Problem Set 2

Your answers will be graded by actual human beings (at least that's what we believe!), so don't limit your answers to machine-gradable responses. Some of the questions specifically ask for explanations; regardless, it's **always a good idea to provide a short explanation for your answer**.

Before doing this PSet, please read Chapters 5 and 6 of the readings. Also attempt the practice problems on this material; these practice problems include both linear block codes and convolutional codes; you may ignore the latter for now.

Please also note that solving the two Python programming tasks in this PSet will be useful in the next PSet, which will require you to compare these block codes with convolutional codes, which you will learn next week.

Due dates are as mentioned above. Checkoff interviews for PS2 and PS3 will be held together and will happen between October 4 and 8.

Problem 1: Properties of linear block codes

For the nine statements given below, consider an (n, k, d) linear block code in systematic form. Fill in the blanks by writing the missing parts for each statement in the space provided below.

a. If the code can correct all single-bit errors, then the number of parity bits **must** be at least _____.
b. Suppose we transmit codewords over a binary symmetric channel with non-zero error probability. Then, the total number of possible **invalid** received words (i.e., those not equal to a valid codeword) is _____.
c. The weight of a codeword is the number of ones in it. The weight of each non-zero codeword is at least _____.
d. If the code can correct all single-bit errors, but not all 2-bit errors, then the possible value(s) of d is/are _____.
e. If d is odd, then this code can be transformed into a (n+1, k, d+1) code by _____.
f. Suppose two codewords c1 and c2 from this code have a Hamming distance of 4. If an even parity bit (i.e., a bit equal to the sum of the other bits) is appended to each codeword to produce two new codewords, c1' and c2', the Hamming distance between c1' and c2' is _____.

g. The generator matrix for the code has _____ rows and _____ columns.
h. The parity check matrix for the code has _____ rows and _____ columns.
i. For a single-error correction code with parity check matrix H, each of the n syndromes corresponding to an error case is obtained by calculating H $\mathbf{v}^T$. Here, **v** is a _____ x _____ matrix with _____ zeroes and _____ ones.
j. With maximum-likelihood decoding, the syndrome 00...0 (all zeroes) corresponds to the case when there are _____ errors in the codeword.

(points: 5)

Problem 2: Decoding error probability

A binary symmetric channel has bit-flip probability e. Suppose we take a stream of S bits in which zeroes and ones occur with equal probability, divide it into blocks of k bits each, and apply an (n,k,3) code to each block. To correct bit errors, we decode each block of n bits at the receiver using maximum-likelihood decoding. Your goal in this problem is to obtain a lower bound on the probability that the stream of S bits is decoded correctly. To this end, you may assume that any n-bit block with two or more errors will not be decoded correctly. You may also assume that S is an integral multiple of k. Show your work.

(points: 2)

Problem 3. Parity Inc.'s Code

Parity Inc. has developed a simple linear block code with three message bits, D1, D2, and D3, and three parity bits, P1, P2, and P3. The parity equations are:

```
P1 = D1 + D2
P2 = D1      + D3
P3 = D1 + D2 + D3
```

All additions are modulo 2, as usual. Parity Inc. has implemented a maximum-likelihood decoder that is able to correct a certain number of bit errors with this code. Assume that the decoder returns the most-likely message sequence if the number of bit errors is no larger than this correctable number, and returns "uncorrectable errors" otherwise. The sender transmits coded bits in the order D1 D2 D3 P1 P2 P3.

A. What is the minimum Hamming distance of Parity Inc.'s code? Explain your answer.

(points: 1)

B. If the receiver gets the word 111110, what will the decoder return?

(points: 1)

C. If the receiver gets the word 110100, what will the decoder return?

(points: 1)

Problem 4. Angry Coders (1 points)

Angry Coders, Inc., is hiring smart hackers to join their team. You apply for a position. Your interviewer asserts that their product uses a (19, 12, 5) linear block code that corrects all combinations of 0-, 1-, and 2-bit errors. He asks you to construct such a code. How would you respond to his question?

