Week 1

1 Communication Theory

Communication theory is a field of information theory and mathematics that studies the technical process of information. Communication theory consists of:

- Analyzing real world communication systems using structured thinking or mathematics.
- Synthesizing real world communication systems from mathematics.

2 Signals

Signals are defined to be a function that conveys information about a phenomenon. They can also be said to be a medium which carry information relevant to a reciever.

Electromagnetic wave, voltage or current, digital signals carrying information are some examples of signals.

2.1 Are there any non-time based signals?

- Images: They are represented as a combination of monochrome images in three primary colors.

$$Image : \mathbb{R}^2 \to \{R, G, B\}$$

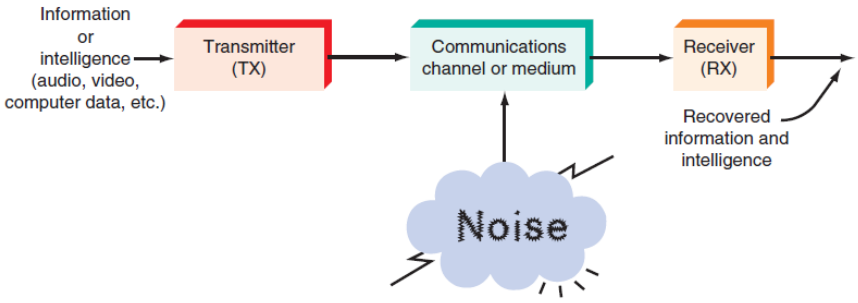
- Temperature at various points in a room.

$$Temperature : \mathbb{R}^3 \to K \quad (or\ Celsius)$$

- Videos: A video signal is a sequence of images.
- Text or a page in a book.

3 Communication

Communication is the process of exchanging information. Transmission, reception and reconstruction of the transmitted signal take place in this process.



Channel: Separates the source and reciever and is not directly under our control. This leads to some disturbances or imperfections in the channel, which is called noise.

3.1 Transmitter

The first step in sending a message is to convert it into electronic form suitable for transmission. The transmitter consists of various components designed to convert the electrical signal to a signal for sending it through the medium.

$$\boxed{\text{Source}} \xrightarrow{info.} \boxed{\text{Channel Coding/Amplification/Modulation/Antenna}} \rightarrow \boxed{\text{Channel}}$$

The transmitter is also called TX or TRX.

3.2 Reciever

$$\text{Noisy TX signal} \rightarrow \boxed{\text{Detect/estimate signal}} \xrightarrow{\text{Channel decoding}} \boxed{\text{Extract info.}} \rightarrow$$

A receiver is a collection of electronic components and circuits that accepts the transmitted message from the channel and converts it back to an understandable form. The dectection and estimation step is called 'demodulation'.

Channel coding and decoding is mainly used in digital communication systems. The reciever is called RX.

3.3 Features of a good communication system

- Low degradation in quality of original signal, called high fidelity.
- Quick reception of information, called high rate communication. (in digital comms., this happens via intelligent modulation, power control)
- Time delay between transmission and communication should be small, called latency. (eg: high latency required in video streams but not for downloads)

3.4 Modes of Communication

1. Point-to-point communication, in which the communication process takes place over a link between a single transmitter and a receiver. eg: Wired phone call.

2. Broadcasting, which involves the use of a single transmitter and multiple receivers. eg: Radio/TV broadcast, wi-fi, wireless cellular downlink

    - Multiple access channel, consists of multiple transmitters and a single reciever. eg: Cellular uplink.

4 Information

4.1 What is information?

Information is often thought of as the resolution of uncertainity. It can also mean the arrangement of particles.

Say that X is a random quantity representing the signal of interest (at some point in time or space).

$X \in \{0, 1\}$ and it can take value 0 with a probability of p and value 1 with probability $1 - p$. This statement is a general abstraction of many different real-world scenarios. X is called a random variable, while 0 and 1 together form the sample space.

How do we find the information-content (or "surprise", "average uncertainty") in X?

The information content(surprise) is inversely proportional to the probability that the event occurs. i.e.

$$Info.\ content\ in\ event\ \text{“}X = 0\text{”} \propto \frac{1}{P(X = 0)}$$

Our goal is to assign a function to the above 'information content'.

