**Stochastic Processes**

Coursework Project Report

Spring 2020

Name: Xin Yi (易鑫)

Student ID: 1950271009

Name: Cheng Wen (文成)

Student ID: 1950271008

1. **Introduction**

In radio channel or optical channel, noise can cause transmission errors, that can be modeled as a stochastic process. However, real-life experiments show that bit errors often appear in bursts. The Gilbert–Elliott model [1] is a simple channel model introduced by Edgar Gilbert and E. O. Elliott widely used for describing burst error patterns in transmission channels, that enables simulations of the digital error performance of communications links.

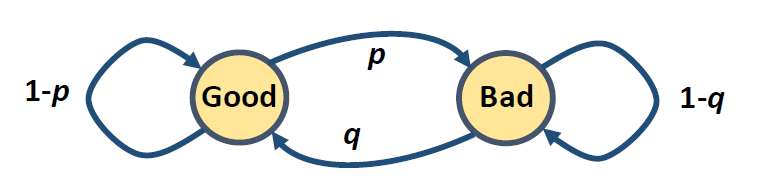
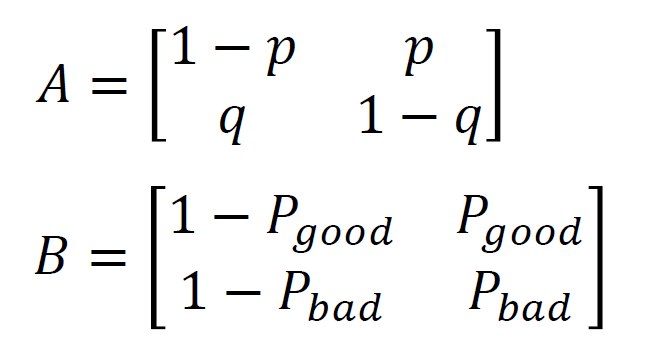
 

Figure 1. General Illustration of Gilbert-Elliot Model

In our research scenario, we aim to answer two research question.

**Research Question 1**: Given the bit error sequence, how to find out the most likely parameters for this Gilbert-Elliot model?

Transmission error in a radio channel appears when a transmitted symbol is received incorrectly. Digital information is usually processed and transmitted as bits; therefore, we can say that there is a bit error, if transmitted bit 0 is received as 1, or if transmitted bit 1 is received as 0. It has been observed that when digital information is transmitted over a radio channel, transmission errors usually appear in bursts, rather than isolated from each other. Therefore, bit errors are often simulated by using Gilbert-Elliot model [1]. A general illustration of Gilbert-Elliot model is shown in Figure 1. More specifically, the Gilbert–Elliott model is based on a Markov chain with two with two states Good (for good or gap) and Bad (for bad or burst). In Good state, bit error probability is low; in Bad state, bit error probability is high. We can define that the observed state is 0 if there is no bit error, and the observed state is 1, if there is a bit error. Transition between Good state and Bad state can be expressed by parameters *p* and *q*, using the transition matrix A in Figure 1. In the first part of the project, our research question is to find the most likely parameters λ = {p, q, , } for Gilbert-Elliot model producing the bit error sequence included in text file “*biterrors.txt*”, which is in the form of "00000000...1...00000..."

**Research Question 2**: How to perform simulation of bit errors by using Gilbert-Elliot model?

In the second part of the project, our research question is to implement a radio channel simulator, using a Gilbert-Elliot model to simulate bit errors. We use the scenario illustrated in Figure 2 as an inspiration for our study. We generate a sequence of bits randomly, apply simulated bit errors to the bitstream, and then compare the transmitted and received bitstreams to compute bit error rate , defined as a ratio of erroneously received bits , and the total number of bits : .

Using the channel simulator, we have conducted a small-scale research study, analyzing the impact of Gilbert-Elliot model parameters for transmission performance. We ran the simulation with different Gilbert-Elliot model parameters and compare the experimentally achieved bit error rate results against analytically derived results for the respective parameters. We also implement encode FEC and decode FEC, including non-encoding, redundancy encoding and hamming encoding. Furthermore, we study the impact of encoding/decoding on the residual bit error rate.

