Aplicações: Teoria Algébrica dos Códigos

Consideremos o seguinte código binário, a que chamaremos $\mathcal{C}_1$, que permite dar as instruções de comando a um leitor de DVD, através de um comando à distância:

PLAY	REW	FORWARD	STOP
00	01	10	11

Suponhamos que carregamos na tecla PLAY do comando, a que corresponde a palavra 00 do código; o comando transmite esta palavra ao leitor de DVD mas se, porventura, nessa comunicação ocorrer o erro

$$\mathbf{0}0 \overset{erro}{\dashrightarrow} \mathbf{1}0$$

o leitor receberá a palavra 10, e como esta faz parte de $\mathcal{C}_1$ (corresponde à instrução FORWARD), aquele não terá nenhuma maneira de detectar o erro e executará a instrução FORWARD!

O código $\mathcal{C}_1$ é um exemplo de *código binário*, ou seja, um código definido sobre o alfabeto (corpo) $\mathbb{F}_2$, constituído por todas as palavras de comprimento 2 nesse alfabeto. Trata-se de um código muito pobre, pois nem sequer detecta erros *simples* (*singulares*) como o do exemplo acima.

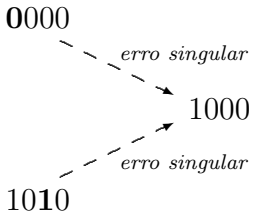
O que fazemos habitualmente quando não entendemos o que outra pessoa nos quer dizer? Pedimos que repita a mensagem. Façamos isso no código $\mathcal{C}_1$, isto é, pensemos no código $\mathcal{C}_2$ que se obtem de $\mathcal{C}_1$ repetindo a informação em cada palavra uma vez:

PLAY	REW	FORWARD	STOP
0000	0101	1010	1111

Agora, ao ser transmitida a instrução PLAY (ou seja, a palavra 0000), se ocorrer o mesmo erro singular de há pouco,

$$\mathbf{0}000 \overset{erro}{\dashrightarrow} \mathbf{1}000$$

como a palavra recebida não faz parte de $\mathcal{C}_2$, o leitor de DVD pode concluir imediatamente que ocorreu algum erro na transmissão. Neste caso, o código $\mathcal{C}_2$ já detecta este erro singular (e é fácil de ver que detecta qualquer outro erro singular). Terá maneira de corrigir esse erro, isto é, de identificar a palavra original (*assumindo que na transmissão só poderão ocorrer, quando muito, erros singulares*)? Não; de facto, há duas palavras em $\mathcal{C}_2$ que poderiam ser as originais:



Consideremos, finalmente, o código $\mathcal{C}_3$, definido pela tabela

PLAY	REW	FORWARD	STOP
000000	010101	101010	111111

Agora, além de qualquer erro singular ser detectável, também pode ser corrigido automaticamente (assumindo novamente que *na transmissão só poderão ocorrer, quando muito, erros singulares*). Por exemplo, o erro singular

$$\mathbf{0}00000 \overset{erro}{\dashrightarrow} \mathbf{1}00000$$

é evidentemente detectado e corrigido; a única palavra de $\mathcal{C}_3$ que poderia ter dado origem à palavra 100000, *na assumpção que só ocorreram erros singulares*, é a palavra 000000:

Palavra de $\mathcal{C}_3$	000000	010101	101010	111111
Palavra recebida	**1**00000	**10**000	10**0**0**0**	1**00000**
Número de erros	1	4	2	5

É claro que se puderem ocorrer erros duplos no canal de comunicação, $\mathcal{C}_3$ já não corrige o erro singular acima: a palavra original poderia muito bem ser a palavra 101010.

Assim, esta ideia de construir códigos correctores de erros só funciona se conhecermos *a priori* um limite para o número de erros que pode ocorrer no respectivo canal de comunicação. Ou, então, se adoptarmos o seguinte princípio de bom senso (o chamado *princípio do vizinho mais próximo*):

> *A palavra original correspondente a uma palavra recebida com erros deve ser a palavra do código "mais próxima" da palavra recebida*
>
> (isto é, assumimos que é mais provável que o menor número de erros possível tenha ocorrido na transmissão).

