una matriz de

chequeo de un código C si y solo si $[I_k|A^t]$ es una matriz generadora de C

Obtención de G a partir de H y viceversa

- Prueba: probaremos el primer caso, el segundo es similar.
- Lo que debemos ver es que $Nu([I_r|A])$ es igual al espacio generado por las filas de $[A^t|I_k]$.
- Llamemos C a lo segundo, y veamos primero que $C \subseteq Nu([I_r|A])$.
- Para ver eso, tomemos un $v \in C$ y veamos que está en $Nu([I_r|A])$.
- Como $v \in C =$ espacio generado por filas de $[A^t|I_k]$, entonces existe $u \in \{0,1\}^k$ tal que $v = u[A^t|I_k]$.

Prueba

- Por lo tanto:

$$\begin{aligned}
[I_r|A]v^t &= [I_r|A](u[A^t|I_k])^t \\
&= [I_r|A]\left[\frac{A}{I_k}\right]u^t \\
&= (I_r.A + A.I_k)\,u^t \\
&= (A + A)u^t \\
&= 0u^t = 0
\end{aligned}$$

- Concluimos que $v \in Nu([I_r|A])$, que era lo que queriamos.

Prueba

- Asi que hemos demostrado que $C \subseteq Nu([I_r|A])$.
- Pero dim $C = k$, pues C es el espacio generado por las filas de $[A^t|I_k]$).
- Y dim$Nu([I_r|A])$ =número de columnas $-$ número de filas.
- Como A tiene k columnas, tenemos dim$Nu([I_r|A]) = (r + k) - r = k$.
- Asi, dim$C$ =dim$Nu([I_r|A])$.
- Como probamos que $C \subseteq Nu([I_r|A])$, concluimos de lo anterior que $C = Nu([I_r|A])$
- Fin

- Este teorema nos permite pasar rapidamente de una matriz generadora si necesitamos codificar o decodificar, a una matriz de chequeo si necesitamos detectar y corregir errores ,y viceversa.
- Nos queda entonces demostrar lo que habiamos dicho sobre H y errores, y ver un algoritmo que nos va a permitir corregir errores usando H.
- Dijimos arriba que vamos a probar que si una matriz de chequeo del código no tiene ni la columna cero ni columnas repetidas, entonces el código corrige un error y que corrige exactamente un error si ademas hay una columna que es suma de dos o tres otras columnas.
- Probaremos primero algo mas general.

δ a partir de matriz de chequeo

Teorema

Si H es matriz de chequeo de C, entonces;

$$\delta(C) = Min\{j : \exists \text{ un conjunto de } j \text{ columnas LD de } H\}$$

(LD es “linealmente dependiente")

- Prueba: Sea
  $s = Min\{j : \exists \text{ un conjunto de } j \text{ columnas LD de } H\}$.
- Probaremos que $\delta(C) \leq s$ y luego que $s \leq \delta(C)$.
- Denotaremos la columna j-ésima de H como $H^{(j)}$.

Continuación prueba

- Por definición de s, existe un conjunto LD de columnas de H: $\{H^{(j_1)}, H^{(j_2)}, ..., H^{(j_s)}\}$.
- Esto significa que existen $c_1, ..., c_s$ no todos nulos tales que:
- $c_1 H^{(j_1)} + c_2 H^{(j_2)} + ... + c_s H^{(j_s)} = 0$
- Sea $w = c_1 e_{j_1} + c_2 e_{j_2} + ... + c_s e_{j_s}$.
- Donde e_j es el vector con todos 0 salvo un 1 en la coordenada j.
- Como no todos los c_j son 0 entonces $w \neq 0$.

Continuación prueba

$$\begin{aligned} Hw^t &= H(c_1 e_{j_1} + c_2 e_{j_2} + \ldots + c_s e_{j_s})^t \\ &= c_1 He_{j_1}^t + c_2 He_{j_2}^t + \ldots + c_s He_{j_s}^t \\ &= c_1 H^{(j_1)} + c_2 H^{(j_2)} + \ldots + c_s H^{(j_s)} \qquad \text{pues } He_j^t = H^{(j)} \\ &= 0 \end{aligned}$$

- Por lo tanto, $w \in C$, pues $C = Nu(H)$.
- Pero su peso es $\leq s$, pues es suma de a lo sumo s e_j
- Y vimos que $w \neq 0$.

Continuación prueba

- En la clase anterior habiamos probado que en el caso de un código lineal,

$$\delta(C) = \text{Min}\{|v| : v \in C, v \neq 0\} \leq |w|$$

- Por lo tanto, como $w \in C$, y $w \neq 0$, entonces w está en ese conjunto, asi que tenemos $\uparrow$.
- Pero su peso es $\leq s$...

