Códigos de Corrección de Errores Lineales 1ra Clase (2da de Códigos)

Daniel Penazzi

4 de junio de 2021

Tabla de Contenidos

Definición de códigos lineales y repaso
Definición
Subespacios vectoriales sobre el cuerpo $\{0, 1\}$.
Ejemplos
δ de códigos lineales
Codificación y Decodificación
Dimensión de un código lineal
Codificación
Transformaciones lineales y matrices
Matriz Generadora
Ejemplos
Ventajas de los códigos lineales

Códigos lineales

- En la clase pasada vimos la teoría base sobre códigos. (de bloque, binarios).
- Recordemos que los códigos binarios de bloque son subconjuntos de $\{0, 1\}^n$ para algún n.
- Pero ¿que tal si pedimos "mas" que sólo ser subconjunto?
- Es decir, requerir algún tipo de estructura algebraica sobre nuestro código.
- Al tener una estructura mas rica, pueden quizás deducirse mas cosas sobre el código u operar mas fácilmente.
- Esto es exactamente lo que pasa con los códigos lineales

Códigos lineales

Definición:

Un código lineal de longitud n es un subespacio vectorial de $\{0, 1\}^n$.

- Varios de uds. probablemente se hayan olvidado de álgebra lineal, asi que haremos un repaso rápido de algunos conceptos.
- Si bien no es necesario acordarse de todo lo que vieron en álgebra lineal, si es necesario sentirse cómodo cuando hablemos de “espacios vectoriales” y manejando matrices. (aunque en nuestro caso serán matrices de 0s y 1s, mas fáciles que las que vieron en álgebra lineal).

- Recordemos que dado un cuerpo $I\!K$, el conjunto $I\!K^n$ es un espacio vectorial, tomando como suma de vectores la suma coordenada a coordenada, y el producto por un escalar tambien coordenada a coordenada.
- Y $\{0, 1\}$ es un cuerpo, con la suma y el producto modulo 2.
- Asi que $\{0, 1\}^n$ tiene una estructura natural de espacio vectorial, y es respecto de esa estructura que estamos hablando de "subespacio vectorial" .
- Ahora bien, como estamos trabajando sobre $\{0, 1\}$ la propiedad de ser subespacio vectorial puede simplificarse respecto de la usual dada en álgebra lineal.

Repaso de subespacios vectoriales

- Recordemos que W era subespacio vectorial de un espacio vectorial V si:
  1. $W \neq \emptyset$
  2. $u, v \in W \Rightarrow u + v \in W$.
  3. $u \in W,\ c \in I\!K \Rightarrow c.u \in W$
- Nota: en algunos textos la segunda y tercera propiedad a veces se juntan en una sola:
  - $u, v \in W,\ c \in I\!K \Rightarrow c.u + v \in W$.
- En el caso del cuerpo $\{0, 1\}$ se puede simplificar a pedir simplemente 1 y 2.
- Veamos esto.

Subespacios vectoriales en el caso $I\!K = \{0, 1\}$

- En $\{0, 1\}$, la suma cumple $x + x = 0$.
- Por lo tanto para todo $v \in \{0, 1\}^n$ tenemos $v + v = \vec{0}$, donde $\vec{0}$ es el vector con todas las coordenadas 0, pues la suma de $\{0, 1\}^n$ es la suma coordenada a coordenada.
- Supongamos que $W \subseteq \{0, 1\}^n$ satiface sólo 1 y 2.
- Entonces como vale 1, existe algún $v \in W$.
- Como vale 2, entonces $v + v \in W$.
- Pero $v + v = \vec{0}$, asi que deducimos que $\vec{0} \in W$.

Subespacios vectoriales en el caso $I\!K = \{0, 1\}$

- Sea ahora $u \in W$, $c \in \{0, 1\}$.
  - Si $c = 1$, entonces $c.u = 1.u = u \in W$.
  - Si $c = 0$, entonces $c.u = 0.u = \vec{0} \in W$.
- Asi que en cualquier caso, $c.u \in W$ y vale la propiedad 3.
- Conclusión: un código C es lineal sii es un subconjunto no vacio de $\{0, 1\}^n$ invariante por la suma. (es decir, que $u, v \in C \Rightarrow u + v \in C$.)

