BÀI 11 MÃ KHỐI TUYẾN TÍNH

11.1 Giới thiệu

Mã khối tuyến tính là một lớp mã được dùng rất phổ biến trong việc chống nhiễu. Loại mã này được xây dựng dựa trên các kết quả của đại số tuyến tính. Ở đây chúng ta cũng chỉ nghiên cứu về mã nhị phân.

Định nghĩa

Một mã khối có chiều dài n gồm 2^k từ mã được gọi là mã tuyến tính $C(n, k)$ nếu và chỉ nếu 2^k từ mã hình thành một không gian vectơ con k chiều của không gian vectơ n chiều gồm tất cả các vectơ n thành phần trên trường GF(2).

Trường GF(2) (Galois Field (2)) là trường nhị phân đồng thời phép cộng là phép cộng modul 2 (kí hiệu là $\oplus$), còn phép nhân là phép và (AND). Cụ thể

$0 \oplus 0 = 0$	$0 \oplus 1 = 1$	$1 \oplus 0 = 0$	$1 \oplus 1 = 0$
$0 \,.\, 0 = 0$	$0 \,.\, 1 = 0$	$1 \,.\, 0 = 0$	$1 \,.\, 1 = 1$

Mã tuyến tính $C(n, k)$ có mục đích mã hoá những khối tin (hay thông báo) k bit thành những từ mã n bit. Hay nói cách khác trong n bit của từ mã có chứa k bit thông tin. Các phần tiếp theo sau sẽ trình bày cách biểu diễn mã, cách mã hoá các thông báo thành từ mã, cách giải mã từ từ mã thành thông báo, cách phát hiện sai và sửa sai.

Qui ước

Để đơn giản sau này chúng ta sẽ viết dấu + thay cho dấu $\oplus$ và dấu + sẽ được hiểu theo ngữ cảnh.

11.2 Các khái niệm và nguyên lý hoạt động

Cách biểu diễn mã – Ma trận sinh

Mã tuyến tính $C(n, k)$ là một không gian con k chiều của một không gian vectơ n thành phần. Do vậy có thể tìm được k từ mã độc lập tuyến tính trong C chẳng hạn $(g_0, g_1, ..., g_{k-1})$ sao cho mỗi từ mã trong C là một tổ hợp tuyến tính của k từ mã này:

$$v = a_0 g_0 + a_1 g_1 + ... + a_{k-1} g_{k-1}$$

với $a_i \in \{0, 1\}$ với mọi $i = 0, 1, ..., k-1$.

Đặt k từ mã độc lập tuyến tính này thành những hàng chúng ta có được một ma trận cấp $k \times n$ như sau

$$G_{k \times n} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0(n-1)} \\ g_{10} & g_{11} & \cdots & g_{1(n-1)} \\ \vdots & \vdots & & \vdots \\ g_{(k-1)0} & g_{(k-1)1} & \cdots & g_{(k-1)(n-1)} \end{bmatrix}$$

Với $g_i = (g_{i0}, g_{i1}, \ldots, g_{i(n-1)})$, với $i = 0, 1, \ldots, k-1$.

Cách mã hoá

Nếu $u = (a_0, a_1, \ldots, a_{k-1})$ là thông tin cần được mã hoá thì từ mã v tương ứng với u được ta bằng cách lấy u nhân với G.

$$v = u \times G = (a_0, a_1, \ldots, a_{k-1}) \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

hay

$$v = a_0 g_0 + a_1 g_1 + \ldots + a_{k-1} g_{k-1}$$

Vì các từ mã tương ứng với các thông báo được sinh ra bởi G theo cách như trên nên G được gọi là ma trận sinh (generating matrix) của bộ mã.

Ví dụ 11.1

Cho ma trận sinh của một mã tuyến tính (7, 4) sau

$$G_{4\times 7} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Nếu $u = (1101)$ là thông tin cần mã hoá thì từ mã tương ứng sẽ là

$$v = 1.g_0 + 1.g_1 + 0.g_2 + 1.g_3$$
$$= (1100101)$$

Chú ý

1. Bất kỳ k từ mã độc lập tuyến tính nào cũng có thể được dùng để làm ma trận sinh cho bộ mã. Hay nói cách khác các ma trận sinh khác nhau có thể biểu diễn cùng một bộ mã tuyến tính (hay còn gọi là không gian mã) như nhau, hay ngược lại một bộ mã tuyến tính có thể có nhiều ma trận sinh khác nhau biểu diễn.
2. Tương ứng với mỗi ma trận sinh chúng ta có một phép mã hoá. Có nghĩa là ứng với hai ma trận sinh khác nhau chúng ta có hai phép mã hoá khác nhau. Vì vậy với cùng một bộ mã tuyến tính việc chọn ma trận sinh nào là rất quan trọng vì nó quyết định việc ánh xạ thông báo nào thành từ mã nào.