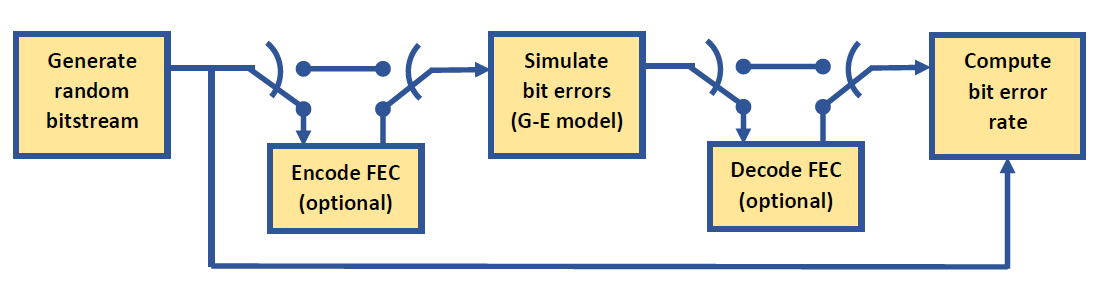


Figure 2. Example Bit Error Simulation Scenario

In summary, this article makes the following significances:

* + We summarized the Baum-Welch algorithm [3] and try to use Baum-Welch algorithm to find out the most likely parameters in Gilbert-Elliot model for the given bit error sequence. We also perform estimation based on statistical probability to validate the result.
  + We have designed and implement a channel simulator with and without the encoding/decoding. Our encoding/decoding method includes non-encoding/decoding, redundancy encoding/decoding, and Hamming encoding/decoding. We have conducted a small-scale research study on the impact of encoding/decoding method on the Bit Error Rate (BER).
  + We implement the prototype codes (in Matlab), of which source code are available via a publicly accessible repository: <https://github.com/wcventure/StochasticCoursework>.
  + We provide compelling results and the analysis for each experiment.

The remainder of this paper is organized as follows. We describe the detail of our methodology in Section 2, followed by the experimental results and analysis in Section 3. Section 4 declares the contribution of each author. Section 5 concludes the paper.

1. **Methods**

This section mainly contains two parts. In the first part, we try to use One of the Expectation-Maximization (EM) algorithm (i.e. Baum-Welch Algorithm) to learn the parameters of an HMM. We borrow the concept and original idea of these two algorithms from [2][3]. Moreover, we further perform estimation based on statistical probability to validate the result. In the second part, we have conducted a small-scale research study of performing simulations of bit errors, by using the channel simulator. We have not only implemented a Gilbert-Elliot model, but also three encoding/decoding method.

* 1. **Learning the parameters of an HMM (RQ1)**

We describe the maximum-likelihood parameter estimation problem and how the Baum-Welch Algorithm can be used for its solution. As we know, an HMM describes the joint probability of a collection of "hidden" and observed discrete random variables. It relies on the assumption that the  hidden variable given the hidden variable is independent of previous hidden variables, and the current observation variables depend only on the current hidden state. In an especial HMM, Gilbert-Elliot Model, there exist two matrixes (i.e. A and B in Figure 1). However, what if parameters λ = {p, q, , } is not known, and we want to find it out? We want to find out the maximum likelihood estimate of the model parameters λ given some training data. In the following, we derive the Baum-Welch algorithm for finding the maximum-likelihood estimate of the parameters of hidden Markov model given a set of observed bit error sequence.

* + 1. **Baum-Welch Algorithm**

The Baum-Welch algorithm uses the well-known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors. Let be a discrete hidden random variable with states in total. We assume the is independent of time , which leads to the definition of the time-independent stochastic transition matrix . The initial state distribution (i.e. when ) is given by . The observation variables can take one of possible values. We also assume the observation given the “hidden” state is time independent. The probability of a certain observation at time for state is given by . Taking into account all the possible values of and , we obtain the matrix where belongs to all the possible states and belongs to all the observations. An observation sequence is given by . Thus, we can describe a hidden Markov chain by . The Baum-Welch algorithm finds a local maximum for  (i.e. the HMM parameters  that maximize the probability of the observation) [2].