Ejemplos

- De los ejemplos que dimos en la primera parte de códigos, todos eran lineales menos $C3$.
- $C3$ era igual a $\{000111, 101010, 110001, 011100\}$.
- Recien probamos que $\vec{0}$ debe estar en C si C es lineal.
- Asi que $C3$ no es lineal.
- Aunque le agregaramos 000000 seguiria sin ser lineal pues pej $000111 + 101010 = 101101$ no está en $C3$.
- Es facil ver que los otros C_i son lineales pues son todos de la forma $\{\vec{0}, u, v, u + v\}$ y como $u + (u + v) = v$ y $v + (u + v) = u$, es claro que la suma de dos elementos cualesquiera del código queda dentro del código.

Ejemplos

- Habiamos mencionado que $C4$ tenia una ventaja sobre $C3$ a pesar de tener la misma longitud, corregir la misma cantidad de errores pero detectar uno menos.
- Justamente la ventaja es que $C4$ es lineal y $C3$ no.
- Hay varias razones por las cuales se prefieren los códigos lineales a los no lineales, y por qué son los mas usados.
- Veamos algunas, empezando por el hecho que es mas fácil calcular δ en códigos lineales.

Peso de Hamming

Definición

Dada una palabra v de un código, el peso de Hamming de v es $|v| = d_H(v, 0)$, es decir, es el número de unos que tiene v.

Por ejemplo $|1001001100101001000| = 7$

Observación:

$d_H(x, y) = |x + y|$

Pues $d_H(x, y) =$número de bits de diferencia entre x e y=(número de 1s en $x + y$)=$|x + y|$.

δ en códigos lineales

Lema

Si C es lineal, entonces $\delta(C) = \text{Min}\{|v| : v \in C, v \neq 0\}$

Observemos que este lema dice que, en vez de tener que calcular δ en forma cuadrática en el número de palabras (pues deberiamos hacer $d_H(x, y)$ para cada par de palabras), lo podemos calcular en tiempo lineal en el número de palabras, si el código es lineal.

Prueba del lema

- Sea $m =$Min$\{|v| : v \in C, v \neq 0\}$.
- Sean $x, y \in C, x \neq y : d_H(x, y) = \delta$.
- Entonces, $\delta = |x + y|$.
- Pero como C es lineal, $x + y \in C$.
- Y como $x \neq y$, entonces $x + y \neq 0$.
- Por lo tanto $\delta = |x + y| \geq m$ pues m es el mínimo de los $|v|$ con $v \in C$, $v \neq 0$.

Prueba del lema

- Para probar la desigualdad para el otro lado, sea ahora $v \in C$ con $v \neq 0$ tal que $m = |v|$.
- Como C es lineal, entonces $0 \in C$.
- Por lo tanto $d_H(v, 0)$ es una distancia entre dos palabras de C.
- Como $v \neq 0$, esas dos palabras son distintas.
- Entonces $d_H(v, 0) \geq \delta$ por definición de δ.
- Asi, $m = |v| = d_H(v, 0) \geq \delta$. Fin prueba lema.
- Antes de seguir con otras ventajas, recordemos el concepto de dimensión de un espacio vectorial.

Dimensión de un código lineal

- La dimensión de un espacio vectorial es la cardinalidad de cualquier base del espacio. (eso lo probaron en álgebra lineal: dos bases cualesquiera tienen la misma cardinalidad).
- Y ¿qué era "base" ?
- Una base de un espacio vectorial es un conjunto
  1. generador y:
  2. LI (linealmente independiente)
- En otras palabras, para el caso finito que es el único que veremos, $\{u_1, ..., u_k\}$ es base de V si:
  1. Genera V: $u \in V \Rightarrow \exists c_1, ..., c_k : u = c_1 u_1 + \cdots + c_k u_k$.
  2. Es LI: $c_1 u_1 + \cdots + c_k u_k = 0 \Rightarrow c_1 = \cdots = c_k = 0$.