Continuación prueba

- En la clase anterior habiamos probado que en el caso de un código lineal,

$$\delta(C) = \text{Min}\{|v| : v \in C, v \neq 0\} \leq |w| \leq s$$

- Por lo tanto, como $w \in C$, y $w \neq 0$, entonces w está en ese conjunto, asi que tenemos $\uparrow$.
- Pero su peso es $\leq s$...
- asi que tenemos la desigualdad que queriamos.
- Ahora veamos la otra desigualdad.

Continuación prueba

- Sea $v \in C$ tal que $\delta(C) = |v|$.
- $\delta(C) = |v|$ implica que existen $i_1, ..., i_{\delta(C)}$ con $v = e_{i_1} + e_{i_2} + \cdots + e_{i_{\delta(C)}}$.
- Como $v \in C$, entonces $Hv^t = 0$.
- Con el mismo cálculo que antes, concluimos que $H^{(i_1)} + H^{(i_2)} + ... + H^{(i_{\delta(C)})} = Hv^t = 0$.
- Pero $H^{(i_1)} + H^{(i_2)} + ... + H^{(i_{\delta(C)})} = 0$ dice que $\{H^{(i_1)}, H^{(i_2)}, ..., H^{(\delta(C))}\}$ es un conjunto LD de columnas de H.
- Por lo tanto $s = Min\{j : \exists \text{ un conjunto de } j \text{ columnas LD de } H\} \leq \delta(C)$
- y hemos completado la prueba.

Corolario

Corolario

Si H es una matriz que no tiene la columna cero ni tiene columnas repetidas, entonces $C = Nu(H)$ corrije al menos un error. Si ademas alguna columna es suma de otras dos u otras tres columnas, entonces C corrije exactamente un error.

- Veamos primero que $\delta(C) \geq 3$
  1. Si no tiene la columna cero, entonces $\delta(C) \geq 2$
     - Pues $\delta(C) = 1$ diria que existe un conjunto $\{H^{(j)}\}$ LD
     - El único conjunto de cardinalidad 1 que es LD es el conjunto $\{0\}$.
     - Asi que si $\delta(C) = 1$, tendriamos que existe j con $H^{(j)} = 0$, absurdo.

Prueba del corolario

- Ya sabemos entonces que $\delta(C) \geq 2$.
  2. Supongamos ahora que $\delta(C) = 2$
     - Tendriamos un conjunto $\{H^{(j_1)}, H^{(j_2)}\}$ LD
     - Esto dice que existen c_1, c_2, no ambos nulos, tales que $c_1 H^{(j_1)} + c_2 H^{(j_2)} = 0$.
     - Pero en realidad ninguno puede ser cero, pues si uno fuese cero, pej, c_1, entonces tendriamos $H^{(j_2)} = 0$, contradice la hipotesis.
     - Entonces, como estamos en $\{0, 1\}$ y $c_1, c_2 \neq 0$, tenemos que $c_1 = c_2 = 1$.
     - Asi, $H^{(j_1)} + H^{(j_2)} = 0 \quad \therefore \quad H^{(j_1)} = -H^{(j_2)}$
     - Como estamos en $\{0, 1\}$ entonces $1 = -1, \ \therefore \quad H^{(j_1)} = H^{(j_2)}$
     - Lo que contradice la hipotesis de que no hay dos columnas iguales.
- Entonces, hemos probado que $\delta(C) \geq 3$.
- Por lo tanto tenemos la primera parte del corolario, pues C corrige al menos $\lfloor \frac{3-1}{2} \rfloor = \lfloor \frac{2}{2} \rfloor = 1$ error.

Prueba del corolario

- Supongamos ahora que hay una columna que es suma de otras dos: $H^{(j_1)} = H^{(j_2)} + H^{(j_3)}$
- entonces $H^{(j_1)} + H^{(j_2)} + H^{(j_3)} = 0$
- Lo que dice que $\{H^{(j_1)}, H^{(j_2)}, H^{(j_3)}\}$ es LD.
- Por lo tanto $\delta(C) \leq 3$.
- Como vimos que si no tiene la columna cero ni columnas repetidas entonces $\delta(C) \geq 3$, concluimos que en este caso $\delta(C) = 3$
- Y corrije exactamente un error, pues $\lfloor \frac{\delta(C)-1}{2} \rfloor = \lfloor \frac{2}{2} \rfloor = 1$

Prueba del corolario

- Si hay una columna que es suma de otras tres: $H^{(j_1)} = H^{(j_2)} + H^{(j_3)} + H^{(j_4)}$
- entonces $H^{(j_1)} + H^{(j_2)} + H^{(j_3)} + H^{(j_4)} = 0$
- Lo que dice que $\{H^{(j_1)}, H^{(j_2)}, H^{(j_3)}, H^{(j_4)}\}$ es LD.
- Por lo tanto $\delta(C) \leq 4$.
- y $\lfloor \frac{\delta(C)-1}{2} \rfloor \leq \lfloor \frac{3}{2} \rfloor = 1$.

