

Aula 10 – 19-11-2004

- Sumário

Correcção de erros – distância de Hamming.
Outros códigos de correcção de erros de blocos.

- Bibliografia

CN3 3.2

Códigos de correcção de erros

- Para detectar e corrigir erros
- Acrescenta redundância à mensagem original
- Indispensável quando é impossível reenviar a mensagem (comunicações interplanetárias, armazenamento de informação) ou quando o canal é muito ruidoso (comunicação wireless)

Códigos de bloco

- Associado a cada n bits de dados há k bits redundantes
- Redução do ritmo efectivo de transmissão – $n/(n+k)$ – esta redução permite atingir um determinado BER (bit error rate)

Distância de Hamming

- Distância de Hamming – para duas sequências binárias de n -bits, é o número de bits diferentes:
 - Ex., $v_1=011011$; $v_2=110001$; $d(v_1, v_2)=3$
 - Pode ser feito com um XOR

Distância de Hamming de um código

- Exemplo de codificação

A	-> 00	-> 00000
B	-> 10	-> 10110
C	-> 01	-> 01011
D	-> 11	-> 11101

- A distância de Hamming de um código d é a distância mínima entre quaisquer dois códigos distintos
- No exemplo, $d=3$

Set 2004 - Jan 2005

TI

5

Detecção de erros baseada na distância de Hamming

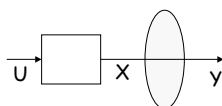
- Com n bits de dados e k bits de verificação são possíveis
 - $2^{(n+k)}$ configurações
 - 2^n deles são usadas
- As 2^n configurações válidas são escolhidas de forma a que exista entre elas uma distância d
- Quando é recebido um código não válido é assumido o código com distância menor (*descodificação pela máxima semelhança*)

Set 2004 - Jan 2005

TI

6

Descodificação pela máxima semelhança



Canal com ruído

- Descodificar pela semelhança máxima corresponde a encontrar a mensagem U que maximiza a probabilidade condicional $P(U|Y)$

Set 2004 - Jan 2005

TI

7

Correcção de erros

- Maior distância de Hamming entre palavras de código válidas
- Hipótese básica
 - Se o padrão de bits x recebido não corresponde a nenhum código válido, e a palavra de código a é que tem a menor distância de Hamming para x então é provável que x seja uma versão corrompida do código a
 - Quanto maior for a separação (distância de Hamming entre códigos válidos), maior será a probabilidade dessa correspondência

Set 2004 - Jan 2005

TI

8

Exemplo

- Códigos com 4 bits
- Dois códigos válidos
 - 0000
 - 1111
- Qual é a distância de Hamming deste código?
- Quantos bits errados podemos detectar? E corrigir?
- Suponha-se que se recebe 1110
 - Assume-se que é uma versão corrompida of 1111, e não 0000
- Suponha-se que se recebe 1100?

De uma forma geral,

- É preciso decidir que o padrão de bits recebido é mais próximo de uma palavra de código válida do que de qualquer outra.
- Os empates não ajudam
- Se a distância entre códigos válidos é m
 - Detectam-se até $m-1$ bits errados
 - Corrigem-se até $(m-1)/2$ (arredondado para baixo) bits errados

Exemplos

- Se a distância de Hamming é
 - 5
 - Detectam-se até 4 bits errados
 - Corrigem-se até 2 bits errados
 - 13
 - Detectam-se 12 bits errados
 - Corrigem-se até 6 bits errados
 - 9
 - Detectam-se até ____ bits errados
 - Corrigem-se até ____ bits errados

Código de Hamming

- Permite correcção automática de 1 bit errado
- Distância de Hamming: 3

Código de Hamming

- Forma sistemática de atribuir valores aos bits redundantes de forma a que se saiba o código
- Numerar os bits da esquerda para a direita, começando em 1
- As posições que são potências de 2 são bits de verificação
- Todas as outras posições são bits de dados "reais"

Set 2004 - Jan 2005

TI

13

Cálculo dos bits de verificação de Hamming

- Colocar os bits de dados deixando das posições de verificação vazias
- Preencher o bit de verificação 1
 - Contar os 1s nas posições 3, 5, 7, 9, 11, 13, ... - todas as ímpares
 - Atribuir o valor 1 à posição 1 que torne o número de 1s par
- Preencher o bit de verificação 2
 - Paridade par com os bits 3, 6, 7, 10, 11, 14, etc.
- Preencher o bit de verificação 4
 - Paridade par com as posições positions 5, 6, 7, 12, 13, 14, etc.
- Preencher o bit de verificação 8
 - 8, 9, 10, 11, 12, 13 ... e assim por diante

Set 2004 - Jan 2005

TI

14

Exemplo(1)

- Palavra de dados com 8 bits. Quantos bits de verificação são precisos?
- Seja a palavra de dados = 0110 1110
- Palavra de Hamming (sem bits de verificação):

12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	0		1	1	1		0		

Set 2004 - Jan 2005

TI

15

Exemplo (2)

- Palavra de Hamming (com bits de verificação):