Cách giải mã

Ở đây chúng ta sẽ trình bày cách giải mã từ từ mã về thông tin ban đầu. Lấy ma trận sinh như ở trong Ví dụ 11.1. Chúng ta gọi thông báo là $u = (a_0, a_1, a_2, a_3)$ và từ mã tương ứng là $v = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$. Chúng ta có hệ phương trình sau liên hệ giữa u và v.

$$v = u \times G$$

Suy ra

$$b_0 = a_0 + a_1 + a_3 \quad (1)$$
$$b_1 = a_0 + a_2 \quad (2)$$
$$b_2 = a_1 + a_3 \quad (3)$$
$$b_3 = a_0 + a_1 \quad (4)$$
$$b_4 = a_1 \quad (5)$$
$$b_5 = a_2 \quad (6)$$
$$b_6 = a_2 + a_3 \quad (7)$$

Để giải các a_i theo các b_j chúng ta chỉ cần chọn bốn phương trình đơn giản nhất để giải. Chẳng hạn chúng ta chọn các phương trình (4), (5), (6), (7) chúng ta sẽ giải được

$$a_0 = b_3 + b_4$$
$$a_1 = b_4$$
$$a_2 = b_5$$
$$a_3 = b_5 + b_6$$

Hệ phương trình trên được gọi là hệ phương trình giải mã. Dĩ nhiên ứng với các ma trận sinh khác nhau cho dù biểu diễn cùng một bộ mã chúng ta sẽ có các hệ phương trình giải mã khác nhau.

Mã tuyến tính hệ thống, ma trận sinh hệ thống

Một mã tuyến tính $C(n, k)$ được gọi là mã tuyến tính hệ thống nếu mỗi từ mã có một trong hai dạng sau

***Dạng 1*: Từ mã bao gồm phần thông tin k bit đi trước và phần còn lại (gồm $n - k$ bit) đi sau (phần này còn được gọi là phần dư thừa hay phần kiểm tra).**

k bit thông tin	$n - k$ bit kiểm tra

***Dạng 2*: Ngược của dạng 1, từ mã bao gồm phần kiểm tra đi trước và phần thông tin đi sau.**

$n - k$ bit kiểm tra	k bit thông tin

Từ điều kiện về dạng từ mã của mã tuyến tính hệ thống, chúng ta cần xác định dạng của ma trận sinh tương ứng. Đối với mã tuyến tính hệ thống dạng 1, để đáp ứng điều kiện của nó ma trận sinh phải có dạng như sau:

$$G_{k\times n} = \left[\underbrace{\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix}}_{k\times k} \quad \underbrace{\begin{matrix} P_{00} & P_{01} & \cdots & P_{0(n-k-1)} \\ P_{10} & P_{11} & \cdots & P_{1(n-k-1)} \\ \vdots & \vdots & & \vdots \\ P_{(k-1)0} & P_{(k-1)1} & \cdots & P_{(k-1)(n-k-1)} \end{matrix}}_{k\times(n-k)}\right]$$

trong đó k cột đầu của ma trận tạo thành một ma trận đơn vị còn $n - k$ cột sau tuỳ ý với $P_{ij} =$ 0 hoặc 1. Với dạng này chúng ta thấy khi thông báo u được mã hoá thành từ mã v bằng công thức $v = u \times G$ thì k bit thông báo của u sẽ trở thành k bit đầu của từ mã v đáp ứng yêu cầu của mã tuyến tính hệ thống. Ma trận có dạng trên của mã tuyến tính hệ thống được gọi là ma trận sinh hệ thống và có thể biểu diễn đơn giản như sau:

$$G_{k\times n} = [I_{kk} \mid P_{k(n-k)}]$$

trong đó I_{kk} là ma trận đơn vị kích thước $k \times k$.

Tương tự đối với mã tuyến tính hệ thống có dạng 2 thì ma trận sinh hệ thống phải có dạng

$$G_{k\times n} = [P_{k(n-k)} \mid I_{kk}]$$

Qui ước

Nếu không có phát biểu gì khác thì khi dùng mã tuyến tính hệ thống chúng ta sẽ dùng mã tuyến tính hệ thống dạng 1.

Ví dụ 11.2

Ma trận sinh hệ thống cho mã tuyến tính hệ thống tướng ứng với mã tuyến tính trong Ví dụ 11.1 là

$$G_{ht} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Nếu $u = (1101)$ là thông tin cần mã hoá thì từ mã tương ứng sẽ là $v = u \times G_{ht} = (1101000)$. Tương tự nếu $u = (0110)$ thì $v = 0110100$.

Mã tuyến tính hệ thống có lợi điểm là giúp cho việc giải mã từ từ mã thành thông báo nhanh chóng bằng cách lấy k bit đầu hay k bit cuối của từ mã tuỳ theo mã thuộc dạng 1 hay 2 mà không phải giải hệ phương trình như đối với ma trận sinh bình thường. Chẳng hạn đối với mã trong Ví dụ 11.2 nếu chúng ta nhận được từ mã $v = (0101110)$ thì dễ dàng xác định được thông báo tương ứng là $u = 0101$.