Consecuencias del corolario

- Algunas consecuencias ya las vimos.
- Si queremos construir un código que corrija un error, basta tomar una matriz de las dimensiones adecuadas, que no tenga la columna cero ni columnas repetidas.
- Por ejemplo, a partir de la matriz de chequeo

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

- habiamos generado el código

$$C = \begin{Bmatrix} 0000000 & 1110001 & 0111010 & 1001011 \\ 1011100 & 0101101 & 1100110 & 0010111 \end{Bmatrix}$$

Consecuencias del corolario

- Ese código tiene $k = 3$ (por eso tiene 8 palabras), $n = 7$ y habiamos dicho que corregia un error porque no tenia la columna cero ni columnas repetidas pero la ultima columna era suma de las tres primeras.
- Les podriamos preguntar que calcularan δ.
- Es decir, por el teorema, sabemos que $3 \leq \delta \leq 4$.
- Para saber cuantos errores corrije, da igual si es 3 o 4, en ambos casos $\lfloor \frac{\delta(C)-1}{2} \rfloor$ es igual a 1.
- Pero si le pedimos que calculen δ en este caso tendrian que dar un paso mas.

Consecuencias del corolario

- O bien prueban que no hay ninguna columna que es suma de otras dos para ver que $\delta = 4$
- O bien dan una columna que es suma de otras dos para ver que $\delta = 3$
- O bien, si la cantidad de palabras no es muy grande (como en este caso) pueden calcular δ directamente de $\delta(C) = \text{Min}\{|v| : v \in C, v \neq 0\}$

Consecuencias del corolario

- En este caso nos conviene la ultima estrategia, y mirando todas las palabras no nulas, vemos que todas tienen peso 4, asi que $\delta = 4$.
- Podriamos hacernos un par de preguntas:
  1. ¿Existe un código con 8 palabras que corrija un error mas eficiente que C?
  2. ¿Existe un código con 8 palabras y $\delta = 4$ mas eficiente que C?
- Por “mas eficiente" nos referimos a que por cada 3 bits de información que queramos mandar, el C de nuestro ejemplo debe mandar 7 bits, y querriamos saber si podemos hacer lo mismo con menos bits.

Ejemplo II

- En el caso de la primera pregunta la respuesta es si.
- Pues podriamos tomar $\tilde{C} = Nu(\tilde{H})$

$$\tilde{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- La cual tambien tiene k =número de columnas-número de filas$= 6 - 3 = 3$ asi que $\tilde{C}$ tiene 8 palabras.
- Y no tiene la columna 0 ni columnas repetidas, asi que $\tilde{C}$ también corrije al menos un error.
- (dado que la cuarta columna es suma de las 2 primeras, corrije exactamente un error)

Ejemplo II

- Pero $\tilde{C}$ tiene longitud 6.
- Es decir, para mandar 3 bits de información con posibilidad de corregir un error, C requiere 4 bits extras, pero $\tilde{C}$ sólo 3.
- Asi que $\tilde{C}$ es mas eficiente que C.
- Pero como $\delta = 3$, este no es un buen ejemplo para la pregunta 2.
- Antes de analizar la pregunta 2, sigamos preguntandonos si no podemos mejorar $\tilde{C}$ respecto del criterio de la pregunta 1.
- Para mejorarlo deberiamos usar un código de longitud 5.
- Como necesitamos 8 palabras, k debe ser 3.
- Entonces tendriamos que tener una matriz $r \times (r + 3)$.

Posibilidad de mejora

- Como queremos $r + 3 = 5$, entonces $r = 2$ y necesitamos una matriz 2×5
- Pero tenemos que tener todas las columnas distintas (y no nulas)
- Y con sólo 2 filas sólo tenemos 4 columnas posibles (y una de ellas la nula)
- Asi que no podemos mejorar a $\tilde{C}$ con un código lineal
- Si usamos la cota de Hamming vemos que tampoco podriamos mejorarlo con ningún tipo de código, lineal o no lineal, que corrija un error, pues deberiamos tener
- $8 \leq \frac{2^5}{1+5} = \frac{32}{6} = 5,33...$, absurdo.

Posibilidad de mejora pregunta 2

- Bueno, ahora vamos a la pregunta 2 sobre C. Por el razonamiento anterior, sabemos que no puede ser de longitud 5, asi que deberia ser uno de longitud 6, es decir la matriz será 3×6.
- Queremos $\delta = 4$ asi que necesitamos que ninguna columna sea suma de otras dos.
- Aun sin la matriz no tiene la identidad, podemos hacer reducción gaussiana y llegar a una matriz equivalente por filas (por lo tanto con el mismo nucleo) que tenga la identidad.
- Entonces podemos suponer que la matriz será de la forma

$$\begin{bmatrix} 1 & 0 & 0 & * & * & * \\ 0 & 1 & 0 & * & * & * \\ 0 & 0 & 1 & * & * & * \end{bmatrix}$$

Posibilidad de mejora pregunta 2

- Pero no queremos que ninguna columna sea suma de otras dos.
- Como tenemos la identidad eso significa que ninguna de las columnas con $*$ puede tener solo dos unos, pues en ese caso seria suma de dos de las primeras tres columnas.
- Pero eso significa, como solo hay 3 filas, que las ultimas tres columnas son todas iguales a la columna $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
- Absurdo pues esto significaria que $\delta = 2$.