Dimensión de un código lineal

- La dimensión de un código lineal se suele denotar con la letra k.
- Un código lineal con dimensión k, longitud n y $\delta(C) = \delta$ se suele denotar como un código (n, k, δ)
- Un teorema elemental de álgebra lineal, que usaremos, es que si k es la dimensión de un código y $\mathcal{B}$ tiene k elementos, entonces $\mathcal{B}$ es base de C si y solo si $\mathcal{B}$ genera C si y solo si $\mathcal{B}$ es LI.
- Es decir, para probar que algo es base de un código de dimensión k, podemos ver que genera y es LI, o que tiene k elementos y genera, o que tiene k elementos y es LI.

Cantidad de elementos de un código lineal

- Si k es la dimensión de C, ¿cuantos elementos tiene C?.
- Sea $\{u_1, ..., u_k\}$ una base de C.
- Entonces como genera, cada elemento u de C se puede representar como $u = c_1 u_1 + \cdots + c_k u_k$ para ciertos c_i en $\{0, 1\}$.
- Pero como es LI, esa representación es única.
- Asi que la cantidad de elementos de C es la misma que la de el conjunto de k-uplas $(c_1, ...c_k) \in \{0, 1\}^k$, es decir 2^k
- Por ejemplo, no hay códigos lineales con 6 elementos, pues 6 no es potencia de 2.

- Como un código lineal de dimensión k tiene exactamente 2^k palabras podemos decir que la dimensión es el logaritmo en base 2 del número de palabras.
- La dimensión es importante porque nos esta diciendo cuantos de los bits del código son “bits de información”
- Y el resto, $n - k$, son los bits que tenemos que agregar para poder corregir la cantidad de errores que querramos corregir.
- En los ejemplos que vimos la clases pasada la información que queriamos mandar eran dos bits.
- El código C1, que no corregia ni detectaba, mandaba exactamente dos bits pero los otros códigos tenian que mandar mas bits.

Codificación

- En el caso general de un código lineal, las palabras a codificar serán las palabras de $\{0, 1\}^k$, donde k es su dimensión pero las palabras del código estarán en $\{0, 1\}^n$ para algún n.
- En realidad eso ocurre con cualquier código: se tiene una serie de palabras P que se quieren mandar, y que se mandarian si no hubiera posibilidad de errores.
- Pero como tenemos posibilidad de errores, en vez de mandar las palabras de P, se mandan las palabras de un código C que puede corregir errores.
- En el caso de un código lineal, sabemos que P tiene cardinalidad exactamente 2^k, y podemos asumir que es $P = \{0, 1\}^k$

Codificación y Decodificación

- En cualquier caso, lineal o no, el transmisor necesita algo que transforme cada palabra de P en una de C, para poder mandarla.
- Eso es la “codificación".
- Y tambien, el receptor necesita, luego de haber corregido los errores, poder transformar la palabra de C que le queda en la palabra de P correspondiente.
- Eso es la “decodificación".

Codificación/Decodificación

- Siempre podemos simplemente hacer una tabla arbitraria de correspondencias entre P y C.
- Pero entonces hay que guardar toda la tabla.
- Pero los códigos lineales tienen un algoritmo eficiente que, dada una palabra de P calcula la palabra de C que le corresponde, y viceversa.
- Y no hace falta guardar toda la tabla de correspondencia.
- Ni siquiera hace falta guardar C: las palabras se generan a medida que se las necesita.
- ¿Cómo? Con transformaciones lineales.

Transformaciones lineales

- Recordemos que una transformación lineal $T : V_1 \mapsto V_2$ entre espacios vectoriales es una función tal que $u, v \in V_1, c \in I\!K \Rightarrow T(c.u + v) = c.T(u) + T(v)$
- La imagen de T es simplemente la imagen como función $Im(T) = \{v \in V_2 : \exists u \in V_1 : T(u) = v\}$
- Es fácil ver (y lo deben haber hecho en álgebra) que si $T : V_1 \mapsto V_2$ es lineal, entonces la imagen de T es un subespacio vectorial de V_2
- Asi que esto nos permite construir códigos lineales usando transformaciones lineales.