Suppose $X_1 \,\&\, X_2$ are independant random variables and x_1, x_2 are some values. Then

$$P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1) \times P(X_2 = x_2)$$

We expect the joint information content in $X_1 \,\&\, X_2$ to be the sum of the individual information contents.(as product is being converted to a sum, the log function is a good choice)

The information content in the event $X = x$ is $\log(\frac{1}{P(X=x)})$.

4.2 Entropy

The average (or expected) information content for a random variable X, with possible outcomes $x_1, \cdots, x_n$, which occur with probability $P(x_1), \cdots, P(x_n)$ is defined as:

$$H(X) := -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

$H(X)$ is called the entropy of X. Entropy in information theory is directly analogous to the entropy in statistical thermodynamics.

4.2.1 Lemma

Suppose $x_1 \in \mathcal{X}_1$ & $x_2 \in \mathcal{X}_2$ are **independant** random variables. Then,

$$H(x_1,\ x_2) = H(x_1) + H(x_2)$$

Week 2

1 Independent random variables

Definition: Two random variables X_1, X_2 are said to be independent if,

$$P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1)P(X_2 = x_2) \quad \forall\, x_1 \in \mathcal{X}_1 \,\&\, x_2 \in \mathcal{X}_2$$

$\boxed{P(X_1 = x_1, X_2 = x_2) \text{ means that } X_1 = x_1 \textbf{ AND } X_2 = x_2 \text{ occur.}}$

1.1 Claim

$$\sum_{x_2 \in \mathcal{X}_2} P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1)$$

Proof: Suppose A & B are disjoint/mutually exclusive. Then,

$$P(A \cap B) = 0 \quad \& \quad P(A \cup B) = P(A) + P(B)$$

Now the events $(X_1 = x_1) \cap (X_2 = x_2)$ are disjoint for different values of $x_2 \in \mathcal{X}_2$.(if $x_2 \neq x_2' \;\forall\; x_2 \in \mathcal{X}_2$)
Thus,

$$\begin{aligned}\sum_{x_2 \in \mathcal{X}_2} P(X_1 = x_1, X_2 = x_2) &= \sum_{x_2 \in \mathcal{X}_2} P\big((X_1 = x_1) \cap (X_2 = x_2)\big) \\ &= P\left(\bigcup_{x_2 \in \mathcal{X}_2} (X_1 = x_1) \cap (X_2 = x_2)\right)\end{aligned}$$

Now,

$$\begin{aligned}\bigcup_{x_2 \in \mathcal{X}_2} [(X_1 = x_1) \cap (X_2 = x_2)] &= (X_1 = x_1) \bigcap \left[\bigcup_{x_2 \in \mathcal{X}_2} (X_2 = x_2)\right] \\ &= (X_1 = x_1) \bigcap [x_2 \in \mathcal{X}_2]\end{aligned}$$

As $[x_2 \in \mathcal{X}_2]$ forms the entire sample space,

$$\begin{aligned}P\left(\bigcup_{x_2 \in \mathcal{X}_2} (X_1 = x_1) \cap (X_2 = x_2)\right) &= P\left((X_1 = x_1) \bigcap [x_2 \in \mathcal{X}_2]\right) \\ &= P\left((X_1 = x_1) \bigcap \Omega\right) \\ &= P(X_1 = x_1)\end{aligned}$$

2 Entropy

Suppose $X_1 \in \mathcal{X}_1$ & $X_2 \in \mathcal{X}_2$ are **independant** random variables. Then,

$$H(X_1,\ X_2) = H(X_1) + H(X_2)$$

Proof:

$$\sum_{\substack{x_1 \in \mathcal{X}_1 \\ x_2 \in \mathcal{X}_2}} P(X_1 = x_1, X_2 = x_2) \log \left(\frac{1}{P(X_1 = x_1, X_2 = x_2)} \right)$$

$$= \sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} P(X_1 = x_1, X_2 = x_2) \left[\log \frac{1}{P(X_1 = x_1)} + \log \frac{1}{P(X_2 = x_2)} \right]$$

$$= \sum_{x_1 \in \mathcal{X}_1} \log \frac{1}{P(X_1 = x_1)} \left(\sum_{x_2 \in \mathcal{X}_2} P(X_1 = x_1, X_2 = x_2) \right)$$

$$+ \sum_{x_2 \in \mathcal{X}_2} \log \frac{1}{P(X_2 = x_2)} \left(\sum_{x_1 \in \mathcal{X}_1} P(X_1 = x_1, X_2 = x_2) \right)$$