Posibilidad de mejora

- Conclusión: podemos mejorar C con $\tilde{C}$ si sólo nos interesa la capacidad de corrección de errores, y no podemos mejorar C si tambien queremos $\delta = 4$.
- En general, supongamos que les pedimos dar una matriz de chequeo lo mas chica posible que sirva para codificar 2^k palabras con un código lineal que corrija al menos un error.
- Sea r el número de filas de una posible matriz de chequeo.

Matrices óptimas

- Para minimizar el número de filas, queremos que sean LI.
- Como necesitamos codificar 2^k palabras, entonces la matriz será $r \times (r + k)$.
- Queremos corregir al menos un error, asi que queremos que nuestra matriz de chequeo tenga todas las columnas distintas, pero ademas no debe tener la columna 0.
- Entonces, tenemos que elegir el menor r tal que exista una matriz $r \times (r + k)$ sin la columna cero, con todas las columnas distintas.

Matrices óptimas

- El número total de distintas columnas que tienen r filas es 2^r, tomando todas las posibles combinaciones de 1s y 0s.
- Como no podemos tener la columna 0, esto nos deja con $2^r - 1$ columnas posibles.
- Por lo tanto necesitamos que $r + k \leq 2^r - 1$ para poder tener $r + k$ columnas distintas y no nulas.
- Viceversa, si $r + k \leq 2^r - 1$, de todas las $2^r - 1$ columnas no nulas posibles, elegimos $r + k$ para formar nuestra matriz.
- Asi que $r + k \leq 2^r - 1$ es una condición necesaria y suficiente.

Ejemplos

- Ejemplo, supongamos que queremos un código lineal con 1024 palabras del menor tamaño que corrija un error.
- Como 1024=2^{10} queremos r con $r + 10 \leq 2^r - 1$.
- Como $2^3 = 8 < 10$ tenemos que empezar con al menos 4.
- Si $r = 4$, tenemos $4 + 10 = 14 < 15 = 2^4 - 1$, asi que una matriz 4×14 es la menor.
- Si en vez de pedir 1024 palabras hubieramos pedido 4096, como $4096 = 2^{12}$ ahora 4 no nos serviria pues $2^4 - 1 = 15 < 4 + 12$.
- La menor sería entonces una 5×17.

Cambiando el punto de vista

- Todo esto desde el punto de vista de, DADO k, hallar el óptimo r.
- Pero podriamos preguntarnos al revés: dado un r, ¿hasta cual k podemos llegar?
- Bueno, lo que vimos antes nos da la respuesta: el máximo k es aquel para el cual $2^r - 1 = r + k$ es decir $k = 2^r - 1 - r$
- Consiste en tomar una matriz con TODAS las $2^r - 1$ columnas no nulas posibles.
- Esos códigos se llaman códigos de Hamming.

Códigos de Hamming

- Por supuesto, dependiendo de el orden en que ordenemos las columnas, tendremos distintos códigos, pero a cualquiera de ellos se le llama un “código de Hamming".
- A pesar de que si cambiamos el orden de las columnas los códigos serán distintos, son todos, para un r fijos, equivalentes entre si, porque las palabras de uno son permutación de los bits de las palabras de otro.
- Por esta razón, a pesar de la ambigüedad del orden, denotaremos por $\mathcal{H}_r$ a cualquier código de Hamming con r filas. (con $r \geq 2$).

Propiedades de los códigos de Hamming

$\mathcal{H}_r$:

1. Tiene r filas, obviamente.
2. Tiene dimensión $k = 2^r - 1 - r$, como vimos.
3. Por lo tanto, tiene 2^{2^r-1-r} palabras.
4. Tiene $\delta = 3$, pues
   1. por construcción no tiene la columna 0 ni columnas repetidas, asi que $\delta \geq 3$
   2. pero como tiene todas las columnas, en particular tambien tiene las columnas e_1^t, e_2^t y $(e_1 + e_2)^t$ en algún lugar, asi que $\delta \leq 3$.
5. Por lo tanto corrije exactamente un error.