Chú ý, từ một ma trận sinh kích thước $k \times n$ chúng ta có thể dùng các phép biến đổi sơ cấp trên hàng (nhân một hàng với một hệ số khác 0, thay một hàng bằng cách cộng hàng đó với một hàng khác) chúng ta có thể biến đổi thành một ma trận có k cột tạo thành một ma trận đơn vị.

Ví dụ 11.3

Cho ma trận sinh sau

$$G_{4\times 7} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Bằng cách thực hiện các phép biến đổi hàng như bên dưới

$$G'_{4\times 7} = \begin{bmatrix} g_0 + g_1 + g_2 + g_3 \\ g_2 \\ g_1 + g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

chúng ta có trong ma trận mới G' có các cột 1, cột 4, cột 6 và cột 3 tạo thành một ma trận đơn vị (các cột được đánh số từ trái sang phải và bắt đầu bằng 1).

11.3 Vấn đề phát hiện sai và sửa sai

Định lý 10.2 cho chúng ta thấy khả năng phát hiện sai và sửa sai của một bộ mã. Ở đây chúng ta chỉ trình bày cách phát hiện sai và sửa sai cho những bộ mã đã thoả điều kiện phát hiện sai và sửa sai như trong Định lý 10.2. Tức là khoảng cách Hamming d của bộ mã và số bit sai t đã thoả Định lý 10.2.

Từ Định lý 10.2 chúng ta rút ra nguyên lý phát hiện sai rất đơn giản như sau: **Kiểm tra xem tổ hợp nhận có phải là từ mã hay không, nếu không thì tổ hợp nhận là sai**.

Việc kiểm tra này có thể được thực hiện bằng cách so trùng tổ hợp nhận được với các từ mã. Như vậy việc kiểm tra này sẽ tốn một số bước bằng với số lượng các từ mã.

Tương tự đối với việc sửa sai chúng ta có nguyên lý sau: **Kiểm tra xem tổ hợp nhận có khoảng cách Hamming gần với từ mã nào nhất, nếu gần với từ mã nào nhất thì từ mã đó chính là từ mã đúng đã được phát đi. Nguyên lý này được gọi là nguyên lý khoảng cách Hamming tối thiểu.** Việc kiểm tra này tốn một số bước bằng với số lượng các từ mã.

Tuy nhiên đối với mã tuyến tính, dựa vào các tính chất của mã chúng ta sẽ có cách phát hiện sai và sửa sai hiệu quả hơn. Các phần tiếp theo sẽ trình bày lần lượt về vấn đề này, nhưng trước hết chúng ta sẽ trình bày một số kiến thức toán học cần thiết cho việc chứng minh một số kết quả trong loại mã này.

Không gian bù trực giao

Cho S là một không gian con k chiều của không gian n chiều V, S_d là tập tất cả các vectơ trong V sao cho $\forall\, u \in S$, $v \in S_d$, thì $u \times v = 0$ (phép nhân ở đây là phép nhân vô hướng của hai vectơ) thì S_d là một không gian con của V và có số chiều là $n - k$. S_d được gọi là không gian bù trực giao của S và ngược lại.

Dựa trên kết quả này chúng ta suy ra rằng với ma trận G bất kỳ kích thước $k \times n$ với k hàng độc lập tuyến tính luôn tồn tại ma trận H kích thước $(n-k) \times n$ với $(n-k)$ hàng độc lập tuyến tính sao cho $G \times H^T = 0$, trong đó H^T là ma trận chuyển vị của ma trận H. Hay nói cách khác các vectơ hàng của H đều trực giao với các vectơ hàng của G.

Cách phát hiện sai

Ứng dụng kết quả trên vào vấn đề phát hiện sai, chúng ta thấy rằng

Nếu v là một từ mã được sinh ra từ ma trận sinh G có ma trận trực giao tương ứng là H thì do v là một tổ hợp tuyến tính của các vectơ hàng của G nên

$$v \times H^T = 0$$

Và ngược lại nếu $v \times H^T = 0$ thì v phải là một tổ hợp tuyến tính của các vectơ hàng của G do đó v là một từ mã.

Syndrome – vectơ sửa sai (corrector)

$v \times H^T$ thường được gọi là syndrome hay vectơ sửa sai của v và kí hiệu là $s(v)$. Vậy chúng ta có

v là từ mã khi và chỉ khi $s(v) = 0$

Với tính chất này chúng ta thấy H có thể được sử dụng để kiểm tra một tổ hợp có phải là từ mã không hay nói cách khác H có thể được dùng để phát hiện sai. Vì lý do này mà ma trận H còn được gọi là ma trận kiểm tra.