(points: 1)

Problem 5: Rectangular parity

Consider the following variant of the rectangular parity code with r rows and c columns. Each row has a row parity bit. Each column has a column parity bit. In addition, **each codeword has an overall parity bit**, to ensure that the number of ones in each codeword is even.

A. What is the rate of the code?

(points: 0.5)

B. Ben Bitdiddle takes each codeword of this code and *removes* the first row parity bit from each codeword. Note that the overall parity bit is calculated by **including** the row parity bit he eliminated. Ben then transmits these codewords over a noisy channel. What is the largest number of bit errors in a codeword that this new code can always **correct**? Explain.

(points: 1)

Problem 6: The Matrix Reloaded

Neo receives a 7-bit string, D1 D2 D3 D4 P1 P2 P3 from Morpheus, sent using a code, C, with parity equations

```
P1 = D1 + D2 + D3
P2 = D1 + D2 + D4
P3 = D1 + D3 + D4
```

A. Write down the generator matrix, G, for C.

(points: 0.5)

B. Write down the parity check matrix, H, for C.

(points: 0.5)

C. If Neo receives 1 0 0 0 0 1 0 and does maximum-likelihood decoding on it, what would his estimate of the data transmission D1 D2 D3 D4 from Morpheus be? For your convenience, the syndrome si corresponding to data bit Di being wrong are given below, for i=1, 2, 3, 4:

$s1 = (1\ 1\ 1)^T$, $s2 = (1\ 1\ 0)^T$, $s3 = (1\ 0\ 1)^T$, $s4 = (0\ 1\ 1)^T$.

(points: 0.5)

On Trinity's advice, Morpheus decides to augment each codeword in C with an **overall parity bit**, so that each codeword has an even number of ones. Call the resulting code C^+.

D. Write down the generator matrix, G^+, of code C^+. Express your answer as a concatenation (or stacking) of G (the generator for code C) and another matrix (which you should specify). Explain your answer.

(points: 1)

E. Morpheus would like to use a code that corrects all patterns of 2 or fewer bit errors in each codeword, by adding an appropriate number of parity bits to the data bits D1 D2 D3 D4. He comes up with a code, C^{++}, which adds 5 parity bits to the data bits to

produce the required codewords. Explain whether or not C^{++} will meet Neo's error correction goal.

(points: 1)

Python programming tasks

Zip archive of all required files for the programming tasks on this PS. Extract using unzip.

Python Task #1: Rectangular parity SEC code (6 points)

Useful download link:

PS2_rectparity.py -- template file for this task

The intent in this task is to develop a decoder for the rectangular parity single error correction (SEC) code using the structure of the rectangular parity code to "triangulate" the location of the error, if any. Your job is to take a received codeword which consists of a data block organized into `nrows` rows and `ncols` columns, along with even parity bits for each row and column. The codeword is represented as a binary sequence (i.e., a list of 0's and 1's) in the following order:

```
[D(0,0), D(0,1), ..., D(0,ncols-1),        # data bits, row 0
 D(1,0), D(1,1), ...,                      # data bits, row 1
 ...
 D(nrows-1,0),   ..., D(nrows-1,ncols-1),  # data bits, last row
 R(0), ..., R(nrows-1),                    # row parity bits
 C(0), ..., C(ncols-1)]                    # column parity bits
```

in other words, all the data bits in row 0 (column 0 first), followed all the data bits in row 1, ..., followed by the row parity bits, followed by the column parity bits. The parity bits are chosen so that all the bits in any row or column (data and parity bits) will have an even number of 1's.

Define a Python function `rect_parity(`*`codeword`*`, `*`nrows`*`, `*`ncols`*`)` as follows:

`message_sequence` `= rect_parity(`*`codeword`*`,`*`nrows`*`,`*`ncols`*`)`

`codeword` is a binary sequence of length `nrows*ncols + nrows + ncols` whose elements are in the order described above.

