Assignment: Linear Codes, Due Wednesday 11/30, 11:59pm

Chris Betsill

November 30, 2016

1. Decode using a coset leader syndrome table

The following matrix is a generator matrix for a linear binary code.

```
bch15 <- matrix(c(1,0,0,0,0,1,1,1,0,1,1,0,0,1,0,
                  0,1,0,0,0,0,1,1,1,0,1,1,0,0,1,
                  0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,
                  0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,
                  0,0,0,0,1,1,1,0,1,1,0,0,1,0,1),
                byrow=TRUE, nrow=5)
```

Answer the following questions, showing the mathematics or calculations that you used to determine your answers.

a. How many codewords are there in this code?

```
C <- generateCode(bch15)
nrow(C)
```

```
[1] 32
```

b. How many errors can this code correct?

It can correct up to 3 errors. This is equivalent to one less than the minimum distance between codes, divided by two.

```
codeDistance(C)
```

```
[1] 7
```

c. How many different syndromes are there?

```
length(cosetLeaderSyndromeTable(bch15))
```

```
[1] 1024
```

d. Bob receives the following message. Is it a codeword? If not, decode it to the nearest codeword. How many errors were corrected?

```
mess <- c(1,0,1,1,0,1,0,0,0,1,0,0,1,1,1)
cs <- syndrome(mess, parityCheckMatrix(bch15))
clst <- cosetLeaderSyndromeTable(bch15)
synds <- names(clst)
cos <- clst[[match(cs, synds)]]
code <- (cos+mess) %% 2

code
```

```
 [1] 1 0 1 1 1 1 0 0 0 1 0 0 1 1 0
```

```
hammingDistance(mess, code)
```

```
[1] 2
```

2. Decode the Hamming $[63, 57]$ code

The following commands compute the parity check matrix for the Hamming $[63, 57]$ code.

```
m <- 6
powersOf2 <- 2^(0:(m-1))
ham63H <- sapply(c(setdiff(1:(2^m-1), powersOf2), powersOf2),
                 function(x){as.numeric(intToBits(x)[1:m])})
```

Decode the following message (without computing the entire coset leader syndrome table). In which position was the error? Show your work.

```
mess <- c(1,1,1,0,0,0,0,0,1,1,0,0,1,0,1,0,1,1,1,0,0,1,0,1,0,0,0,0,0,1,0,1,
          1,1,0,1,0,1,1,1,0,1,1,0,1,0,1,0,0,0,1,1,0,1,0,0,1,1,1,0,1,0,0)

tM <- t(ham63H)
synd <- strsplit(syndrome(mess, ham63H), ", ")[[1]]
for(i in 1:length(tM)){
  if(all(tM[i,] == synd)){
    print(i)
    mess[i] <- (mess[i]+1) %%2
    return(mess)
  }

}
```

```
[1] 25
```

```
 [1] 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 0
[36] 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 0 0 1 1 1 0 1 0 0
```

3. Prove the lemma

Use the fact that $vH^T = \mathbf{0}$ for any codeword v to prove both parts of the following lemma.

Lemma. Let C be a sytematic linear binary code, and let H be its parity check matrix.

1. If C contains a codeword of weight 1, then H contains a zero column.
2. If C contains a codeword of weight 2, then H has two identical columns.

Since C contains a codeword of weight 1,(such as v = 0,0,0,1), and $vH^T = \mathbf{0}$, you know that all but one vector in the resulting matrix will be zero (since the zero's in v nullify the rows in H^T), there must be at least one row (in this case the last) that is all zero's to allow the $vH^T = \mathbf{0}$ (by matrix multiplication). When transposed again, that zero row becomes a coloumn of zero's in H.

Similarly if If C contains a codeword v of weight 2, all but two rows in H^T will be nullified by the zero's in v, forcing the two non-zero rows to be equal (since they will cancel when added), if $vH^T = \mathbf{0}$. These identical rows in H^T become identical column in H.

4. Build a Golay decoder

The Golay code G_{24} has the following generator matrix.

```
golay24 <- matrix(c(1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,1,1,0,0,0,1,0,
                    0,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,1,1,1,0,0,0,1,
                    0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,1,1,1,0,0,0,
                    0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,1,1,0,1,1,1,0,0,
                    0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,1,0,1,1,0,1,1,1,0,
                    0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,1,1,0,1,1,1,
                    0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,1,1,
                    0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,0,0,0,1,0,1,1,0,1,
                    0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,0,0,0,1,0,1,1,0,
                    0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,1,1,0,0,0,1,0,1,1,
                    0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,1,1,1,0,0,0,1,0,1,
                    0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1),
                  byrow=TRUE, nrow=12)
```

a. Use the fact that this code is a linear code to compute its minimum distance. How many errors can this code correct? (Show your work. You probably want to avoid using your `codeDistance` function.)

```
C <- generateCode(golay24)
min <- ncol(C)
for(i in 1:nrow(C)){
  weight <- sum(C[i, ])
  if(weight < min && weight != 0)
    min <- weight
}
print(min)
```

```
[1] 8
```

The minimum non-zero weight of a code word is 8, so its minimum distance is also 8, and thus it can correct up to 3 errors.

b. Use the following code block to compute the coset leader syndrome table for this code. It may take a few minutes, so the following code block is set not to evaluate when this file is knitted. The second line of this code block saves the result to a file so you don't have to do this computation again, in case you

need it for a future assignment. Change the path and name of the file `~/Downloads/filename.RData` to something appropriate. How much memory does this table occupy in the R environment? (Look in the environment tab.) How big is the stored file?

```
golayTable <- cosetLeaderSyndromeTable(golay24)
save(golayTable, file="~/Downloads/golayTable.RData")
```

The stored file is 1.3 Mb

c. Decode the following message. How many errors were corrected?

```
mess <- c(1,1,1,0,0,0,1,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,1,1)
code <- cosetLeader(mess, generateCode(golay24))+mess
code
```

```
 [1] 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1
```

```
errors <- hammingDistance(code, mess)
errors
```

```
[1] 2
```