Ejemplos de matrices de chequeo de Hamming

$$r = 2 \mapsto \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

- es una matriz 2×3 que da origen a un código de Hamming $\mathcal{H}_2$.
- El código tiene dimensión $3 - 2 = 1$ asi que tiene sólo dos palabras:
- es $C = \{000, 111\}$

Ejemplos de matrices de chequeo de Códigos de Hamming

- Para el siguiente $r = 3$ seria una matriz como

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- Tiene longitud 7 y $2^4 = 16$ palabras.
- Para $r = 4$, tendriamos longitud $2^4 - 1 = 15$ y $2^{15-4} = 2^{11} = 2048$ palabras.
- Y para $r = 5$ tenemos longitud $2^5 - 1 = 31$ y $2^{26} \simeq 67$ millones de palabras.

Perfección

Los códigos de Hamming son muy usados por varias razones. Una de ellas es:

Propiedad

Los códigos de Hamming son perfectos

- Prueba:Hay que ver que $\#\mathcal{H}_r = \frac{2^n}{1+n+\binom{n}{2}+\ldots+\binom{n}{t}}$ con $t = \lfloor \frac{\delta-1}{2} \rfloor$.
- En nuestro caso $n = 2^r - 1$, $|\mathcal{H}_r| = 2^{2^r-1-r}$ y $t = 1$ asi que simplemente coloquemos los números:

Perfección

$$
\begin{aligned}
\frac{2^n}{1+n+\binom{n}{2}+\ldots+\binom{n}{t}} &= \frac{2^n}{1+n} && (t=1)\\
&= \frac{2^n}{1+2^r-1} && (n=2^r-1)\\
&= \frac{2^n}{2^r}\\
&= 2^{n-r} = 2^{2^r-1-r} && (n=2^r-1)\\
&= \#\mathcal{H}_r
\end{aligned}
$$

Ordenes óptimos de las columnas de un código de Hamming

- Si bien todos los ordenes posibles de columnas para $\mathcal{H}_r$ son equivalentes, hay cuatro ordenes que son los mas usados, y que le dan a los códigos de Hamming otra ventaja respecto de otros códigos lineales.
- Uno podria suponer que son alguno con la identidad ya sea a derecha o a izquierda, pero no.
- Es un raro caso en donde NO CONVIENE poner la identidad toda a derecha o toda a izquierda.
- El orden “óptimo" es el que tiene como columna i-esima la representacion binaria del numero i.

Orden óptimo de las columnas

- Hay cuatro ordenes “óptimos", dependiendo de:
  1. Si contamos las columnas desde la izquierda o desde la derecha
  2. Si representamos los números con el bit mas representativo arriba o abajo.
- Por ejemplo si contamos desde la izquierda y con el bit mas representativo abajo (por lo tanto el menos representativo arriba) tendriamos para $r = 3$ la siguiente matriz:

$$r = 3 \mapsto \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Ejemplos

y para $r = 4$:

$$\begin{bmatrix} 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \end{bmatrix}$$

Justificación del orden óptimo de las columnas

- ¿Por qué este orden es mejor que un orden con la identidad a izquierda, pej?
- Para la corrección de errores.
- Recordemos que si queremos corregir un error, y nos llega w, tenemos que hacer Hw^t.
  1. Si $Hw^t = 0$, tomamos $v = w$ como palabra mandada.
  2. Si $Hw^t = H^{(j)}$, tomamos $v = w + e_j$ como palabra mandada.
- El problema es que si no nos da 0, debemos chequear Hw^t contra todas las columnas de H, hasta encontrar el j para el cual $Hw^t = H^{(j)}$.
- En cambio con el orden ese que dimos, lo único que hay que hacer es leer Hw^t como un número.

Justificación del orden óptimo de las columnas

- Es decir, para un código de Hamming con el orden anterior, el algoritmo de corrección es:
  1. Si $Hw^t = 0$, tomamos $v = w$ como palabra mandada.
  2. Si $Hw^t \neq 0$, tomamos $v = w + e_j$ como palabra mandada, donde j es tal que su representación binaria es la columna Hw^t.
- Mas aún, calcular Hw^t es mucho mas fácil:
- Hw^t es la suma de las columnas de H correspondientes a los lugares donde w tiene un 1.
- Pero la columna j-ésima de H es j (en binario).
- Asi que calcular Hw^t es simplemente hacer suma (modulo 2, i.e., xor) de los js correspondientes a los bits 1 de w.

Ejemplo

- Como las computadoras trabajan internamente todas con representación binaria, basta un par de trucos de representación de las columnas para codificar todo esto en forma eficiente.
- Veamos un ejemplo con la matriz 4×15 anterior.
- Supongamos que llega $w = 100000001001000$.
- Tiene 1s en los lugares 1,9,12 asi que “sumamos” 1,9 y 12 (en binario, bit a bits)
- Es decir 0001+1001+1100=0100
- Como 0100 es el número 4, cambiamos el 4to bit:

Ejemplo

- Como las computadoras trabajan internamente todas con representación binaria, basta un par de trucos de representación de las columnas para codificar todo esto en forma eficiente.
- Veamos un ejemplo con la matriz 4×15 anterior.
- Supongamos que llega $w = 100000001001000$.
- Tiene 1s en los lugares 1,9,12 asi que “sumamos” 1,9 y 12 (en binario, bit a bits)
- Es decir 0001+1001+1100=0100
- Como 0100 es el número 4, cambiamos el 4to bit: 100000001001000.

