Projeto 1 de Sistemas Digitais 1
Código de Hamming (4,7)

I. Contextualização

Considere o problema de comunicação onde o emissor envia uma mensagem binária $\vec{x}$ e a envia para o receptor por um canal ruidoso. Como o canal é ruidoso, o receptor recebe a mensagem $\vec{y}$, onde, em geral, $\vec{y} \neq \vec{x}$. Como fazer para minimizar os erros decorrentes desse ruído?

A Teoria de Codificação de Canal (ou códigos corretores de erro), proposta por Shannon na década de 40 [1], [2], diz que existem códigos de bloco tais que possibilitam a transmissão de informação com uma probabilidade de erro arbitrariamente pequena, desde que o tamanho do bloco seja grande o suficiente (o que, naturalmente, resulta em uma taxa menor do que a capacidade do canal).

O objetivo da Codificação de Canal é inserir redundância na mensagem de forma que, mesmo que parte da informação seja perdida ou corrompida, seja possível recuperar a mensagem no receptor. O esquema mais simples é simplesmente repetir a informação (esses códigos são chamados de códigos de repetição). Por exemplo, se deseja-se enviar o bit $\mathbf{1}$, envia-se na verdade o código $\mathbf{11111}$ e, para enviar o bit $\mathbf{0}$, envia-se o código $\mathbf{00000}$. Este esquema envia 5 símbolos para enviar apenas 1 bit e, por isso, dizemos que ele tem uma taxa de $\frac{1}{5}$ bits por símbolo. Neste esquema, se o receptor receber 3 ou mais bits iguais a $\mathbf{1}$, a informação é decodificada como $\mathbf{1}$, caso contrário, a informação é decodificada como $\mathbf{0}$. Um erro irá ocorrer *se e somente se* três ou mais bits forem corrompidos. Podemos aumentar a capacidade de correção de erros (ou seja, diminuir a probabilidade de erro) utlizando repetições mais longas, porém, a taxa do código (a quantidade de informação de fato que é enviada por cada símbolo) também tende a zero.

Podemos melhorar esses códigos de repetição combinando bits de forma mais inteligente. Um exemplo simples são os códigos de paridade. Considere um bloco de $n-1$ bits de informação. Podemos escolher o n-ésimo bit de forma que a paridade do bloco inteiro seja 0 (ou seja, o número de 1s seja par). Se houver apenas um erro de transmissão (na verdade, se houve um *número ímpar* de erros de transmissão), o receptor vai perceber que a paridade recebida será ímpar e detectar o erro. Este código não é capaz de *corrigir* erros, apenas de detectá-los, e é chamado de *código detector de erros*.

Podemos extender essa ideia para permitir mais que um bit de paridade, de forma que possamos não apenas detectar erros mas também corrigí-los. Um tipo especial de códigos de blocos lineares foi proposto por Hamming [3], e são chamados de *códigos de Hamming*.

II. Códigos de Hamming

O esquema geral do código de Hamming utilizado aqui é mostrado na Fig. II. A mensagem que se quer enviar, $\vec{m}$, tem 4 bits de informação. Codifica-se essa mensagem, gerando a palavra-código (*codeword*) $\vec{x}$, que tem 7 bits. Essa mensagem é enviada pelo canal e recebida como $\vec{y}$ no receptor. Como podem ter acontecido erros, $\vec{x}$ não é necessariamente igual a $\vec{y}$. O decodificador então decodifica essa palavra-código, decodificando a mensagem $\vec{m'}$.

Definimos aqui a *distância de Hamming* $d(\vec{x}, \vec{y})$ entre duas palavras binárias $\vec{x}$ e $\vec{y}$ como o número de bits diferentes entre as duas palavras. Por exemplo, se:

$$\vec{x} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$
$$\vec{y} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$
$$d(\vec{x}, \vec{y}) = 3$$

Assim, o código que descreveremos abaixo garante que:

$$\vec{m'} = \vec{m}, \text{se } d(\vec{x}, \vec{y}) \leq 1 \tag{1}$$

ou seja, a mensagem recebida será igual a mensagem enviada se houver no máximo um erro.

A. Codificador

Para gerar a palavra-código $\vec{x}$, utilizamos a matriz geradora do código $\mathbf{G}$. Existem diversas matrizes geradoras para códigos de Hamming, mas a matriz que utilizaremos é:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

E a palavra-código $\vec{x}$ é formada por uma simples multiplicação de matrizes:

$$\vec{x} = \vec{m}\mathbf{G} \tag{3}$$

onde a única diferença é que *a matemática é feita em módulo 2! Isto é,* $1 + 1 = 0$.