Ma trận kiểm tra

Ma trận kiểm tra của một bộ mã có ma trận sinh $G_{k \times n}$ là ma trận H có kích thước $(n-k) \times n$ sao cho

$$G \times H^T = 0$$

Ví dụ 11.4

Tìm ma trận kiểm tra tương ứng với ma trận sinh trong Ví dụ 11.1.

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

Chúng ta thấy ma trận kiểm tra cần tìm phải có kích thước 3×7. Gọi $h = (a_0, a_1, a_2, a_3, a_4, a_5, a_6)$ là một hàng bất kỳ của H. Vì h trực giao với mọi vectơ hàng của G nên chúng ta có hệ bốn phương trình sau tương ứng với bốn hàng của G:

$$\begin{aligned} a_0 + a_1 + a_3 &= 0 \\ a_0 + a_2 + a_3 + a_4 &= 0 \\ a_1 + a_5 + a_6 &= 0 \\ a_0 + a_2 + a_6 &= 0 \end{aligned}$$

Vấn đề là bây giờ chúng ta làm sao tìm được 3 vectơ hàng độc lập tuyến tính là nghiệm của hệ phương trình trên. Chú ý, hệ phương trình trên có thể cho phép chúng ta giải bốn biến theo ba biến còn lại. Chẳng hạn chúng ta giải a_3, a_4, a_5, a_6 theo a_0, a_1, a_2 như sau:

$$\begin{aligned} a_3 &= a_0 + a_1 \\ a_4 &= a_1 + a_2 \\ a_5 &= a_0 + a_1 + a_2 \\ a_6 &= a_0 + a_2 \end{aligned}$$

Chú ý các phép cộng, +, ở đây chính là phép cộng, $\oplus$, trong GF(2) như đã qui ước và trong GF(2) phép trừ – hoàn toàn giống phép +.

Bây giờ chúng ta cho (a_0, a_1, a_2) lần lượt các giá trị (1, 0, 0), (0, 1, 0), (0, 0, 1) thì chúng ta sẽ xác định được (a_3, a_4, a_5, a_6) lần lượt như sau (1, 0, 1, 1), (1, 1, 1, 0), (0, 1, 1, 1). Vậy chúng ta có ma trận H như sau:

$$H_{3\times 7} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Chú ý

Các ma trận kiểm tra khác nhau của cùng một bộ mã đều có khả năng kiểm tra như nhau tức là đều có thể giúp chúng ta phát hiện một tổ hợp có phải là từ mã hay không.

Đối với ma trận sinh hệ thống thì việc xác định ma trận kiểm tra dễ hơn nhiều, dựa trên bổ đề sau:

Bổ đề

Nếu ma trận sinh hệ thống của một mã tuyến tính hệ thống có dạng

$$G_{k\times n} = [I_{kk} \mid P_{k(n-k)}]$$

thì

$$H_{(n-k)\times n} = [P_{k(n-k)}{}^T \mid I_{(n-k)(n-k)}]$$

là một ma trận kiểm tra của mã.

Tương tự nếu ma trận sinh có dạng

$$G_{k\times n} = [P_{k(n-k)} \mid I_{kk}]$$

thì ma trận kiểm tra có dạng

$$H_{(n-k)\times n} = [I_{(n-k)(n-k)} \mid P_{k(n-k)}{}^T]$$

trong đó $I_{(n-k)(n-k)}$ là ma trận đơn vị kích thước $(n-k) \times (n-k)$, còn $P_{k(n-k)}{}^T$ là ma trận chuyển vị của ma trận $P_{k(n-k)}$.

Chứng minh

Xét ma trận sinh hệ thống có dạng 1

$$G_{k\times n} = [I_{kk} \mid P_{k(n-k)}] = \left[\underbrace{\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix}}_{k\times k} \underbrace{\begin{matrix} P_{00} & P_{01} & \cdots & P_{0(n-k-1)} \\ P_{10} & P_{11} & \cdots & P_{1(n-k-1)} \\ \vdots & \vdots & & \vdots \\ P_{(k-1)0} & P_{(k-1)1} & \cdots & P_{(k-1)(n-k-1)} \end{matrix}}_{k\times(n-k)} \right]$$

Xét ma trận H sau

$$H_{(n-k)\times n} = [P_{k(n-k)}{}^T \mid I_{(n-k)(n-k)}] = \left[\underbrace{\begin{matrix} P_{00} & P_{10} & \cdots & P_{(k-1)0} \\ P_{01} & P_{11} & \cdots & P_{(k-1)1} \\ \vdots & \vdots & & \vdots \\ P_{0(n-k-1)} & P_{1(n-k-1)} & \cdots & P_{(k-1)(n-k-1)} \end{matrix}}_{(n-k)\times k} \underbrace{\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix}}_{(n-k)\times(n-k)} \right]$$

Ta chứng minh

$$G \times H^T = 0$$

Để chứng minh điều này ta chứng minh $g_i \times h_j = 0 \ \forall\ i = 0, \ldots, k-1, j = 0, \ldots, n-k-1$ trong đó $g_i = (g_{i0}, \ldots, g_{i(n-1)})$ là hàng i của G còn $h_j = (h_{j0}, \ldots, h_{j(n-1)})$ là hàng j của ma trận H.