Daqui em diante, assumimos sempre este princípio. (mais adiante, tornaremos precisa a noção de proximidade implícita no termo "mais próxima".)

Os códigos $\mathcal{C}_1$, $\mathcal{C}_2$ e $\mathcal{C}_3$ são exemplos do tipo de códigos que vamos estudar, e que podem ser formalizados do seguinte modo:

CÓDIGOS SOBRE UM CORPO FINITO $\mathbb{F}_q$. CÓDIGOS LINEARES

Um *código de comprimento n sobre* o corpo $\mathbb{F}_q$ é um subconjunto $\mathcal{C}$ de $(\mathbb{F}_q)^n$. Portanto, $\mathcal{C}$ é formado por palavras de comprimento n, $a_1a_2\ldots a_n$, formadas com o alfabeto $\mathbb{F}_q$ (isto é, cada $a_i \in \mathbb{F}_q$).
Note que $\mathbb{F}_q^n$ é um espaço vectorial sobre $\mathbb{F}_q$, de dimensão n. Assim, as palavras de $\mathcal{C}$ são simplesmente vectores deste espaço. Quando $\mathcal{C}$ é um subespaço linear de $\mathbb{F}_q^n$, de dimensão k, diz-se que $\mathcal{C}$ é um *código linear* ou (n,k)-*código* sobre $\mathbb{F}_q$.

Exemplos: $\mathcal{C}_1 = \mathbb{F}_2^2$, pelo que $\mathcal{C}_1$ é um $(2,2)$-código sobre $\mathbb{F}_2$. Os códigos $\mathcal{C}_2$ e $\mathcal{C}_3$ também são códigos lineares sobre $\mathbb{F}_2$ (binários), como é fácil de ver: $\mathcal{C}_2$ é um $(4,2)$-código enquanto $\mathcal{C}_3$ é um $(6,2)$-código.

Os (n,k)-códigos sobre o corpo $\mathbb{F}_2$ foram o tipo de códigos utilizados pelas sondas que viajaram até Marte, na transmissão das fotografias para a Terra. No caso dos CDs de música, utiliza-se o corpo $\mathbb{F}_{256} = \mathbb{F}_{2^8}$.

Precisemos agora a noção de distância entre duas palavras de $\mathbb{F}_q^n$.

DISTÂNCIA DE HAMMING

A *distância de Hamming* entre duas palavras $\vec{a} = a_1a_2\ldots a_n$ e $\vec{b} = b_1b_2\ldots b_n$ é o número de índices $i \in \{1,2,\ldots,n\}$ tais que $a_i \neq b_i$.
Note que $d(\vec{a},\vec{b})$ indica o número de erros ocorridos se $\vec{a}$ é a palavra transmitida e $\vec{b}$ é a palavra recebida.

Por exemplo, $d(1101,0111) = 2$.
É muito fácil de ver que a distância de Hamming é uma métrica em $\mathbb{F}_q^n$, isto é, para quaisquer $\vec{a},\vec{b},\vec{c} \in \mathbb{F}_q^n$, tem-se:

(1) $d(\vec{a},\vec{b}) \geq 0$; $d(\vec{a},\vec{b}) = 0$ se e só se $\vec{a} = \vec{b}$.

(2) $d(\vec{a},\vec{b}) = d(\vec{b},\vec{a})$.

(3) $d(\vec{a},\vec{b}) \leq d(\vec{a},\vec{c}) + d(\vec{c},\vec{b})$.

DISTÂNCIA MÍNIMA

Chama-se *distância mínima* de um código $\mathcal{C}$, que se denota por $\delta(\mathcal{C})$, ao número

$$\min_{\vec{a},\vec{b}\in\mathcal{C},\vec{a}\neq\vec{b}} d(\vec{a},\vec{b}).$$

Este número mede o grau de vizinhança das palavras em $\mathcal{C}$. Por exemplo, $\delta(\mathcal{C}_1) = 1$, $\delta(\mathcal{C}_2) = 2$ e $\delta(\mathcal{C}_3) = 3$.

Quanto maior é o valor de $\delta(\mathcal{C})$, mais eficiente é o código. Portanto, um dos objectivos na construção de um código é que tenha as palavras o mais afastadas entre si. Por outro lado, isto limita o número de palavras do código, logo limita a sua capacidade de armazenar e transmitir informação. Reconciliar estes dois objectivos (isto é, procurar o ponto de equilíbrio entre eles) é um dos problemas da teoria dos códigos.