Ejemplo

- Como las computadoras trabajan internamente todas con representación binaria, basta un par de trucos de representación de las columnas para codificar todo esto en forma eficiente.
- Veamos un ejemplo con la matriz 4×15 anterior.
- Supongamos que llega $w = 100000001001000$.
- Tiene 1s en los lugares 1,9,12 asi que “sumamos” 1,9 y 12 (en binario, bit a bits)
- Es decir 0001+1001+1100=0100
- Como 0100 es el número 4, cambiamos el 4to bit: 100100001001000.

Ejemplo

- Si llega $\tilde{w} = 000110100100000$
- Tiene bits no nulos en los lugares 4,5,7,10.
- Sumamos 4,5,7,10 en binario bit a bit:
- $0100 + 0101 + 0111 + 1010 = 1100$ que es el número 12.
- Por lo tanto la palabra enviada era
  $v = \tilde{w} + e_{12} = 000110100101000$

Hamming extendidos

- Los códigos de Hamming corrigen un error, pues $\delta = 3$.
- De acuerdo con la definición de “detectar” , detectan $\delta - 1 = 2$ errores.
- Pero aca hay una sutileza con la definición de “detectar” 2 errores. (quizas deberian haber usado otra palabra, pero ya ha quedado como estandard “detectar”)
- Detectar 2 errores significa que si se producen 2 errores o menos, el receptor se va a dar cuenta que la palabra que recibe NO ES la palabra que le mandaron.
- Pero lo que NO significa es que el receptor pueda saber si hubo 1 error o si hubieron 2 errores.

Hamming extendidos

- Y justamente en el caso de Hamming, si se producen 2 errores, no hay forma de darse cuenta que hubo 2 errores y no 1, pues la suma de dos columnas va a dar otra, siempre.
- Asi que cuando se calcule Hw^t, va a dar alguna de las columnas de H tanto si hubo un solo error como si hubo 2.
- Por lo tanto dos errores SIEMPRE serán corregidos como si hubiera habido uno solo, y por lo tanto corregido mal.
- Los códigos de Hamming “extendidos” resuelven este problema.
- De todos modos esto sólo sirve en caso de poder pedir retransmisión, pues si bien con los códigos de Hamming extendidos podemos darnos cuenta que ha habido 2 errores y no uno sólo, no podemos corregirlos.
- Veamos como son.

Bit de Paridad

- Dado un código C de longitud n, podemos agregarle a C un bit de paridad, formando:
  $C^+ = \{v_1 \ldots v_n v_{n+1} : v_1 \ldots . v_n \in C \& v_{n+1} = v_1 + v_2 + \ldots . + v_n\}$.
- (donde, recordemos, $+$ es la suma modulo 2)
- C^+ se puede construir para cualquier código, pero si C es lineal, es obvio que C^+ tambien lo es.
- Calculemos $\delta(C^+)$ en términos de $\delta(C)$.

$\delta(C^+)$

- Sean $v, w \in C^+$, $v \neq w$.
- Como v_{n+1} se calcula en base a los otros v_is, y w_{n+1} se calcula en base a los otros w_is, entonces $v \neq w$ implica que $v_1....v_n \neq w_1....w_n$.
- Sea $d = d_H(v_1....v_n, w_1....w_n)$.
- Si d es impar, entonces tenemos una cantidad impar de w_is distintos de los v_i y por lo tanto: $w_1 + ... + w_n = 1 + v_1 + ... + v_n$.
- Es decir, $w_{n+1} = 1 + v_{n+1}$ y esto es $\neq v_{n+1}$
- Lo cual dice que $d_H(v, w) = d + 1$.

$\delta(C^+)$

- Si d es par, tendremos una cantidad par de w_is distintos de los v_i y por lo tanto: $w_1 + ... + w_n = v_1 + ... + v_n$, es decir $w_{n+1} = v_{n+1}$
- En este caso, tendremos $d_H(v, w) = d$
- Esto vale para todas las palabras $v \neq w$ del código.
- En particular, tomando v, w tales que $\delta(C) = d_H(v_1....v_n, w_1....w_n)$ concluimos que:

$$\delta(C^+) = \begin{cases} \delta(C) + 1 & \text{si } \delta(C) \text{ es impar} \\ \delta(C) & \text{si } \delta(C) \text{ es par} \end{cases}$$

Ejemplo

- Un ejemplo de esta construcción es C2: C2 es $(C1)^+$.
- Como $\delta(C1) = 1$ es impar, por eso $\delta(C2) = 1 + 1 = 2$.
- Pero si quisieramos “iterar" la construcción y construir $(C2)^+$, no ganariamos nada.
- Pues al ser $\delta(C2)$ par, tendriamos $\delta((C2)^+) = \delta(C2) = 2$
- Asi que tendriamos un bit mas sin ningún beneficio.

