Simulation Results of Channel Capacity using Parity Bits for Error Correction

Austin Minor

November 10, 2016

1 Introduction

Claude Shannon was a major figure in creating the mathematical discipline of information theory. This paper analyizes how emperical results line up with the statements made by this mathematical model of information. More formally, we will be analyizing the effect of using parity bits for error correction and how well they are at approaching the formal limit on the conveyance of data.

2 Mathematical Description

The official description of converyable information is the mutual information between to system of events. Let A be the source alphabet. Let B be the reception alphabet. Let Q be the transmission matrix (IE the probability of receiving $b_j \in B$ given a transmission of $a_i \in A$. Let these entries be represented as $q_{i,j}$ respectively. For our example we will only consider the simplier channel known as the binary symettric channel. Thus

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} \\ q_{0,1} & q_{1,1} \end{pmatrix} = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$$

Given such a channel, the probability of bit errors can be represented by a Bernoulli Trial.

The mutual information between two system of events (definition given elsewhere) is defined as following:

$$I(A,B) = \sum_{i=1}^{m} \sum_{j=1}^{n} Pr(A_i \cup B_j) log(\frac{Pr(A_i \cup B_j)}{Pr(A_i)Pr(B_j)}) = \sum_{i=1}^{m} Pr(A_i) \sum_{j=1}^{n} q_{i,j} log(\frac{q_{i,j}}{\sum_{t=1}^{m} Pr(A_t) q_{t,j}})$$

It has been proved elsewhere that the maximal channel capacity (maximum I(A,B) based on $Pr(A_i)$– probability that a source letter is transmitted) for a BSC is when $Pr(A_1 = 0) = Pr(A_2 = 1) = \frac{1}{2}$. Furthermore,

$$I(A,B) = p * log(2 * p) + (1-p) * log(2 * (1-p))$$

For example, if $p = .99$ then $I(A,B) = .919$.

3 Problem Statement

To test this model of channel capacity and mutual information, a Matlab program was written that would satisfy the important requirements listed above ($Pr(A_i) = \frac{1}{2}$, Q as described above). The Matlab code is attached below.

```
PRECISION = 1000;
RETRY_COUNT = 100;

%first avg transmission
%second error rate of transmission
A = zeros(2, PRECISION);

temp = 0;
for i = 1:PRECISION
  for j = 1:RETRY_COUNT
    temp = transmit(generate_random_msg(10,4), i/PRECISION, 20);
    if temp == -1
      A(2,i) = A(2,i) + 1;
    else
      A(1,i) = A(1,i) + temp;
    end
  end
  %compute avg ignoring failed transmissions
  temp = RETRY_COUNT - A(2,i);
  if temp == 0
  %do nothing because A(1,i) will be zero also
  else
    A(1,i) = A(1,i)/(RETRY_COUNT-A(2,i));
  end
end

figure
%bits of info needed for transmission
subplot(1,2,1);
plot(A(1,:))
%number of errors in transmission
subplot(1,2,2);
plot(A(2,:))
```

Listing 1: Main Simulator

```
function bit_count = transmit(msg, err_prob, retry_lim)
  msg = insert_parity_bit(msg);

  num_transmissions = 0;
  transmitted_msg = 0;
  for i = 1:size(msg)(1)
    transmitted_msg = transmit_msg(msg(i,:),err_prob);
    %check rough equality with parity bit
    %guarentees proper transmission for one or less errors
    temp_c = 0;
    while transmitted_msg(1) != parity(transmitted_msg(2:end)) && temp_c < retry_lim
      num_transmissions = num_transmissions + 1;
      temp_c = temp_c + 1;
      transmitted_msg = transmit_msg(msg(i,:),err_prob);
    end
    %msg would fail using given parity scheme and error probality
    if isequal(transmitted_msg, msg(i,:)) == false
      bit_count = -1;
      return
    end
    %always one transmission
    num_transmissions = num_transmissions + 1;
  end

  %num trans multiplied by the size of packet transmitted
  bit_count = num_transmissions * size(msg)(2);
end
```

Listing 2: Transmission Simulator

```
function msg = transmit_msg(msg_in, err_prob)
  msg = zeros(size(msg_in));
  rnd_num = 0;
  for i = 1:size(msg_in)(2)
    rnd_num = rand();

    if rnd_num < err_prob
      if msg_in(i) == 0
        msg(i) = 1;
      else
        msg(i) = 0;
      end
    else
      msg(i) = msg_in(i);
    end
  end
end
```

Listing 3: Per-packet Transmission Simulator

```
function msg = insert_parity_bit(msg_in)
  c = size(msg_in)(1);
  msg = zeros(size(msg_in)(1),size(msg_in)(2) + 1);
  for i = 1:c
    %https://www.mathworks.com/matlabcentral/newsreader/view_thread/331396
    msg(i,:) = [parity(msg_in(i,:)) msg_in(i,:)];
  end
end
```

Listing 4: Per-packet Parity Bit Inserter

```
function par = parity(packet)
  num_ones = 0;
  for i = 1:size(packet)(2)
    if packet(i) == 1
      num_ones = num_ones + 1;
    end
  end
  if (mod(num_ones,2) == 0)
    par = 0;
  else
    par = 1;
  end
end
```

Listing 5: Parity Bit Message Inserter

```
function msg = generate_random_msg(num_packets, packet_len)
  msg = zeros(num_packets,packet_len);
  for i = 1:num_packets
    for j = 1:packet_len
      msg(i,j) = floor(2*rand());
    end
  end
end
```

Listing 6: Generator of Random Messages with Packets

4 Results

5 Analysis