Thật vậy ta có

$$g_i \times h_j = \sum_{s=0}^{n-1} g_{is} h_{js} = \sum_{s=0}^{k-1} g_{is} h_{js} + \sum_{s=k}^{n-k-1} g_{is} h_{js} = h_{ji} + g_{i(k+j)} = P_{ij} + P_{ij} = 0$$

Chứng minh tương tự cho dạng còn lại của G.

Khả năng chống nhiễu tương đương

Chúng ta đã biết rằng khả năng phát hiện sai và sửa sai của một mã tuyến tính phụ thuộc vào khoảng cách Hamming của bộ mã. Vì vậy chúng ta định nghĩa rằng

Hai mã tuyến tính $C(n, k)$ được gọi là có khả năng chống nhiễu tương đương nếu chúng có cùng khoảng cách Hamming.

Từ đây chúng ta dẫn đến bổ đề sau

Bổ đề

Nếu hoán vị hai cột của một ma trận sinh sẽ tạo ra một bộ mã mới có khả năng chống nhiễu tương đương với bộ mã cũ. Nói cách khác việc hoán vị hai cột của ma trận sinh không làm thay đổi khả năng chống nhiễu.

Áp dụng điều này nên người ta thường dùng phép hoán vị này cộng với các phép biến đổi sơ cấp trên hàng để tạo ra những ma trận sinh hiệu quả hơn trong việc mã hoá và giải mã nhưng khả năng chống nhiễu vẫn không thay đổi.

Cách tính khoảng cách Hamming của bộ mã

Chúng ta đã biết khoảng cách Hamming của hai từ mã bằng trọng số của tổng hai từ mã đó. Mà do đối với mã tuyến tính tổng hai từ mã là một từ mã nên từ đây chúng ta suy ra khoảng cách Hamming của hai từ mã bằng trọng số của một từ mã nào đó. Khái quát lên chúng ta có bổ đề sau:

Bổ đề

Khoảng cách Hamming của một mã tuyến tính bằng trọng số nhỏ nhất khác 0 của bộ mã.

Vì vậy để tính khoảng cách Hamming của một mã tuyến tính chúng ta sẽ tìm từ mã nào khác không mà có trọng số nhỏ nhất.

Ngoài ra để tính khoảng cách Hamming của một mã tuyến tính chúng ta còn có một cách được phát biểu thông qua bổ đề sau:

Bổ đề

Gọi H là ma trận kiểm tra của một mã tuyến tính, nếu một từ mã có trọng số d thì tồn tại d cột của H có tổng bằng 0.

Hệ quả

Nếu trong ma trận kiểm tra H của một mã tuyến tính số cột phụ thuộc tuyến tính nhỏ nhất là d thì khoảng cách Hamming của bộ mã đó bằng d.

Ví dụ 11.5

Xét ma trận H trong Ví dụ 11.4

$$H_{3\times 7} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Chúng ta thấy số cột phụ thuộc tuyến tính của H là 3, cụ thể là các cột số 3, 4 và 6 (chỉ số được tính bắt đầu từ 1). Vì vậy bộ mã tương ứng có khoảng cách Hamming $d = 3$, do đó mã có thể phát hiện sai 2 bit và sửa sai được 1 bit.

Cách sửa sai

Trước hết chúng ta sẽ định nghĩa khái niệm vectơ lỗi (error pattern vector) và khái niệm tập giải mã (decoding set) hay đôi lúc còn gọi là tập coset là những khái niệm làm nền tảng cho việc sửa sai.

Vectơ lỗi

Là vectơ biểu diễn các vị trí lỗi giữa từ mã truyền và tổ hợp nhận, mỗi vị trí lỗi được biểu diễn bằng bit 1, còn các vị trí còn lại sẽ có giá trị 0.

Nếu từ mã được truyền đi là w, vectơ lỗi là e và vectơ nhận là v thì chúng ta có:

$$v = w + e$$
$$e = v + w$$

Ví dụ nếu từ mã $w = 1011011$, vectơ lỗi là $e = 0010100$ có nghĩa là sai ở vị trí số 3 và 5 (tính từ trái, bắt đầu bằng 1) thì vectơ nhận sẽ là $v = w + e = 1001111$. Ngược lại nếu từ mã $w = 0110010$ còn vec tơ nhận là $v = 0010011$ thì vectơ lỗi là $e = w + v = 0100001$ có nghĩa là đã có sai xảy ra ở các vị trí số 2 và số 7.