Hamming extendidos

- Los códigos $\mathcal{H}_r^+$ son los llamados "códigos de Hamming extendidos"
- Dado que los códigos de Hamming $\mathcal{H}_r$ tienen longitud $2^r - 1$, entonces los Hamming extendidos tienen longitud 2^r.
- Como $\delta(\mathcal{H}_r) = 3$, entonces $\delta(\mathcal{H}_r^+) = 3 + 1 = 4$.
- Debido a eso, los Hamming extendidos pueden diferencia entre el caso en que ha habido 2 errores y el caso de 1 error.
- Pues supongamos que $v \in \mathcal{H}_r^+$ se envia y llega w que sólo sabemos que tuvo 1 o 2 errores.

Hamming extendidos

- Si tuvo 1 error, lo vamos a corregir correctamente.
- Si tuvo 2 errores, $w = v + e_i + e_j$ pero supongamos que se pudiera corregir como si fuera 1 error.
- Corregiriamos w a $w + e_k$ para algún k.
- Pero eso significaria que $w + e_k$ está en el código.
- Como v tambien está en el código y $\delta = 4$, tendriamos que $4 \leq d(v, w + e_k)$.
- Pero $d(v, w + e_k) = d(v, v + e_i + e_j + e_k) = 3$, absurdo.
- Asi que si hay 2 errores no se podrá corregir como si hubiera uno solo.

Singleton Bound

Teorema: Cota de Singleton

Si C es un código lineal de longitud n y dimensión k, entonces;

$$\delta(C) \leq n - k + 1$$

- Nota: Códigos con $\delta = n - k + 1$ se llaman MDS codes (Maximum Distance Separable Codes) y son extremadamente importantes no sólo en transmisión de datos sino en criptografía.
- De hecho el algoritmo estandard de cifrado que usan los bancos y una buena parte de las páginas web (AES) usa códigos MDS.

Prueba

- Prueba: Sea H matriz de chequeo de C con filas LI.
- Habiamos visto que H es $r \times n$ con $r = n - k$ y que ademas r es el rango columna pues las filas son LI.
- Por lo tanto, al ser r la dimensión del espacio columna, cualquier conjunto con MAS de r columnas es LD.
- En particular cualquier conjunto de $r + 1$ columnas es LD.
- Como

$$\delta(C) = Min\{j : \exists \text{ un conjunto de } j \text{ columnas LD de } H\}$$

concluimos que $\delta(C) \leq r + 1 = n - k + 1$

- ¿Qué pasa si queremos corregir mas de un error?
- Para construirlos, es mas díficil en general.
- Por ejemplo, si queremos construir un código que corrija dos errores, necesitamos $\delta \geq 5$.
- La matriz debe:
  1. No tener la columna 0.
  2. No tener columnas repetidas
  3. No tener ninguna columna que sea suma de otras dos columnas
  4. No tener ninguna columna que sea suma de otras tres columnas
- Aun sin la matriz no tiene la identidad, podemos hacer reducción gaussiana y llegar a una matriz equivalente por filas (por lo tanto con el mismo nucleo) que tenga la identidad.
- Pej, de la forma $[I|A]$.

- Por los requisitos de la pagina anterior, las columnas de A deben tener todas al menos 4 unos, pues si tuviera una columna con 3 unos, esa columna seria suma de tres columnas de la identidad.
- Pero no basta con esto, pues debemos luego mirar que pasa entre las columnas de A.
- Pej,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

  corrige dos errores pero tiene dimensión sólo 1.
- Si queremos mas columnas, para mayor dimensión vamos a necesitar mas filas

- Con

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- Vemos que la última columna es la suma de la primera ,quinta y sexta, asi que no nos sirve
- En general es dificil construir de esta forma este tipo de códigos en forma eficiente, tratando de que se satisfagan todas esas condiciones, especialmente para δs grandes.
- En un curso mas largo veriamos algunas formas mejores de crear códigos que corrijan muchos errores.
- Pero hay otras formas, que quizas no den los códigos mas eficientes posibles pero podemos darles un pantallazo rápido en este curso.

