Lecture 1:

Math 342: Algebra and Coding Theory
Introduction to abstract algebra through coding theory.

Abstract algebra: theory developed for solving polynomial equations; turned out to have surprising applications to modern technology, such as error-correcting codes.

Will develop theory along with applications, emphasizing both computation and proof.

Pre-req:
Linear Algebra: MATH 152, MATH 221, MATH 223 AND
Proof Techniques: MATH 220,MATH 226, CPSC 121.

Evaluation:
HW: (bi-weekly) (15%) – you may work together or ask me for help or go to MLC (LSK 301-302), but you must turn in your own work.

First HW: Due on Thursday, January 21, in class.

Two midterms (35%), Feb 11 and March 17

Final exam (50%)

15-35-50 distribution is rough

Textbook: A First Course in Coding Theory, by Raymond Hill (ch. 1-8 + other material); old book, but a good introduction to fundamental material.

Office Hours (Math Building 218): Tuesday, Wednesday, 1:30-2:50, + appt.

Contact: marcus@math.ubc.ca
All administrative information (e.g., HW) posted on course website:

http://www.math.ubc.ca/∼marcus/Math342

(or link through undergraduate tab of Math Dept website or my personal website)

Today:

– Motivation for coding theory and simple examples

– Not-for-marks test: proof skills, from 10:15 - 10:45; just to get an idea of your proof skills; do NOT put your name on your paper; may have another test at the end of the term to assess improvement.

For this course,
CODING THEORY = ERROR-CORRECTION CODING

– But will briefly describe other kinds of coding: cryptographic codes, data compression codes

Basic framework: communication over a noisy channel:

SENDER $\longrightarrow$ NOISY CHANNEL $\longrightarrow$ RECEIVER

Message $\longrightarrow$ Received Message

Information to be transmitted can be data, images, voice

EXAMPLES of Noisy Channels:

– Communications (from here to there):

Telephone, Cell Phone, TV, Radio, Satelitte Communication (orbital and deep space), Internet communication

– Recording or Storage (from now to then):

Hard disk drives for computers, Flashstick, CD, DVD, IPOD, Digital Cameras

Bar codes for product identification (read by scanners)

Errors are inevitable! Sources of noise:

– Electronics noise, mechanical imperfections,atmospheric disturbances, signal power loss over large distances, interference from structures or cosmic rays, . . .

– Humans pushing technology to limits

Simple Mathematical Channel Model: *Binary Symmetric Channel (BSC)* and diagram:

—- binary inputs 0,1

—- binary outputs 0,1

—- a "channel error" means that an input bit is flipped (because of noise), i.e., a transmitted 0 is received as a 1 or a transmitted 1 is received as a 0.

—- error (i.e., flipped bit) with *channel error probability* or *crossover probability* p: think of p as small, say $p \approx 10^{-4}$.

Q: How to detect or correct errors?

A: Add redundancy in order to detect or correct (a limited number of) errors.

SENDER $\longrightarrow$ ENCODER(1-1 mapping) $\longrightarrow$ NOISY CHANNEL

$\longrightarrow$ DECODER (not 1-1) $\longrightarrow$ RECEIVER

WORD $\longrightarrow$ CODEWORD $\longrightarrow$

RECEIVED WORD $\longrightarrow$ DECODED MESSAGE

Example: *2-repetition code*

– Encoder (1-1 mapping):

Message $\longrightarrow$ codeword

$0 \longrightarrow 00$

$1 \longrightarrow 11$

– Decoder (NOT a 1-1 mapping):

$00 \longrightarrow 0$

$11 \longrightarrow 1$

01 or 10: Declare an error and request re-transmission.

Performance, within each 2-bit codeword:

– If neither bit is in error, then transmission is fine.

– If only one of the two bits is in error, then error is detected.

– If both bits are in error, then the decoder mis-corrects.

We say that the code is *1-error-detecting*.

Trade-off: Message Transmission rate is cut in half: it takes two coded bits to represent one message bit. No free lunch!!

Other problems with error detection:

– Need a reliable "return channel" to request re-transmission

– Sometimes transmitted information is gone immediately after transmission

– Re-transmission causes delay in reception of information

It would be much better if we can actually *correct* errors rather than merely detect errors, in a manner that it transparent to the receiver. We can!

– Example: *3-repetition code:*

Encoder:

Message $\longrightarrow$ codeword

$0 \longrightarrow 000$

$1 \longrightarrow 111$

Decoder (Majority vote):

000 or 100 or 010 or 001 $\longrightarrow 0$

111 or 011 or 101 or 110 $\longrightarrow 1$

Performance within each 3-bit codeword:
– If no errors are made, transmission is fine.
– If only 1 error is made, then error is corrected.

– If 2 or 3 errors are made, then the decoder will mis-correct.

We say that the code is *1-error-correcting*.
In contrast, the 2-repetition code cannot correct any errors.

Hamming story: from error-detection to error-correction.

Trade-off: Message Transmission rate is cut by a factor of three: it takes three coded bits to represent one message bit.

Lecture 2:

To discuss your skills test paper, contact Sandi Merchant merchant@math.ubc.ca

Recall 3-repetition code, which can correct any single error in each 3-bit codeword.