- Y por eso son útiles los códigos lineales: no hace falta guardar todo el código, sino sólo la transformación lineal T para poder ir generando las palabras del código a medida que las necesitemos.
- En gral, dado que todo es finito, T se implementa como una multiplicación por una matriz, y sólo hay que guardar la matriz.
- Y si T es de la forma especial $T : \{0, 1\}^k \mapsto \{0, 1\}^n : x \mapsto (x, L(x))$ o $T : \{0, 1\}^k \mapsto \{0, 1\}^n : x \mapsto (L(x), x)$ entonces ni siquiera hay que guardar toda la matriz sino la parte correspondiente a L.

Matrices

- Las transformaciones lineales que usaremos serán “simplemente" multiplicación de un vector por una matriz
- (si avanzaron lo suficiente en álgebra, quizás recuerden que toda transformación lineal entre dos espacios de dimensión finita se puede representar como una multiplicación por una matriz).
- Aca tenemos que ponernos de acuerdo en cómo representamos a los vectores de $\{0, 1\}^n$.
- Los podemos representar en forma “horizontal" o “vertical".
- Es decir, pej para $n = 4$, tipo (a, b, c, d) o tipo $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

Matrices

- Nosotros usaremos la representación horizontal, que es la mas común en teoría de códigos.
- Y usaremos la trasposición del vector cuando necesitemos usarlo en forma vertical:
- $(a, b, c, d)^t = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$
- Pero hay otros libros que representan a los vectores verticalmente, asi que tengan cuidado.

Matrices

- Con esa representación, entonces podemos usar una transformación lineal $T : \{0, 1\}^k \mapsto \{0, 1\}^n$ de la forma $T(u) = u.G$, donde G es una matriz $k \times n$, es decir, k filas y n columnas.
- Entonces si queremos mandar el mensaje $u, \hat{u}, u^*$.... etc, formado por palabras de $\{0, 1\}^k$ que mandariamos si no hubiera errores en el canal, en vez de eso mandamos $u.G, \hat{u}.G, u^*.G, ...,$ etc.
- Asi que codificar palabras es bastante fácil en códigos lineales.
- La matriz G como "genera" las palabras del código, se llama matriz generadora

Matriz generadora de un código lineal

- Es decir, G es matriz generadora de C si C es la imagen de la transformación lineal $T : \{0, 1\}^k \mapsto \{0, 1\}^n$ dada por $T(u) = u.G$, donde $k = dimC$.
- Podemos abreviar esto diciendo que G es generadora de C si $C = Im(G)$ y la dimensión de C es igual al número de filas de G.
- El requerimiento de que el número de filas de G sea igual a la dimensión de C es crucial pues si no podria pasar que dos palabras distintas sean codificadas igual, y luego no podriamos decodificar.

Matriz generadora de un código lineal

- Es decir, queremos que $C = Im(G)$ pero que ademas $T(u) = u.G$ sea inyectiva.
- (Esto para transformaciones lineales es lo mismo que decir que $Nu(T) = \{0\}$, donde recordemos que $Nu(T)$ era el nucleo de T: $\{u : T(u) = 0\}$)
- De todos modos, si la cantidad de filas de G es igual a la dimensión de C, y $C = Im(G)$, esto se satisface automáticamente.
- Es fácil ver que G es una matriz generadora de un código lineal C si y solo si las filas de G forman una base de C:

Matriz generadora, continuación.

- Pues $C = Im(G)$ es decir que $v \in C \iff \exists u = (u_1, ..., u_k)$ tal que $v = uG$.
- Y eso es si y solo si $\forall v \in C \exists u_1, ..., u_k : v = u_1 G_1 + \cdots + u_k G_k$, donde G_i es la fila i-ésima de G.
- Asi que G es generadora del código si y solo si sus filas generan el código.
- Como son k filas, y $k =$dim$C$, entonces las filas generan el código si y solo si son base.
- Observemos que entonces queda claro que no hay una única matriz generadora: cualquier matriz cuyas filas formen base de C es generadora.