From the previous claim,

$$H(X_1, X_2) = \sum_{x_1 \in \mathcal{X}_1} P(X_1 = x_1) \log \frac{1}{P(X_1 = x_1)}$$

$$+ \sum_{x_2 \in \mathcal{X}_2} P(X_2 = x_2) \log \frac{1}{P(X_2 = x_2)}$$

$$= H(X_1) + H(X_2)$$

2.1 Conditional probability distribution

What if X_1 & X_2 are not independent? Then we would use conditional probability distribution. i.e.

$$P(X_2 = x_2 / X_1 = x_1) := \frac{P(X_2 = x_2, X_1 = x_1)}{P(X_1 = x_1)}, \quad P(X_1 = x_1) \neq 0$$

This definition for conditional probability satisfies the probability axioms and hence it is a valid probability measure.

2.2 Conditional entropy

Definition:

$$H(X_2/X_1) := \sum_{x_1 \in \mathcal{X}_1} P(X_1 = x_1) H(X_2/X_1 = x_1)$$

where,

$$H(X_2/X_1 = x_1) := \sum_{x_2 \in \mathcal{X}_2} P(X_2 = x_2/X_1 = x_1) \log \frac{1}{P(X_2 = x_2/X_1 = x_1)}$$

2.2.1 Support of a function

When $P(X = x) = 0$, it is not considered in calculating the entropy.

$$H(X) = - \sum_{\{x \in \mathcal{X}: P(X=x) \neq 0\}} P(X = x) log(P(X = x))$$

Suppose that f : X → R is a real-valued function whose domain is an arbitrary set X. The set-theoretic support of f, written supp(f), is the set of points in X where f is non-zero:

$$supp(X) = \{x \in X : f(x) \neq 0\}$$

Note: $P(X = x)$ can be denoted as $P_X(x)$ or $P(x)$.

2.3 Important Results

$$H(X) = - \sum_{x \in supp(P)} P(x) log(P(x))$$

$$H(X/Y) = \sum_{y \in supp(P_Y)} P_Y(y) H(X/Y = y)$$

$$H(X/Y = y) = - \sum_{x \in supp(P_{X/Y})} P_{X/Y}(x/y) log(P_{X/Y}(x/y))$$

2.4 Chain Rule

$$H(X, Y) = H(X) + H(Y/X) = H(Y) + H(X/Y)$$

Proof:

$$H(X) + H(Y/X) = - \sum_{x \in supp(P)} P(x) log(P(x)) + \sum_{x \in supp(P_X)} P(x) \left(\sum_{y \in supp(P_Y)} P(y/x) log \left(\frac{1}{P(y/x)} \right) \right)$$

$$= - \sum_{x \in supp(P_X)} \left(\sum_{y \in supp(P_Y)} P(x, y) \right) log(P(x)) + \sum_{x \in supp(P_X)} \left(\sum_{y \in supp(P_Y)} P(x, y) \right) log \left(\frac{1}{P(y/x)} \right)$$

$$= \sum_{x \in supp(P_X)} \sum_{y \in supp(P_Y)} P(x, y) \left(\log \frac{1}{P(x)P(y/x)} \right)$$