The returned value *`message_sequence`* should have `nrows*ncols` binary elements consisting of the corrected data bits D(0,0), ..., D(nrows-1,ncols-1). If no correction is necessary, or if an uncorrectable error is detected, then return the raw data bits as they

appeared in the codeword.

Do not use syndrome decoding in this task. (That will be the next programming task.) This goal of this task is to use the structure of the parity computations to pinpoint error locations. (This method is much faster than syndrome decoding, but is far less general and varies from code to code.)

`PS2_tests.even_parity(seq)` is a function that takes a binary sequence *`seq`* and returns True if the sequence contains an even number of 1's, otherwise it returns False. This parity check will be useful when performing the parity computations necessary to do error correction. PS2_rectparity.py is a template that you will extend by writing the function `rect_parity`.

The `PS2_tests.test_correct_errors` function will try a variety of test codewords and check for the correct results. If it finds an error, it'll tell you which codeword failed; if your code is working, it'll print out

```
Testing all 2**n = 256 valid codewords
...passed
Testing all possible single-bit errors
...passed
(8,4) rectangular parity code successfully passed all 0,1 and 2 bit error tests
```

When you're ready, please submit the file with your code using the field below.

True Browse...

(points: 6)

Python Task #2: Syndrome Decoding of Linear Block Codes

Useful download link:

PS2_syndecode.py -- template file for this task

Write `syndrome_decode`, a function that takes the four arguments given below and returns an array of bits corresponding to the most likely transmitted message:

```
syndrome_decode(codeword, n, k, G)
```

`codeword` is a numpy array of bits received at the decoder. The array has size `n`, which we explicitly specify as an argument to `syndrome_decode` for clarity. The original message is `k` bits long. `G` is the **generator matrix** of the code. See Chapter 6 for what the generator matrix means and the syndrome decoding procedure.

You may assume that `G` is a code that can correct all combinations of single bit errors. So your syndrome decoder needs to handle only that case. Of course, you may feel free to be ambitious and handle up to `t` bit errors, but our testing won't exercise those cases.

The following implementation notes may be useful:

1. Syndrome decoding uses matrix operations. You will very likely find it convenient to use numpy's `matrix` module, which is described fairly succinctly in this tutorial. The table on this page titled "Linear Algebra Equivalents" is a useful resource.
2. All arithmetic should be modulo 2, in F_2. When you multiply two matrices whose elements are all 0 or 1, you may get numbers different from 0 or 1. Of course, one needs to replace each such number with its modulo-2 value. We have given you a function, `mod2(A)`, which does that for an integer matrix, `A`.
3. We have also given you a simple function, `equal(a, b)`, which returns True iff two matrices a and b are equal element-by-element.
4. In this task, we are not expecting you to pre-compute the syndromes for the code, but to compute the syndromes in `syndrome_decode` just before decoding a codeword. It would be more efficient to pre-compute syndromes, but it would also mean that we need to specify the interface for a function like `compute_syndromes`, and for your software to adhere to that interface. In the interest of simplifying the interfaces and specifying only the behavior of `syndrome_decode`, we will take the small performance hit and just have you compute the syndromes every time you decode a codeword.

Testing: We will test your code with a few different generator matrices. These tests will tell you whether your code works on that input; if not, it will print out the input on which the decoder failed (printing out what it produced and what was expected) and exit. We test all single-bit error patterns and the no-error case over the `n`-bit codeword.

A successful test prints out lines similar to these:

```
Testing all 2**k = 16 valid codewords
...passed
Testing all n*2**k = 112 single-bit error codewords
...passed
All 0 and 1 error tests passed for (7,4,3) code with generator matrix G =
[[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]]
```

A failed test might print out lines similar to these:

```
Testing all 2**k = 16 valid codewords
...passed
Testing all n*2**k = 112 single-bit error codewords
OOPS: Error decoding [1 0 0 0 0 0 0] ...expected [0 0 0 0] got [1 0 0 0]
```

When you're ready, please submit the file with your code using the field below.

True Browse...

(points: 9)

6.02 Introduction to EECS II: Digital Communication Systems

Fall 2012