The Baum-Welch algorithm can be seemed as 6 steps.

1. **Initialize , and .** Set with random initial conditions. They can also be set using prior information about the parameters if it is available; this can speed up the algorithm and also steer it toward the desired local maximum.
2. **Forward procedure.** Let , the probability of seeing the observations and being in state at time . This is found recursively: 1. , 2. . Since this series converges exponentially to zero, the algorithm will numerically underflow for longer sequences [[4]](https://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm#cite_note-6). However, this can be avoided in a slightly modified algorithm by scaling  in the forward and  in the backward procedure below.
3. **Backward procedure.** Let that is the probability of the ending partial sequence given starting state at time . We calculate as, 1. , 2. .
4. **Compute temporary variables and .**

,

Which is the probability of being in state at time given the observed sequence and the parameters

, which is the probability of being in state and at times and respectively given the observed sequence Y and parameters . The denominators of and are the same; they represent the probability of making the observation given the parameters .

1. **Update HMM parameters , and .** The parameters of the hidden Markov model can be updated: , which is the expected frequency spent in state at time 1. , which is the expected number of transitions from state  to state  compared to the expected total number of transitions away from state . To clarify, the number of transitions away from state  does not mean transitions to a different state , but to any state including itself. This is equivalent to the number of times state  is observed in the sequence from to . , where is an indicator function, and is the expected number of times the output observations have been equal to  while in state  over the expected total number of times in state . These steps are now repeated iteratively until a desired level of convergence.
2. **Repeat from step 2) until desired level of convergence is reached.**

**2.1.2 Estimation Based on Statistical Probability**

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable. The point in the parameter space that maximizes the likelihood function is called the maximum likelihood estimate [5].

Let  be observations from  independent and identically distributed random variables drawn from a Probability Distribution ​, where ​​ is known to be from a family of distributions  that depend on some parameters . For example, ​ could be known to be from the family of normal distributions , which depend on (standard deviation) and  (mean), and    would be observations from ​.

The goal of MLE is to maximize the likelihood function:

Often, the *average log-likelihood* function is easier to work with:

* 1. **Simulation of Bit Errors with encoding/decoding (RQ2)**

Generalized block codes with n transmitted symbols (e.g., bits), k source symbols and n-k redundant symbols, can be used both for error detection (and error correction). Different encoding/decoding methods have been developed. Our implementation includes non-encoding/decoding, redundancy encoding/decoding, hamming encoding/decoding.