CÓDIGOS t-DETECTORES E t-CORRECTORES DE ERROS

Seja $t \in \mathbb{N}$. Diz-se que um código $\mathcal{C}$ é *t-detector de erros* se detecta qualquer combinação de t erros em qualquer palavra.
Diz-se que $\mathcal{C}$ é *t-corrector de erros* se corrige qualquer combinação de t erros em qualquer palavra.

Teorema. *Seja $\mathcal{C}$ um código com distância mínima $\delta(\mathcal{C})$.*

(a) *Se $t \leq \delta(\mathcal{C}) - 1$, então $\mathcal{C}$ é t-detector de erros.*

(b) *Se $t \leq \frac{\delta(\mathcal{C})-1}{2}$, então $\mathcal{C}$ é t-corrector de erros.*

Demonstração. (a) Suponhamos que na transmissão de uma palavra $\vec{a} \in \mathcal{C}$ ocorreram t erros, resultando na palavra recebida $\vec{b}$:

$$\vec{a} \quad \overset{\underset{erros}{t}}{\dashrightarrow} \quad \vec{b}$$

(portanto, $d(\vec{a},\vec{b}) = t$). Para provarmos que o código terá a capacidade de detectar o erro, teremos que garantir que $\vec{b} \notin \mathcal{C}$, o que é fácil: como $d(\vec{a},\vec{b}) = t < \delta(\mathcal{C})$ e $\vec{a} \in \mathcal{C}$ então $\vec{b} \notin \mathcal{C}$.

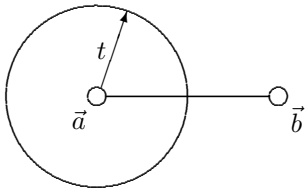
(b) Suponhamos que na transmissão de uma palavra $\vec{a} \in \mathcal{C}$ ocorreram t erros, resultando na palavra recebida $\vec{b}$ (portanto, $d(\vec{a},\vec{b}) = t$). Agora, para provarmos que

o código terá a capacidade de corrigir o erro, bastará garantir que mais nenhuma palavra em $\mathcal{C}$ além de $\vec{a}$ pode ter dado origem à palavra errada $\vec{b}$, ou seja, que qualquer outra palavra $\vec{c} \in \mathcal{C}$ está a uma distância de $\vec{b}$ maior do que t, o que também é fácil: pela desigualdade triangular da distância,

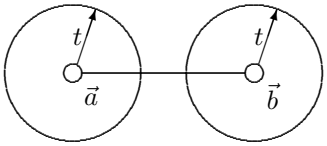
$$d(\vec{b}, \vec{c}) \geq d(\vec{a}, \vec{c}) - d(\vec{a}, \vec{b}) \geq \delta(\mathcal{C}) - t \geq 2t + 1 - t = t + 1.$$

■

Portanto, um código consegue detectar t erros se quaisquer duas palavras do código estiverem a uma distância de Hamming pelo menos $t + 1$:



Por sua vez, um código consegue corrigir t erros se quaisquer duas palavras do código estiverem a uma distância de Hamming pelo menos $2t + 1$:



Nos exemplos que vimos anteriormente, tem-se:

Código	$\delta(\mathcal{C})$	No. erros que detecta	No. erros que corrige
$\mathcal{C}_1$	1	0	0
$\mathcal{C}_2$	2	1	0
$\mathcal{C}_3$	3	2	1

Portanto $\mathcal{C}_2$ é 1-detector de erros e $\mathcal{C}_3$ é 1-corrector de erros e 2-detector de erros.

A definição de código t-corrector implica que quaisquer bolas de raio t, centradas em palavras distintas, sejam disjuntas. Se, além disso, estas bolas cobrirem a totalidade do espaço (uma propriedade rara mas interessante), o código diz-se *perfeito*. Assim, um código t-corrector $\mathcal{C}$ sobre $\mathbb{F}_q$ diz-se *perfeito* se

$$\bigcup_{\vec{a} \in \mathcal{C}} B(\vec{a}, t) = \mathbb{F}_q^n.$$