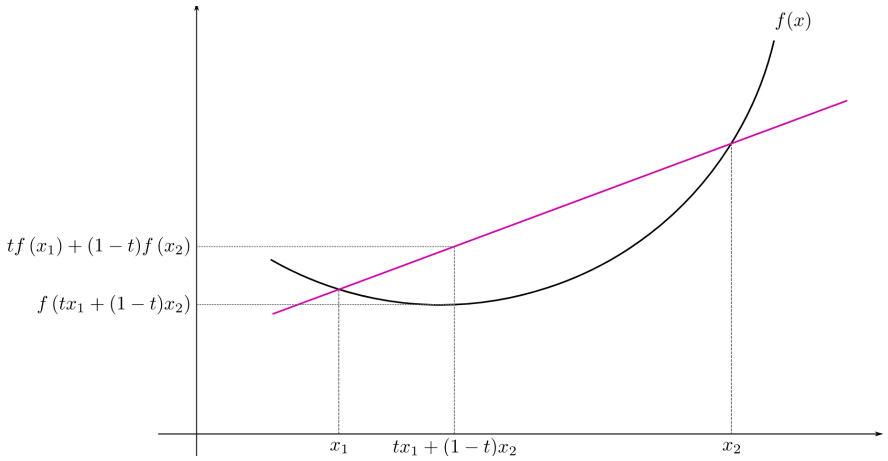
$$= - \sum_x \sum_y P(x, y) \log P(x, y) = H(x, y)$$

2.5 Upper and lower bound of entropy

$$0 \leq H(X) \leq |log\mathcal{X}|$$

Proof:
$H(X) = \sum_x P(x) log \frac{1}{P(x)} \rightarrow$ always positive as $P(x) > 0 \ \forall x \in supp(P_X)$.
$H(X) = 0$ when $|supp(P_X)| = 1$.

2.5.1 Jensen's inequality (Concave/convex functions)



$x_1 < x_3 < x_2$ such that $x_3 = tx_1 + (1-t)x_2$
If $f(x_3) \geq tf(x_1) + (1-t)f(x_2) \rightarrow$ Concave function.
If $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \rightarrow$ Convex function.
Hence, the above graph is a convex function.

We shall use the concavity property of the log function to obtain the proof for $H(X) \leq |log\mathcal{X}|$.

Proof:

$$H(X) = \sum_{x \in supp(P)} P(x) log \frac{1}{P(x)}$$

Let us assume λ_x is another notation for $P(x)$.

$$H(X) = \sum_{x \in supp(P)} \lambda_x log \frac{1}{P(x)}$$

This is a convex combination of $log \frac{1}{P(x)} : x \in supp(P_X)$ as $\sum_x P(x) = 1$.

$$H(X) \leq \log \left(\sum_{x \in supp(P_X)} \lambda_x \frac{1}{P(x)} \right) \leq \log |supp(P_x)|$$

$$\Rightarrow H(X) \leq \log |\mathcal{X}|$$

2.5.2 When is $H(X)$ is $log|\mathcal{X}|$?

When Jensen's inequality is satisfied with equality, i.e. if $f(x)$ is a straight line, but the $\log x$ curve is strictly concave.

Suppose $f(x)$ is strictly concave(or convex) & $\lambda_1, \lambda_2 \neq 0$ & $\lambda_1 + \lambda_2 = 1$.

If $f(\lambda_1 x_1 + \lambda_2 x_2) = \lambda_1 f(x_1) + \lambda_2 f(x_2)$, then $x_1 = x_2$.

More generally, if $\lambda_i \neq 0$, $\sum_{i=1}^{n} \lambda_i = 1$ and Jensen's inequality holds with equality, then $x_1 = x_2 \cdots x_n$.

Applying this to $H(X) \leq log|\mathcal{X}|$, from the previous proof,

$$H(X) = \sum_{x \in supp(P)} \lambda_x log \frac{1}{P(x)} \leq \log |supp(P_x)|$$

Suppose equality holds, then by the above claim we must have:

$$\frac{1}{P(X)} = const \quad \forall x \in supp(P_x)$$

$$\Rightarrow const.\ c = P(x) = \frac{1}{|supp(P_x)|}$$

If $|supp(P_x)| = |\mathcal{X}|$, $H(X) = \log |supp(P_x)|$. Then $P(X) = \frac{1}{|\mathcal{X}|}\ \forall x \in \mathcal{X}$.

We have just proved that $H(X) = \log_2 |\mathcal{X}|$, this can be true only when P_x is uniform. Thus:

Lemma: $H(X) = \log_2 |\mathcal{X}|$ iff P_x is uniform.

3 Relative Entropy/ Information Divergence/ Kullback-Leibler Divergence

Suppose there is a random variable X for which we have two probability distributions p_x & q_x.