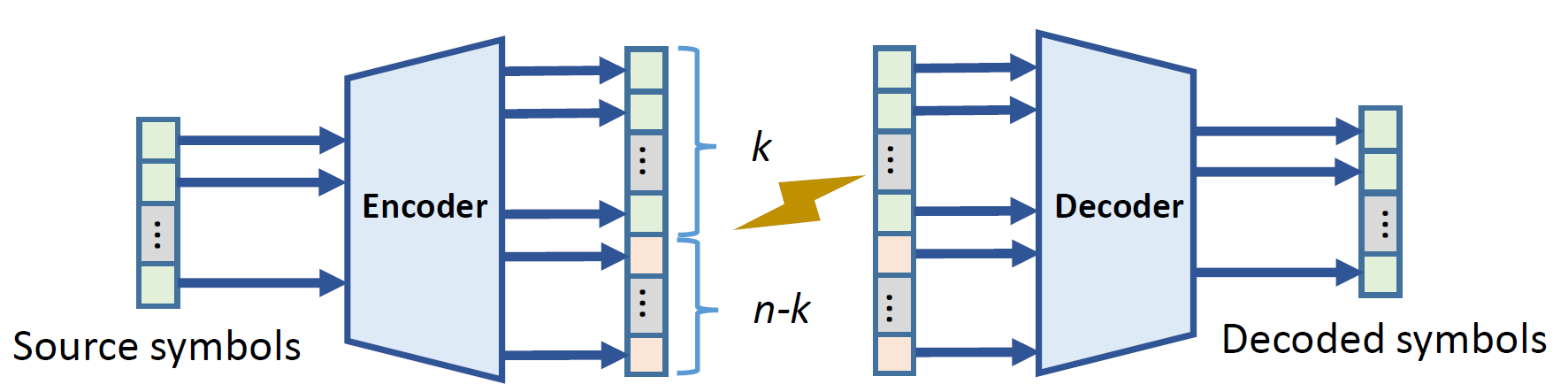


Figure. 3 The Workflow of Encoder and Decoder

* + 1. **Non-encoding/decoding**

We simulate the bit errors by using Gilbert-Elliot model learned. As discuss above, the most important parameter is λ = {p, q, , }. So, we can easily assign different value of λ = {p, q, , } to generate bit sequence for simulation. We will not discuss more here.

* + 1. **Repetition Encoding/Decoding**

The repetition code is one of the most basic error-correcting codes. In order to transmit a message over a noisy channel that may corrupt the transmission in a few places, the idea of the repetition code is to just repeat the message several times. The hope is that the channel corrupts only a minority of these repetitions. This way the receiver will notice that a transmission error occurred since the received data stream is not the repetition of a single message, and moreover, the receiver can recover the original message by looking at the received message in the data stream that occurs most often.

In the case of a binary repetition code, there exist two code words – all ones and all zeros – which have a length of n. The encoder takes as input the data sequence of length and then produces repetition bits. The decoder takes as input the data of length and then produces N bits that are equal to the original data sequence as soon as possible. This gives the repetition code an error correcting capacity of (i.e. it will correct up to errors in any code word). If the length of a binary repetition code is odd, then it’s a perfect code [7]. Moreover, the binary repetition code of length is equivalent to the in Hamming code, which will further be explained in Section 2.2.3.

**2.2.3 Hamming Encoding/Decoding**

Hamming code is an error-correction code that can be used to detect single and double-bit errors. It can correct single-bit errors that can occur when binary data is transmitted. Hamming code development [6] is a very direct construction of a code that permits correcting single-bit errors. It assumes that the data to be transmitted consists of a certain number of information bits, and he adds to these a number of check bits such that if a block is received that has almost one bit in error, then identifies the bit that is in error (which may be one of the check bits). Specifically, in Hamming code is interpreted as an integer which is 0 if no error occurred, and otherwise is the 1-origined index of the bit that is in error. Let be the number of information bits, and the number of check bits used. Because they check bits must check themselves as well as the information bits, the value of , interpreted as an integer, must range from 0 to which is distinct values. Because bits can distinguish cases, we must have , where is the number of bits entered, are parity bits. This is known as the Hamming rule. For example, Hamming Code (7, 4) will take 4 input bits and 3 parity bits are added as redundant bits so the signal transmitted is 7 bits.

1. **Experimental Results and Analysis**

In this section, we explain the experimental results and analysis, including numerical results, graphs and discussion, for each method proposed in Section 2.

* 1. **Evaluation of Learning the parameters of an HMM (RQ1)**

We use the text file “*biterrors.txt*”, which is in the form of "00000000...1...00000...", as the observation sequence. Firstly, we use Baum-Welch algorithm to learn the most likely parameters λ = {p, q, , } for Gilbert-Elliot model producing such bit error sequence. Then, we perform estimation for finding the probability of occurrence of 0 and 1 based on statistical probability.