Matriz generadora, continuación.

- Si G es generadora de C que tiene longitud n y dimensión k, entonces G debe ser $k \times n$: k filas y n columnas.
- Viceversa, dada una matriz G que sea $k \times n$ y cuyas filas sean LI, podemos simplemente definir C como el espacio generado por las filas de G.
- C será automáticamente un código lineal con dimensión k y longitud n.

Decodificación

- Como enfatizamos antes, el hecho que toda palabra v de C sea de la forma $v = uG$ permite codificar fácilmente.
- Pero ¿que hay acerca de la decodificación?
- Luego veremos como se corrijen los errores, pero el receptor recibirá palabras $w, \hat{w}, w^*$, etc a las cuales le corregirá los errores para poder obtener $v, \hat{v}, v^*, \ldots$ etc. todos en C, y debe resolver los sistemas lineales $u.G = v$, $\hat{u}.G = \hat{v}$, $u^*.G = v^*$ para poder recuperar $u, \hat{u}, u^*$ etc.
- La dificultad de resolver el sistema depende de G pero si G "tiene la identidad" a izquierda o derecha, es trivial.

Decodificación

- Es decir, supongamos que G es de la forma $[I_k|A]$, donde I_k es la identidad $k \times k$.
- Enconces $u.G = (u, u.A)$
- Entonces para recuperar u a partir de $u.G$, solo hay que mirar los primeros k bits.
- Algo similar, con los ultimos bits, si G es de la forma $[A|I_k]$.
- Tomar G una matriz de la forma $[I_k|A]$ o $[A|I_k]$ ademas asegura que las filas son realmente LI, sin tener que hacer los cálculos.
- Algunas generadoras tienen la identidad “distribuida" entre las columnas, en vez de toda a la izquierda o toda a la derecha, lo cual no parece tener sentido pero luego veremos un caso donde si lo tiene.

Ejemplo

- Sea
$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$
- El código C del cual G es matriz generadora tiene entonces dimensión 3 y longitud 5.
- Como tiene dimensión 3 entonces tiene $2^3 = 8$ palabras.
- Como son pocas, podemos calcularlas a todas.
- Simplemente tomando todos los $u = (u_1, u_2, u_3) \in \{0,1\}^3$ y haciendole uG a cada uno.

Ejemplo

- Una ventaja de trabajar en $\{0, 1\}$ es que multiplicar matrices o vectores por matrices, es mucho mas fácil que lo que hicieron en álgebra.
- Pej, $(1, 0, 0)G$ simplemente nos da la primera fila de G.
- Y $(1, 0, 1)G$ nos da la suma de las primera con la tercera fila de G.
- Recordemos que “suma" es módulo 2, asi que $1 + 1 = 0$.
- Entonces podemos calcular todo C rapidamente:

Ejemplo

$$
\begin{array}{ccccccccc}
 & u & & & & & uG & & \\
0 & 0 & 0 & -> & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -> & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & -> & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & -> & 1 & 1 & 0 & 0 & 1
\end{array}
$$

Pues $G_1 + G_2 = (1, 0, 0, 1, 0) + (0, 1, 0, 1, 1) = (1, 1, 0, 0, 1)$

Ejemplo

$$\begin{array}{ccccccccc} & u & & & & & uG & & \\ 0 & 0 & 0 & -> & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -> & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -> & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -> & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -> & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & -> & 1 & 0 & 1 & 1 & 1 \end{array}$$

Pues $G_1 + G_3 = (1, 0, 0, 1, 0) + (0, 0, 1, 0, 1) = (1, 0, 1, 1, 1)$

Ejemplo

$$
\begin{array}{ccccccccc}
 & u & & & & & uG & & \\
0 & 0 & 0 & -> & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -> & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & -> & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & -> & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & -> & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & -> & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & -> & 0 & 1 & 1 & 1 & 0
\end{array}
$$