Uma boa propriedade da matriz $\mathbf{G}$ dada é que ela gera *códigos sistemáticos*. Isto é, se a mensagem $\vec{m} = \begin{bmatrix} b_3 & b_2 & b1 & b0 \end{bmatrix}$, então a palavra-código $\vec{x} = \vec{m}\mathbf{G} = \begin{bmatrix} b_3 & b_2 & b1 & b0 & p_2 & p_1 & p_0 \end{bmatrix}$, ou seja, para toda palavra-código, os primeiros 4 bits são sempre os bits da mensagem a ser enviada. Esta propriedade facilita a decodificação.

B. Decodificador

A decodificação da palavra recebida $\vec{y}$ é feita por um decodificador de máxima verossimilhança (*maximum likelihood*), onde, para uma dada palavra $\vec{y}$ recebida, escolhe-se a palavra-código $\vec{x}$ mais próxima de $\vec{y}$ (isto é, a palavra-código com a menor distância de Hamming para a palavra-código $\vec{x}$). Então, a mensagem recebida decodificada $\vec{m'}$ é decodificada como sendo a mensagem $\vec{m}$ que gera $\vec{x}$. Caso o código seja sistemático, a mensagem $\vec{m}$ será composta exatamente pelos primeiros 4 bits de $\vec{x}$!

O decodificador por máxima-verossimilhança descrito acima pode ser implementado por meio de tabelas, isto é, calcula-se todas as possibilidades para as palavras recebidas $\vec{y}$ e guarda-se, para cada uma delas, a mensagem $\vec{m'}$ que será decodificada quando $\vec{y}$ for recebido. Porém, um algoritmo mais simples pode ser concebido utilizando a matriz de paridade $\mathbf{H}$ da matriz geradora $\mathbf{G}$:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

A matriz de paridade $\mathbf{H}$ pode ser encontrada a partir de $\mathbf{G}$ resolvendo a equação $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. O algoritmo para decodificar a palavra recebida $\vec{y}$ consiste em encontrar a *síndrome* (*syndromme*) $\vec{s}$:

$$\vec{s} = \mathbf{H}\vec{y}^T \tag{5}$$

onde, novamente, a matemática é feita em módulo 2 (isto é, $1 + 1 = 0$). Uma propriedade interessante de $\mathbf{H}$ é que todas as palavras código $\vec{x}$ fazem parte do espaço-nulo (*null space*) de $\mathbf{H}$, ou seja, $\mathbf{H}\vec{x}^T = 0$. Logo, caso a síndrome seja 0, sabemos imediatamente que $\vec{y}$ recebido é uma palavra-código válida e portanto assume-se que não ocorreram erros.

Além disso, a síndrome calculada pela Eq. 5 diz exatamente *qual dos bits de $\vec{y}$ deve ser invertido para transformar $\vec{y}$ em uma palavra código correta*! Ou seja, a síndrome me diz qual o bit mais provável de ter sido corrompido. Por exemplo, se $\vec{s} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, sabemos que o primeiro bit deve ter sido corrompido e, portanto, para corrigí-lo, basta inverter esse bit.

C. Exemplo

Considere que queremos enviar a mensagem $\vec{m} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}$. Primeiro, encontramos a palavra-código $\vec{x}$, de acordo com a Eq. 3:

$$\vec{x} = \vec{m}\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

Considere agora que houve um erro e que a palavra recebida foi $\vec{y} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$. Para decodificar, primeiro encontramos a síndrome usando a Eq. 5:

$$\vec{s} = \mathbf{H}\vec{y}^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{7}$$

E, portanto, sabemos que o erro está no terceiro bit (pois $\vec{s} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$) e, para corrigí-lo, invertemos este bit e teremos $\vec{y_c} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$. Agora, encontramos a mensagem $\vec{m}$ que geraria a palavra $\vec{y_c}$, isto é, os 4 primeiros bits. Portanto: $\vec{m'} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}$, que é justamente a mensagem que queríamos enviar.

III. Projeto

O projeto consiste em:

1) Projetar o circuito combinacional para gerar a palavra-código $\vec{x}$ a partir de uma mensagem $\vec{m}$ (isto é, o codificador para o código de Hamming).
2) Projetar o circuito combinacional para decodificar a palavra-código recebida $\vec{y}$ em uma mensagem $\vec{m'}$ (isto é, o decodificador para o código de Hamming).

Os circuitos devem ser projetados pensando-se na eficiência em termos de número de portas (isto é, o circuito deve ser minimizado). O projeto entregue deve conter uma descrição dos circuitos (número de entradas e saídas), as equações e o esquemático do circuito, bem como uma breve explicação de como o circuito foi projetado.

References

[1] C. E. Shannon, "A mathematical theory of cryptography," Bell Labs, Bell Labs Technical Report, September 1945.
[2] ——, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423,623–656, 1948.
[3] R. V. Hamming, "Error detecting and error correcting codes," *Bell Systems Technical Journal*, vol. 29, pp. 147–160, 1950.