* + 1. **Using Baum-Welch Algorithm to Find the Parameters**

For simplicity, we can assume that there are only two states (i.e. Good and Bad) that determine whether the bit is 0 or 1. Now we don’t know the state at the initial starting point, we don’t know the transition probabilities between the two states and we don’t know the probability that the observed bit can be an error given a particular state. To start we first guess the transition and emission matrices).

|  |  |  |
| --- | --- | --- |
| **Transition** | | |
|  |  |  |
|  | 0.8 | 0.2 |
|  | 0.5 | 0.5 |
| **Emission** | | |
|  | **0** | **1** |
|  | 0.5 | 0.5 |
|  | 0.5 | 0.5 |
| **Initial** | | |
|  | 0.8 | |
|  | 0.2 | |

We then take a set of observations: 0, 0, 0, 0, …, 1, 1, 0, 1, 0, 0, …

This gives us a set of observed transitions between bits: 00, 00, 00, …, 11, 10, 01, 10, 00, …

The next step is to estimate a new transition matrix. For example, the probability of the sequence 00 and the state being then is given by the following, P()\*P(0| )\*P()\*P(0 | ).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Estimate a New Transition Matrix** | | | | | | |
| Observed sequence | Probability of | Probability of | Probability of | Probability of | Highest Probability of observing that seq. | |
| 00 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 00 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 00 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| … | … | … | … | … | … | … |
| 11 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 10 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 01 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 10 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| 00 | 0.04=  0.8\*0.5\*0.2\*0.5 | 0.16=  0.8\*0.5\*0.8\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.025=  0.2\*0.5\*0.5\*0.5 | 0.16 |  |
| … | … | … | … | … | … | … |
| Total | 3.96 | 15.84 | 2.475 | 2.475 | 15.84 |  |

Thus, the new estimate for the to transition is now (referred to as “Pseudo probabilities” in the following tables). We then calculate the to , to and to transition probabilities and normalize so they add to 1. This gives us the updated transition matrix:

|  |  |  |
| --- | --- | --- |
| **Old Transition Matrix** | | |
|  |  |  |
|  | 0.8 | 0.2 |
|  | 0.5 | 0.5 |
| **New Transition Matrix**  **(Pseudo Probabilities)** | | |
|  |  |  |
|  | 1 | 0.2 |
|  | 0.15625 | 0.15625 |
| **New Transition Matrix**  **(After Normalization)** | | |
|  |  |  |
|  | 0.84 | 0.16 |
|  | 0.5 | 0.5 |

Next, we estimate a new emission matrix.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Estimate a New Emission Matrix** | | | | | | |
| Observed sequence | Highest Probability if 1 is assumed to come form | Highest Probability if 0 is assumed to come form | Highest Probability if 1 is assumed to come form | Highest Probability if 0 is assumed to come form | Highest Probability of observing that seq. | |
| 00 | / | 0.16 () | / | 0.025 () | 0.16 |  |
| 11 | 0.16 () | / | 0.025 () | / | 0.16 |  |
| 10 | 0.16 () | 0.16 () | 0.025  () | 0.04 () | 0.16 |  |
| 01 | 0.16 () | 0.16 () | 0.04  () | 0.025 () | 0.16 |  |
| Total | 0.48 | 0.48 |  |  | 0.48=0.16\*3 |  |

The new estimate for the 1 coming from emission is now . This allows us to calculate the emission matrix as described above in the algorithm, by adding up the probabilities for the respective observed sequences. We then repeat for it 0 came form and for if 1 and 0 came from and normalize.

|  |  |  |
| --- | --- | --- |
| **Old Emission Matrix** | | |
|  | **0** |  |
|  | 0.5 | 0.5 |
|  | 0.5 | 0.5 |
| **New Emission Matrix**  **(Estimates)** | | |
|  |  |  |
|  | 1 | 1 |
|  | 0.1875 | 0.1875 |
| **New Emission Matrix**  **(After Normalization)** | | |
|  |  |  |
|  | 0.5 | 0.5 |
|  | 0.5 | 0.5 |

Finally, we repeat these steps until the resulting probabilities converge satisfactorily. We get the final result, shown as follow.

|  |  |  |
| --- | --- | --- |
| **Transition** | | |
|  |  |  |
|  | 0.95255 | 0.04745 |
|  | 0.75265 | 0.24735 |
| **Emission** | | |
|  | **0** | **1** |
|  | 0.9901 | 0.0099 |
|  | 0.0103 | 0.9897 |

We have also tried to change the initial parameters λ = {p, q, , }, and compared the finial result. The finial result is convergent to above matrix.