Pues $G_2 + G_3 = (0, 1, 0, 1, 1) + (0, 0, 1, 0, 1) = (0, 1, 1, 1, 0)$

Ejemplo

$$
\begin{array}{ccccccccc}
 & u & & & & & uG & & \\
0 & 0 & 0 & -> & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -> & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & -> & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & -> & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & -> & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & -> & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & -> & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & -> & 1 & 1 & 1 & 0 & 0
\end{array}
$$

Pues $G_1 + G_2 + G_3 =$
$(1, 0, 0, 1, 0) + (0, 1, 0, 1, 1) + (0, 0, 1, 0, 1) = (1, 1, 1, 0, 0)$

Ejemplo

- En este ejemplo, construimos todas las palabras, e hicimos explícitamente la tabla de correspondencia para, justamente, ejemplificar.
- Pero aún sin la tabla, decodificar es trivial: basta leer los primeros 3 bits:
- Si nos llega pej, 01110 sabemos que está codificando la palabra 011.
- Veamos un ejemplo al revés

Otro ejemplo

Sea

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

En este caso $k = 7$, $n = 9$, y C tiene $2^7 = 128$ palabras.
La identidad esta a la derecha en este caso

Continuación del otro ejemplo

- Si les damos una G tan grande no le vamos a pedir que nos den todas las palabras del código.
- Pero si que calculen, dada una palabra que queremos enviar, cual es la palabra que realmente debemos enviar.
- Pej, si queremos mandar $u = 1001101$
- Entonces debemos mandar $v = uG =$ suma de las filas 1,4,5 y 7 de G.
- $v = 011000000 + 110001000 + 110000100 + 110000001 = 101001101$

Continuación del otro ejemplo

- Si les damos una G tan grande no le vamos a pedir que nos den todas las palabras del código.
- Pero si que calculen, dada una palabra que queremos enviar, cual es la palabra que realmente debemos enviar.
- Pej, si queremos mandar $u = 1001101$
- Entonces debemos mandar $v = uG =$ suma de las filas 1,4,5 y 7 de G.
- $v = 011000000 + 110001000 + 110000100 + 110000001 = 101001101$
- Observemos que ocurre lo que sabiamos que deberia ocurrir: 101001101 “tiene" al u a la derecha.
- Que corresponde con que G tiene la identidad a la derecha.

Continuación del otro ejemplo

- Asi que podriamos no haber calculado esos bits y simplemente haber calculado los 2 primeros bits mirando los 2 primeros bits de las filas 1,4,5,7 y agregarlos a u
- : $01 + 11 + 11 + 11 = 10 \mapsto 101001101$.
- Pej, si queremos mandar $u = 0101001$.
- Miramos los 2 primeros bits de las filas 2,4,7: 11+11+11=11.
- y mandamos u con esos 2 bits agregados 110101001.
- O bien para $u = 0001100$ sumamos los primeros 2 bits de las filas $4, 5$: $11 + 11 = 00$ y obtenemos $uG = 000001100$.

Continuación del otro ejemplo

- Asi que codificar es fácil.
- Decodificar es mas fácil: supongamos que luego de corregir el receptor obtiene la palabra 011011101
- Entonces sabe que la información que le querian mandar eran los últimos 7 bits: 1011101.

Ventajas de los códigos lineales

- En conclusión, codificar en un código lineal es simplemente multiplicar por una matriz, y no hace falta guardar todas las palabras del código sino sólo la matriz.
- Ademas, si la matriz es de una cierta forma, decodificar es tan trivial como leer algunos de los bits del resultado.
- Asi que los códigos lineales son eficientes en cuanto a la memoria utilizada y tambien tienen algoritmos eficientes de codificación y decodificación.
- Estas no son sus única ventajas.

Ventajas de los códigos lineales

- Ademas habiamos visto que es mas fácil calcular δ que en códigos no lineales, y la clase que viene hablaremos mas sobre esto.
- Tambien veremos que para ciertos requerimientos, es fácil construir códigos lineales en forma adecuada a esos requerimientos.
- Para el caso de corrección de 1 error, los códigos lineales tienen un excelente algoritmo de corrección, que veremos la proxima clase.