12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	0	0	1	1	1	1	0	0	1

- Suponha-se que quando se lê o dado se recebe : 0110 1111 (há um erro no bit 3 da palavra de Hamming)

Set 2004 - Jan 2005

TI

16

Exemplo(3)

- Recalculam-se os bits de verificação

12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	0	0	1	1	1	1	1	1	0

0101 – bits de verificação recebidos
 XOR 0110 - bits de verificação calculados

 0011 (erro no bit 3)

Comentários sobre o método de Hamming

- Número de bits necessários para uma codificação de Hamming para K bits de dados:
 - K + V bits de verificação
 - Com V bits de verificação é possível representar valores entre 0 (não é usado) e $2^V - 1$
 - V é o menor inteiro que verifica a condição

$$2^V - 1 \geq K + V$$

Família de códigos de Hamming

k	n	R=k/n
4	7	0.57
11	15	0.73
26	31	0.84
57	63	0.91
120	127	0.94
247	255	0.968
502	511	0.982

- A eficiência aumenta à medida que o tamanho do bloco aumenta
- No entanto, à medida que o tamanho do bloco aumenta, surgem dificuldades:
 - Complexidade do codificador aumenta
 - É preciso juntar muitos bits antes de transmitir, o que pode criar atrasos intoleráveis

Códigos BCH (Bose-Chaudhuri-Hocquenghem) 1960

- O código de Hamming faz parte dos códigos BCH
- Os códigos BCH podem corrigir um número de erros tão grande quanto se queira (desde que o bloco tenha bits suficientes)
- Têm um rendimento R=k/n muito baixo
- Usados quando a fiabilidade é essencial

k	n	R=k/n	Erros corrigidos
4	7	0.57	1
5	15	0.33	3
24	63	0.38	7
64	127	0.5	10
247	255	0.97	1
171	255	0.67	11
11	1023	0.01	255

Códigos de Reed-Solomon

- Um código RS(N,K) tem os parâmetros:
 - K número de bytes de informação
 - N número de bytes total num bloco
 - N-K "syndrome bytes"
- É capaz de corrigir (N-K)/2 erros nos K bytes de informação
- Exemplo RS(255,223)
 - 223 bytes de informação
 - 32 bytes de correcção de erros
 - Corrige até 16 bytes errados
- Baseado na teoria dos campos de Galois
- Fácil de implementar em hardware

Set 2004 - Jan 2005

TI

25

Códigos RS (Reed-Solomon)

- Usados em sistemas móveis em alternativa ou em conjunto com "interleaving"
- Usado para corrigir erros (e riscos) nas superfícies dos CDs
- Usado nos sistemas RAID

Set 2004 - Jan 2005

TI

26

Aplicações da correcção de erros – discos RAID

- RAID 0 – blocos espalhados por vários discos (não há tolerância a falhas)
- RAID 1 – "mirror"
- RAID 2 – 1 bit por disco; n discos para dados; correcção de erros por código de Hamming: k discos
- RAID 3 – 1 bit por disco; n discos para dados; 1 disco para bit de paridade
- RAID 4 – n discos com "stripes" de dados; 1 disco para "stripe" de paridade
- RAID 5 – idem; mas com "stripe" de paridade rotativo
- RAID 6 – paridade + reed-solomon

Set 2004 - Jan 2005

TI

27

RAID 2

- K discos de dados
- Número de discos V para correcção de erros é o menor k tque verifica a condição:

$$2^V - 1 \geq K + V$$

- 8 discos com dados (K=8). O menor V que verifica a condição é 4; 4 discos de código de Hamming
- 4 discos de dados (K=4). V = 3

Set 2004 - Jan 2005

TI

28

RAID 3

- 1 bit por disco; $b_0, b_1, b_2, b_3, \dots, b_n$
- O bit de paridade é calculado por hardware:
Bit paridade = $b_0 \text{ Xor } b_1 \text{ Xor } b_2 \text{ Xor } \dots \text{ Xor } b_n$ ou
Bit paridade = $b_0 + b_1 + b_2 + \dots + b_n \pmod{2}$
- Um disco em falha pode ser reconstruído usando o mesmo método. Supondo que o disco 3 falhou:
 $b_3 = b_0 \text{ Xor } b_1 \text{ Xor } b_2 \text{ Xor Bit paridade Xor } b_4 \text{ Xor } \dots$

Set 2004 - Jan 2005

TI

29

RAID 4 e RAID 5

- O “stripe” de paridade é calculado fazendo o XOR dos bits dos “stripes” de dados
- O disco de paridade é um gargalo no RAID 4; resolvido no RAID 5 através da paridade rotativa
- Recuperação em caso de erro como no RAID 3

Set 2004 - Jan 2005

TI

30

RAID 6

- Tolera a falha de mais do que um disco
- Cada “linha” de N discos de dados tem dois métodos de correção de erros
 - 1 disco com um “stripe” de paridade
 - 1 disco com um “stripe” calculado a partir de um código de Reed-Solomon
- Stripes de paridade e de Reed-Solomon rodam nos discos
- Poucas implementações comerciais
 - Má performance em escrita (3 discos: dados, paridade, RS)

Set 2004 - Jan 2005

TI

31