Transmit a string of information by several repeated uses of the 3-repetition code:

message:	0	1	0	0	1	1	0
transmitted codewords:	000	111	000	000	111	111	000
received words:	010	111	110	000	101	001	001
decoded message:	0	1	1	0	1	0	0

In some situations, you can cope with a small number of message errors (e.g., images). In other situations, you need to use a more powerful error-correcting code or a cascade of more than one code.

Is the tradeoff between error correction and message transmission rate worthwhile?

Coded error probability:

– Recall BSC with channel error probability p: diagram

– Assume channel errors are made independently, from time slot to time slot.

– For 3-repetition code, a decoding error is made if 2 or 3 channel errors are made (within a 3-bit block). So, the *coded error probability* is

$p_c := 3p^2(1-p) + p^3 = 3p^2 - 2p^3 << p$, if p is small; e.g., for $p = 10^{-4}$, then is $3 * 10^{-8} - 2 * 10^{-12} < 10^{-7}$.

Reduction in error probability by several orders of magnitude at the cost of slowing the *message* transmission rate by a factor of 3.

Alternatively, we can often use an error-correcting code to actually increase the message transmission rate without increasing the error probability.

– Say that channel operates at R *channel* bits per second, with channel error probability $p = p(R)$

– With the 3-repetition code, it operates at $R/3$ *message* bits per second, with decoded error probability $p_c(R) << p(R)$

– Increase the channel transmission rate from R to kR where k is maximized subject to $p_c(kR) \leq p(R)$.

– Then, with the 3-repetition code, we get $kR/3$ message bits per second with decoded error probability $\leq p(R)$.

– If $k > 3$, we win: coding gives a higher message information rate, than without coding.

– Whether $k > 3$, depends on the nature of the function $p(R)$, which depends on the details of the physical channel.

Decoder tries to recover the message word in two steps:

(i) given received word, finds the most likely transmitted codeword (difficult part)

(ii) inverts the encoder (easy part)

Note: sometimes "decoder" means a function that does only step 1.

Imagine the space of all received words, and balls around each codeword c representing the most likely possible received words. In order to correct errors, the balls should be disjoint or nearly disjoint.

Objective 1: Thus, we want *well-separated codewords*.

Objective 2: On the other hand, we want to transmit lots of distinct messages by distinct codewords and so there should be *lots of codewords*.

These are the two main objectives of a "good" code.

There is a fight between the objectives.

Notation: for a finite set S, $|S|$ denotes the number of elements in S.

Formal Defns:

Code alphabet (symbols): any set finite set, written $F_q = \{a_1, a_2, \ldots, a_q\}$. Usually, $F_q = \mathbb{Z}_q = \{0, 1, \ldots, q-1\}$

– so $q = |F_q|$

– main example: $F_2 = \mathbb{Z}_2 = \{0, 1\}$, $q = 2$ (the binary case)

q-ary word of length n over F_q: a sequence (string) $\overline{x} = x_1 x_2 \ldots x_n$, where each $x_i \in F_q$.

q-ary block code (or code) : a nonempty set C of q-ary words all of the *same* length n.

– $q = 2$: binary code

codeword: an element of a code C

A CODE IS A SET OF WORDS, CALLED CODEWORDS.

length of a code: n, the (common) length of the codewords in C.

size of a code C, denoted $M = |C|$: the number of codewords in C

(n, M)-code: length n and size M

Examples: $q = 2$:

2-repetition code $(n, M) = (2, 2)$

$\{aa, bb\}$ $(n, M) = (2, 2)$

3-repetition code $(n, M) = (3, 2)$

n-repetition code $(n, M) = (n, 2)$

$$C_1 = \{00, 01, 10, 11\},\ (n, M) = (2, 4)$$
$$C_2 = \{000, 011, 101, 110\}\ (n, M) = (3, 4)$$
$$C_3 = \{00000, 01101, 10110, 11011\}(n, M) = (5, 4)$$

$q = 3$ (ternary):

$$C_4 = \{000000, 111111, 222222\}\ (n, M) = (6, 3)$$

Note: in this class, we will *not* consider variable length codes

We need a way to measure distance between words of the same length.

Defn *Hamming distance:* for words $\overline{x} = x_1 \dots x_n, \overline{y} = y_1 \dots y_n$ of the same length,

$$d(\overline{x}, \overline{y}) = |\{1 \leq i \leq n : x_i \neq y_i\}|$$

i.e., $d(\overline{x}, \overline{y})$ is the number of of places at which x and y differ.

– Example: $d(01101, 10011) = 4$, $d(12345, 14134) = 4$;

– Note: magnitude of difference is irrelevant.

Properties of Hamming distance:

Proposition: $d(\overline{x}, \overline{y})$ is a metric, i,e.,

(0) $d(\overline{x}, \overline{y}) \geq 0$

(1) $d(\overline{x}, \overline{y}) = 0$ iff $\overline{x} = \overline{y}$

(2) $d(\overline{x}, \overline{y}) = d(\overline{y}, \overline{x})$

(3) $d(\overline{x}, \overline{z}) \leq d(\overline{x}, \overline{y}) + d(\overline{y}, \overline{z})$ (triangle inequality)

Proof: 0, 1 and 2 are obvious.

Next lecture: proof of 3.