Chúng ta thấy trọng số của vectơ lỗi biểu diễn khoảng cách Hamming giữa từ mã phát và tổ hợp nhận. Khái niệm trên gợi ý cho chúng ta một điều như sau, nếu vectơ nhận là v thì chúng ta có thể tính được vectơ lỗi tương ứng với mỗi từ mã bằng cách cộng v với lần lượt các từ mã và rồi dựa vào nguyên lý khoảng cách Hamming tối thiểu chúng ta thấy rằng vectơ lỗi nào có trọng số nhỏ nhất thì từ mã tương ứng chính là từ mã đã được phát đi. Chúng ta sẽ hình thức hoá điều này bằng khái niệm sau:

Tập giải mã – Coset

Cho S là một không gian con các từ mã của không gian V, coset của một phần tử $z \in V$ đối với S được kí hiệu là $z + S$ và được định nghĩa như sau

$$z + S = \{z + w: w \in S\}$$

Bổ đề

Tập coset $z + S$ có các tính chất sau.

(1) $z \in z + S$. Lí do này vì từ mã $0...0 \in S$ nên $z = z + 0...0 \in S$.

(2) **Nếu $z \in S$ thì $z + S = S$.** Lí do này là vì tổng của hai từ mã là một từ mã. Hơn nữa $z + w_i \neq z + w_j$ với $w_i \neq w_j$ nên không có hai phần tử nào của $z + S$ giống nhau.

(3) **Nếu $v \in z + S$ thì $v + S = z + S$.** Thật vậy vì $v = z + w_i$ với một w_i nào đó $\in S$. Vì vậy $v + S = z + w_i + S$. Mà $w_i + S = S$ nên $v + S = z + S$.

(4) **Nếu $v \notin z + S$ thì $v + S$ và $z + S$ rời nhau.** Thật vậy nếu $v + S$ và $z + S$ không rời nhau. Suy ra tồn tại $u \in v + S$ và $u \in z + S$. Mà $u \in v + S$ suy ra $u + S = v + S$. Tương tự $u \in z + S$ suy ra $u + S = z + S$. Vì vậy $v + S = z + S$ mà $v \in v + S$ suy ra $v \in z + S$. Điều này mâu thuẫn. Vì vậy $v + S$ và $z + S$ phải rời nhau.

Chú ý mỗi coset có số phần tử đúng bằng số phần tử của tập S vì tất cả các phần tử $z + w$ ($w \in S$) là khác nhau. Vì vậy, với bổ đề trên chúng ta suy ra số các coset của V bằng số phần tử của V chia cho số phần tử của S. Cụ thể với V là một không gian 2^n vectơ, còn S có 2^k vectơ thì số tập coset sẽ là 2^{n-k}.

Với các khái niệm trên chúng ta đưa ra sơ đồ giải mã theo nguyên lý khoảng cách Hamming tối thiểu như sau.

(1) Với mỗi vectơ nhận v chúng ta sẽ có một tập coset tương ứng là $v + S$.

(2) Trong tập này chọn phần tử có trọng số nhỏ nhất, chẳng hạn là z. Phần tử này thường được gọi là coset leader.

(3) Thông báo từ mã được truyền chính là $w = v + z$.

Rõ ràng coset leader chính là vectơ lỗi mà bộ giải mã theo nguyên lý khoảng cách tối thiểu gán cho v. Chú ý rằng tất cả các thành viên của tập coset $v + S$ có cùng tập coset như v

(theo tính chất 3). Như vậy, tất cả các thành viên của một tập coset có cùng vectơ lỗi chính là coset leader của tập. Vì vậy nếu chúng ta nhận biết được một vectơ nhận thuộc tập coset nào thì cộng nó với coset leader của tập sẽ được từ mã đúng đã được phát đi tương ứng. Đó là lí do tại sao tập coset còn được gọi là tập giải mã.

Vấn đề bây giờ là làm sao để xác định một cách nhanh nhất một vectơ nhận tương ứng với vectơ lỗi nào hay chính là với coset leader nào. May mắn thay có một cách rất dễ để thực hiện điều này, cái mà được phát biểu thông qua bổ đề sau

Bổ đề

Các phần tử của một tập coset có cùng một syndrome như nhau. Các tập coset khác nhau có các syndrome khác nhau.

Chứng minh

Thật vậy chúng ta có với $w_i \in S$ thì

$$s(v + w_i) = (v + w_i) \times H^T = (v \times H^T) + (w_i \times H^T) = (v \times H^T) + 0 = (v \times H^T) = s(v)$$

Vì vậy các phần tử của tập coset $v + S$ có cùng syndrome như nhau và dĩ nhiên có cùng syndrome của coset leader của tập.

Mặt khác nếu $u \notin v + S$. Giả sử $s(u) = s(v)$, suy ra

$$u \times H^T = v \times H^T$$

Từ đây suy ra

$$(u + v) \times H^T = 0$$

Suy ra $u + v = w_j$ với w_j là một từ mã nào đó. Điều này suy ra $u = v + w_j$ có nghĩa là $u \in v + S$ (mâu thuẫn). Điều này hoàn tất chứng minh của chúng ta.