* + 1. **Estimation Based on Statistical Probability Evaluation**

In this section, we consider directly the probability of occurrence of 0 and 1. Instead of consider the HMM, we consider it as an ordinary Markov model, which only contain two state (i.e. 0 and 1).

We have calculated the following probability.

P (0|0) =

P (1|1) =

P (1|0) =

P (0|1) =

Thus, we got the following transformation matrix, which is also convergent .

* 1. **Evaluation of simulating Gilbert-Elliot Model (RQ2)**

In this experiment, we implement different encoding/decoding methods to simulate the Gilbert-Elliot Model includes non-encoding/decoding, redundancy encoding/decoding, hamming encoding/decoding. We consider the influence of the transition probability *p* and *q*, that is, the value of the emission probability , is fixed. In the experiment we set . We observe a total of 100 cases for and in a 10000-bits sequence, and compared the difference between the experimental Bit Error Rate (BER). Moreover, we also compare against different encoding/decoding methods with same probability *p* and *q*.

* + 1. **Simulating Gilbert-Elliot Model Without Encoding/decoding**

We first implement a Gilbert-Elliot model without encoding/decoding, and use this model with fixed emission probability and to evaluate the influence of different transition probability *p* and *q*. Here is a plot that displays the distribution over 100 cases, as shown in Figure .4. The y-coordinate represents BER, and the x-coordinate represents different values of *p* and *q*. From the table, we can see that, given a pair of transition probability p and q. The corresponding BERs all concentrated in a certain interval.

C:\Users\kingsley\Desktop\Stochastic Process\homework\results\nulldistri.emf

Figure. 4 The distribution of BER in different probability *p* and *q* (non-encoding/decoding)

The Average BER for different probability *p* and *q* can be seen in Figure. 5. The y-coordinate represents BER, and the x-coordinate represents different values of *p*. And we use lines with different colors to represents different values of *q*. We have the following observations: (1) When the probability increases, BER gradually increases. This is because the channel is become easier to enter the Bad state, as probability increases. And the Bad state has a larger probability of transmission error, making the BER increase. (2) When the probability increases, BER gradually decreases. This may be because the channel is become easier to enter the Good state, as probability increases. And the Good state has a smaller probability of transmission error, making the BER decreases. (3) When the probability takes a smaller value and takes a larger value, we can get an optimal BER result since means that the channel is more likely to be stable in the Good state.

C:\Users\kingsley\Desktop\Stochastic Process\homework\output\null.emf

Figure. 5 The Average BER for different probability *p* and *q* (non-encoding/decoding)

* + 1. **Simulating Gilbert-Elliot Model with Redundancy Encoding/decoding**

We then implement a Gilbert-Elliot model with redundancy encoding/decoding. In particular, we implement a model with 3-redundancy encoding/decoding, and use this model with fixed emission probability and to evaluate the influence of different transition probability *p* and *q*. Here is a plot that displays the distribution over 100 cases, as shown in Figure .6. The y-coordinate represents BER, and the x-coordinate represents different values of *p* and *q*. From the table, we can see that, given a pair of transition probability p and q. The corresponding BERs all concentrated in a certain interval.

C:\Users\kingsley\Desktop\Stochastic Process\homework\results\redundistri.emf

Figure. 6 The distribution of BER in different *p* and *q* (3-redundancy encoding/decoding)

In the experiment, the Average BER for different probability *p* and *q* can be seen in Figure. 7. We can also get the following observations: (1) When the probability increases, BER gradually increases. This is because the channel is become easier to enter the Bad state, as probability increases. And the Bad state has a larger probability of transmission error, making the BER increase. (2) When the probability increases, BER gradually decreases. This may be because the channel is become easier to enter the Good state, as probability increases. And the Good state has a smaller probability of transmission error, making the BER decreases. (3) When the probability takes a smaller value and takes a larger value, we can get an optimal BER result since means that the channel is more likely to be stable in the Good state.