Códigos de repetición

- Sea C un código de longitud n.
- Definimos el código de repetición r veces a partir de C que denotaremos como $Rep_r(C)$ de la siguiente forma:

$$Rep_r(C) = \left\{ \begin{array}{lcc} v \in \{0,1\}^{nr} & : & v_1 v_2 \dots v_n \in C \\ & \& & \\ v_1 v_2 \dots v_n & = & v_{n+1} v_{n+2} \dots v_{2n} \\ & = & v_{2n+1} v_{2n+2} \dots v_{3n} \\ & = & \dots. \\ & = & v_{(r-1)n+1} v_{(r-1)n+2} \dots v_{rn} \end{array} \right\}$$

Es decir, las palabras de $Rep_r(C)$ son de la forma $v...v$ con $v \in C$, donde hay r vs.

Observar que la cantidad de palabras de C y $Rep_r(C)$ es la misma.

Códigos de repetición: δ

- Esta claro que si C es lineal entonces $Rep_r(C)$ tambien lo es.
- De hecho, $\mathcal{H}_2$ es $Rep_3(\{0,1\})$.
- Es fácil calcular el δ: si $v, w \in C$ tienen d bits de diferencia.
- Entonces $v....v$ y $w....w$ tendrán dr bits de diferencia.
- Por lo tanto $\delta(Rep_r(C)) = r\delta(C)$
- Y esto nos permite construir códigos con δ arbitrariamente grandes.
- Claro que a un costo grande en cantidad de bits extras.

Códigos de repetición

- Si quisieramos un código capaz de corregir 17 errores y que pueda encodear 2 bits de información, necesitariamos que $17 = \lfloor \frac{\delta-1}{2} \rfloor$ entonces necesitamos $\delta = 35$ como mínimo.
- Si partimos de $\{0,1\}^2$ como código base, podemos construir $C = Rep_{35}(\{0,1\}^2)$.
- Como $\{0,1\}^2$ tiene $\delta = 1$, entonces $\delta(C) = 35$ y corrije 17 errores.
- Pero para transmitir 2 bits de información tendremos que transmitir 70 bits en total.
- Se pueden construir códigos que corrijan 17 errores y que sean mas eficientes.

- Por ejemplo, vimos que C5 tiene $\delta = 3$ y $n = 5$, por lo tanto $Rep_r(C5)$ tendrá $\delta = 3r$ y longitud $5r$.
- Asi que $Rep_{12}(C5)$ tendrá $\delta = 36$ y por lo tanto corregirá 17 errores.
- Pero su longitud será $5 \times 12 = 60$, menor que los 70 bits de $Rep_{35}(\{0,1\}^2)$.
- En gral si bien no es lo óptimo, hacer repetición de un código mas chico elegido adecuadamente permite definir rapidamente un código de acuerdo con las necesidades.
- Pero en gral repetir el $\{0,1\}^n$ no da buenos resultados.

Mas ejemplos

- Supongamos que quiero un código con 8 palabras que corrija 5 errores.
- Necesito $\delta = 11$ al menos.
- Podemos usar $Rep_{11}(\{0,1\}^3)$, pero la longitud es 33.
- Podemos usar el primer código C que dimos como ejemplo de matriz de chequeo.
- Como tiene $\delta = 4$, entonces $Rep_3(C)$ tiene $\delta = 12$ y tiene longitud $3 \times 7 = 21$, sustancialmente menor que 33.

Mas ejemplos

- Tambien podriamos usar el código $\tilde{C}$ que dimos.
- Como $\delta(\tilde{C}) = 3$, $Rep_4(\tilde{C})$ tiene $\delta = 12$ tambien.
- Pero tiene longitud $4 \times 6 = 24$, asi que conviene $Rep_3(C)$.
- Otro ejemplo: Supongamos que necesitamos un código que tenga 2048 palabras y que corrija 37 errores para una situación con alta taza de errores, pej, el espacio interplanetario.
- Necesitamos $\delta = 75$ y tenemos $k = 11$.

Mas ejemplos

- Podemos usar $Rep_{75}(\{0,1\}^{11})$.
- La longitud es $75 \times 11 = 825$ bits por cada palabra.
- En vez de eso, podemos usar repetición de $\mathcal{H}_4$ pues $\mathcal{H}_4$ tiene dimensión $2^4 - 4 - 1 = 11$ por lo tanto 2048 palabras.
- Como $\delta(\mathcal{H}_4) = 3$, entonces $Rep_t(\mathcal{H}_4)$ tiene $\delta = t \times 3$ asi que si tomamos $t = 25$ cumple que $\delta(Rep_{25}(\mathcal{H}_4)) = 75$ que es lo que queremos.
- Pero su longitud es $25 \times 15 = 375$, mucho mas eficiente que los 825 de arriba.
- $\mathcal{H}_4^+$ tiene la misma dimensión y $\delta(\mathcal{H}_4^+) = 4$, por lo tanto $Rep_{19}(\mathcal{H}_4^+)$ tiene $\delta = 19 \times 4 = 76 > 75$ tambien nos sirve.
- Y su longitud es $19 \times 16 = 304$, es incluso mas eficiente.