Vậy mỗi coset được đặt trưng bằng một syndrome hay một vectơ sửa sai duy nhất. Các vectơ sửa sai này có kích thước là $n - k$. Vậy có một sự tương ứng một – một giữa 2^{n-k} tập coset với 2^{n-k} vectơ có chiều dài là $n - k$.

Ứng dụng điều này chúng ta thấy bên nhận sẽ chỉ cần giữ một bảng bao gồm 2^{n-k} dãy có chiều dài $n - k$, mỗi dãy tương ứng với một vectơ lỗi chính là coset leader của các tập coset. Với mỗi vectơ nhận v, bộ giải mã sẽ tính $s(v) = v \times H^T$ rồi tìm trong bảng để xác định vectơ lỗi e tương ứng với $s(v)$. Cuối cùng bộ giải mã sẽ thông báo từ mã đúng được phát đi là $w = v + e$.

Đặt $e = (a_1, a_2, ..., a_n)$, và gọi các cột của H lần lượt bằng $h_1, h_2, ..., h_n$ thì chúng ta có

$$s(e) = e \times H^T = \sum_{i=1}^{n} a_i h_i = \sum_{a_i \neq 0} a_i h_i$$

Có nghĩa là $s(e)$ bằng tổng những cột ở những vị trí tương ứng với những vị trí bằng 1 của e. Chẳng hạn nếu vị trí lỗi sai trong khi truyền là 3 thì syndrome của vectơ nhận tương ứng sẽ bằng cột số 3 của ma trận kiểm tra H. Chi tiết hơn chúng ta xem ví dụ sau đây.

Ví dụ 11.6

Xét một mã tuyến tính $C(7, 4)$ có ma trận sinh hệ thống như sau

$$G_{4\times 7} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Từ đây chúng ta có một ma trận kiểm tra của bộ mã trên như sau:

$$H_{3\times 7} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Bộ mã trên có khoảng cách Hamming $d = 3$. Vì vậy có thể phát hiện sai 2 bit và sửa sai được 1 bit.

Giả sử đường truyền sai tối đa 1 bit. Và vectơ nhận là $v = 1010110$, hãy tìm từ mã đúng đã được phát đi. Để giải bài này chúng ta tính

$$s(v) = v \times H^T = h_1 + h_3 + h_5 + h_6 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

trong đó h_i là cột thứ i của H (tính từ trái và bắt đầu bằng 1).

Chúng ta thấy $s(v) \equiv h_1$ nên suy ra lỗi sai ở vị trí số 1, vì vậy từ mã đúng đã được phát đi là

$$w = v + e = 1010110 + 1000000 = 0010110$$

Từ ý tưởng này gợi ý cho chúng ta một loại mã cho phép phát hiện sai 1 bit nhanh nhất. Mã này có tên gọi là mã tuyến tính Hamming.

Mã tuyến tính Hamming

Mã tuyến tính Hamming là mã có ma trận H có tính chất giá trị của cột h_i bằng i ($i = 1, 2, ...$

Ví dụ ma trận sau biểu diễn mã tuyến tính Hamming $C(7, 4)$.

$$H_{3\times 7} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Bổ đề

Các mã tuyến tính Hamming đều có khoảng cách Hamming $d = 3$. Vì vậy có thể phát hiện sai 2 bit và sửa sai 1 bit.

Chứng minh

Dễ thấy với cách định nghĩa của ma trận H, chúng ta thấy số cột phụ thuộc tuyến tính ít nhất của H là 3 (chẳng hạn 3 cột đầu). Vì vậy mã Hamming có $d = 3$.

Mã tuyến tính Hamming cho phép chúng ta sửa sai một bit một cách đơn giản như sau.

(1) Tính syndrome $s(v)$ của vectơ nhận

(2) Đổi chuỗi nhị phân tương ứng ra giá trị thập phân, kết quả đổi sẽ là vị trí lỗi sai đã xảy ra.

(3) Sửa sai ở vị trí lỗi sai tương ứng.

Chẳng hạn lấy mã tuyến tính Hamming $C(7, 4)$ như trên nếu $s(v) = 101$ thì vị trí lỗi sai là 5 (vì 101_2 đổi ra thập phân bằng 5).

Một bài toán còn lại đặt ra ở đây đối với mã tuyến tính Hamming là, xác định ma trận sinh G để thực hiện việc mã hoá và giải mã. Để làm điều này người ta thường lấy k trong n vị trí để chứa các bit thông tin, các bit còn lại sẽ làm các bit kiểm tra. Chi tiết hơn chúng ta xem ví dụ sau đây.

Ví dụ 11.7

Xét mã tuyến tính Hamming $C(7, 4)$ có các bit thông tin nằm ở các vị trí 3, 5, 6, 7. Hãy xác định ma trận sinh G của bộ mã.