C:\Users\kingsley\Desktop\Stochastic Process\homework\output\Rund.emf

Figure. 7 The Average BER for different probability *p* and *q* (3-redundancy encoding/decoding)

* + 1. **Simulating Gilbert-Elliot Model with Hamming Encoding/decoding**

We then implement a Gilbert-Elliot model with Hamming encoding/decoding. In particular, we implement a model with (4,3) hamming encoding/decoding, and use this model with fixed emission probability and to evaluate the influence of different transition probability *p* and *q*. The other configuration is the same as the two experiments above.

Here is a plot that displays the distribution over 100 cases, as shown in Figure .8. And the Average BER for different probability *p* and *q* can be seen in Figure. 9. Again, we got following observations: (1) When the probability increases, BER gradually increases. This is because the channel is become easier to enter the Bad state, as probability increases. And the Bad state has a larger probability of transmission error, making the BER increase. (2) When the probability increases, BER gradually decreases. This may be because the channel is become easier to enter the Good state, as probability increases. And the Good state has a smaller probability of transmission error, making the BER decreases. (3) When the probability takes a smaller value and takes a larger value, we can get an optimal BER result since means that the channel is more likely to be stable in the Good state.

C:\Users\kingsley\Desktop\Stochastic Process\homework\results\hammdistri.emf

Figure. 8 The distribution of BER in different *p* and *q* (Hamming encoding/decoding (4,3))

C:\Users\kingsley\Desktop\Stochastic Process\homework\output\hamm.emf

Figure. 9 The Average BER for different *p* and *q* (Hamming encoding/decoding (4,3))

* + 1. **Comparation of Three Encoding/decoding Methods**

We also compared three encoding/decoding methods on the distributions of BER results under different *p* and *q*. The experimental results are shown in Figure. 10, which contains 5 cases of different value of *p*, that is, *p =* {0.1, 0.3, 0.5, 0.7, 0.9}. In Figure. 10, the red bar represents non-encoding/decoding, the green bar represents 3-redundancy encoding/decoding, and the blue bar represents Hamming encoding/decoding. Again, we use a fixed value of emission probability and , and observe the BER of different encoding/decoding methods under same *p* and *q*. Thus, we can make comparable results of these three methods.

From the Figure, we can see that the BER of the three methods are very close to each other, under a certain value of *p* and *q*, almost in all cases. In all three methods, when the probability increases, BER gradually decreases. When the probability increases, BER gradually increases. Another interesting observation is that, when the probability *p* is sufficiently low, Hamming encoding/decoding perform worse than the other two methods (the blue bar shows lower BER than the red bar and green bar when p = {0.1, 0.3, 0.5}). And when the probability *p* become sufficiently large, Hamming encoding/decoding perform better than the other two methods (the blue bar shows higher BER than the red bar and green bar when p = {0.7, 0.5}).