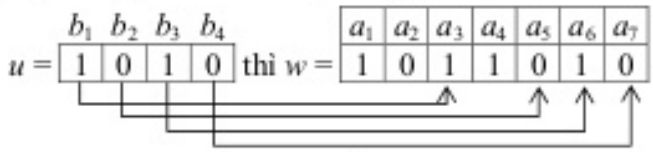
Gọi $w = (a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ là một từ mã. Chúng ta có hệ phương trình sau được dẫn ra từ công thức $w \times H^T = 0$.

$$\begin{aligned} a_4 + a_5 + a_6 + a_7 &= 0 \\ a_2 + a_3 + a_6 + a_7 &= 0 \\ a_1 + a_3 + a_5 + a_7 &= 0 \end{aligned}$$

Từ đây chúng ta suy ra công thức tính các bit kiểm tra a_1, a_2, a_4 theo các bit thông báo a_3, a_5, a_6, a_7 như sau

$$a_1 = a_3 + a_5 + a_7$$
$$a_2 = a_3 + a_6 + a_7$$
$$a_4 = a_5 + a_6 + a_7$$

Công thức này cho phép chúng ta mã hoá được thông báo u thành từ mã w. Chẳng hạn nếu



Biến đổi công thức trên thành dạng $w = u \times G$ để cho phép chúng ta tìm G được dễ dàng:

$$a_1 = b_1 + b_2 + b_4$$
$$a_2 = b_1 + b_3 + b_4$$
$$a_3 = b_1$$
$$a_4 = b_2 + b_3 + b_4$$
$$a_5 = b_2$$
$$a_6 = b_3$$
$$a_7 = b_4$$

Từ đây chúng ta suy ra được ma trận sinh G như sau:

$$G_{4\times 7} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

11.4 Một số giới hạn

Giới hạn trên Hamming về số lượng từ mã

Định lý 11.1

Nếu bộ mã $\{w_1, \ldots, w_M\}$ gồm các từ mã nhị phân chiều dài n có thể sửa sai các lỗi sai ≤ t bit, thì số lượng từ mã M phải thoã bất đẳng thức sau:

$$M \le \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i}}$$

Chứng minh

Đối với mỗi từ mã w_i định nghĩa một tập A_i bao gồm tất cả các dãy v_j có khoảng cách Hamming so với w_i nhỏ hơn hay bằng t. Tổng số dãy trong A_i được cho bằng

$$1 + n + \binom{n}{2} + \cdots + \binom{n}{t} = \sum_{i=0}^{t} \binom{n}{i}$$

Để có thể sửa sai được các lỗi sai ≤ t bit thì các tập $A_1, A_2, \ldots, A_M$ phải rời nhau. Mà tổng số phần tử của tất cả các tập A_i phải nhỏ hơn hoặc bằng tổng số dãy có chiều dài n. Vì vậy

$$M \sum_{i=0}^{t} \binom{n}{i} \le 2^n$$

Từ đây suy ra điều phải chứng minh.

Giới hạn dưới về số lượng bit kiểm tra

Định lý 11.2

Nếu một mã tuyến tính $C(n, k)$ có thể sửa sai các lỗi sai $\le t$ bit, thì số bit kiểm tra $r = n - k$ phải thoã bất đẳng thức sau:

$$2^r \ge \sum_{i=0}^{t} \binom{n}{i}$$

Chứng minh

Đối với mỗi vectơ lỗi e chúng ta có một syndrome (hay vectơ sửa sai) tương ứng. Vậy để sửa sai tất cả các lỗi sai $\le t$ bit thì chúng ta có $\sum_{i=0}^{t} \binom{n}{i}$ vectơ lỗi vì vậy có $\sum_{i=0}^{t} \binom{n}{i}$ vectơ sửa sai. Mà tổng số vectơ sửa sai là bằng $2^{n-k} = 2^r$. Thật vậy vì các vectơ thuộc cùng một tập coset thì có cùng một vectơ sửa sai. Do đó số lượng vectơ sửa sai bằng với số lượng tập coset tức bằng 2^{n-k}. Vì vậy chúng ta phải có

$$2^r \ge \sum_{i=0}^{t} \binom{n}{i}$$

Điều này hoàn tất chứng minh.

Chú ý định lý này chỉ là điều kiện cần chứ không đủ. Chẳng hạn với $n = 10$, $t = 2$ chúng ta suy ra từ định lý trên rằng $r \ge 6$ là cần thiết. Tuy nhiên, chúng ta có thể kiểm tra lại rằng để sửa được các lỗi sai ≤ 2 bit thì phải có ít nhất $r = 7$.

Chú ý giới hạn dưới về số lượng bit kiểm tra đối với mã tuyến tính giống với giới hạn trên về số lượng từ mã trong Định lý 11.1. Thật vậy đối với mã tuyến tính $C(n, k)$ theo Định lý 11.1 chúng ta có $M = 2^k \le \dfrac{2^n}{\sum_{i=0}^{t} \binom{n}{i}}$. Từ đây suy ra $2^r = 2^{n-k} \ge \sum_{i=0}^{t} \binom{n}{i}$.