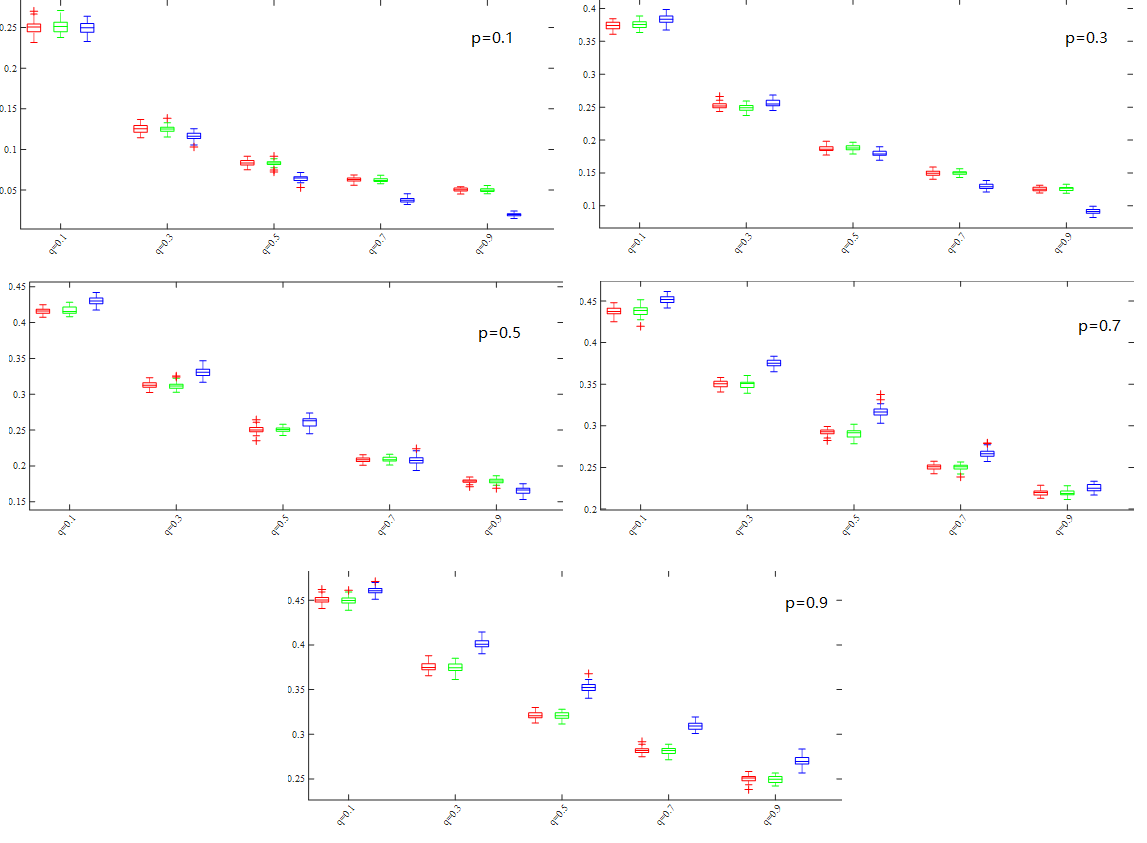
****

Figure. 10 Comparison of three methods

1. **Contributions**

Student *Cheng Wen* and Student *Xin Yi* discuss to overcome technical difficulties, and design our solution together. Student Cheng Wen implemented functions *HmmSeq.m*, *GEChannel.m*, and mainly wrote Section 2.1 and 3.1. Student Xin Yi implemented three different encoding/decoding methods (*Coder\_hamming.m, Coder\_null.m, Coder\_redun.m*), and mainly wrote Section 2.3 and 3.2. The other parts were written jointly by both students.

1. **References**

[1] E. O. Elliot, “Estimates of Error Rates for Codes on Burst-Noise Channels.” *Bell System Technical Journal*, 42: 5, pp 1977-1997, September 1963.

[2] Bilmes, Jeff A. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models." *International Computer Science Institute 4.510* (1998): 126.

[3] Welch, Lloyd R. Hidden Markov Models and the Baum-Welch Algorithm[J]. *IEEE information theory society newsletter*, 2003, 53(2):194-211.

[4] Rabiner L R. A tutorial on hidden Markov models and selected applications in speech recognition[J]. *Proceedings of the IEEE*, 1989, 77(2): 257-286.

[5] Rossi R J. Mathematical statistics: an introduction to likelihood-based inference[M]. John Wiley & Sons, 2018.

[6] Choudhury D R, Podder K. Design of Hamming Code Encoding and Decoding Circuit Using Transmission Gate Logic[J]. *International Research Journal of Engineering and Technology*, 2015, 2(7): 1165-1169.

[7] Bossert M. Channel coding for telecommunications[M]. John Wiley & Sons, Inc., 1999.