Then the RE or ID or KL is defined as:

$$D(p_X||q_Y) := \sum_{x \in supp(p_x)} p(x) \log \frac{p(x)}{q(x)}$$

($D(p_X||q_Y)$ is a 'kind of' a distance measure between distributions p & q.)

3.1 Lower limit of relative entropy

$$D(p||q) \geq 0$$

Proof:

$$D(p||q) = -\sum_{x \in supp(p_x)} p(x) \log \frac{q(x)}{p(x)}$$

$$\geq -\log \left(\sum_{x \in supp(p_x)} p(x) \frac{q(x)}{p(x)} \right)$$

$$\geq -\log\left(\sum_{x\in supp(p_x)} q(x)\right)$$
$$\geq 0 \quad as \quad \sum q(x) \leq 1$$

3.2 When is RE equal to zero?

By applying Jensen's inequality for equality condition.

$$\frac{p(x)}{q(x)} = const.\ c \qquad \forall x \in supp(p_x)$$
$$\Rightarrow p(x) = cq(x) \quad \&$$
$$\sum_{x\in supp(p_x)} p(x) = \sum_{x\in supp(p_x)} cq(x) \forall x \in supp(p_x)$$

These togethermean that $c = 1$.

$$\Rightarrow p(x) = q(x) \forall x \in supp(p_x)$$
$$\Rightarrow D(p||q) \quad iff \quad p_x = q_x$$

3.3 Upper and lower limits of conditional entropy

$$H(X/Y) = \sum_{y\in supp(P_Y)} P_Y(y) \sum_{x\in supp(P_{X/Y})} P_{X/Y}(x/y) \log \frac{1}{P_{X/Y}(x/y)}$$

$$0 \leq H(X/Y) \leq H(X)$$

Proof:
$H(X/Y) \geq 0$ is true because $H(X/Y = y) \geq 0$ & $P(y) \geq 0$.

$$H(X) - H(X/Y) = \sum_{x\in supp(P_x)} P(x) \log \frac{1}{P(x)} - \sum_{x,y} P(x,y) \log \frac{1}{P(x/y)}$$
$$= \sum_{x,y} P(x,y) \log \frac{1}{P(x)} - \sum_{x,y} P(x,y) \log \frac{1}{P(x/y)}$$
$$= \sum_{x,y} P(x,y) \log \left(\frac{P(x/y)}{P(x)}\right) = \sum_{x,y} P(x,y) \log \left(\frac{P(x,y)}{P(x)P(y)}\right)$$

As both $P(x,y)$ & $P(x)P(y)$ are valid joint distribution of x, y. Hence,

$$H(X) - H(X/Y) = D(P(x,y)||P(x)P(y))$$
$$\geq 0 \qquad (as\ D(p||q) \geq 0)$$
$$\Rightarrow H(X/Y) \leq H(X)$$

Week 3

0.1 Joint Entropy in multiple RV's

$$H(X_1, \cdots, X_n) = \sum_{(x_1,\cdots,x_n)\in supp(P_{X_1,\cdots,X_n})} P(x_1, \cdots, x_n) \log_2 \frac{1}{P(x_1, \cdots, x_n)}$$

Where $P_{X,Y} := \mathcal{X} \times \mathcal{Y} \rightarrow$ Cartesian Product.
$\mathcal{X} \times \mathcal{Y} := \{(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$

0.2 Conditional Joint Entropy

$$H(X, Y/U, V) := \sum_{(u,v)\in supp(P_{U,V})} P_{U,V}(u, v) H(X, Y/U = u, V = v)$$

where,

$$H(X, Y/U = u, V = v) = \sum_{(x,y)\in supp(P_{X,Y/U=u,V=v})} P(x, y/u, v) \log_2 \frac{1}{P(x, y/u, v)}$$

This can be extended to any number of variables before and after the conditioning.

eg: $P(x_1, x_2, x_3/y_1, y_2) = P(x_1, x_2/x_3, y_1, y_2) + P(x_3/x_1, x_2, y_1, y_2)$.

1 Chain Rule for Joint Entropy

Lemma:

$$H(X_1, \cdots, X_n) = H(X_1) + H(X_2/X_1) + H(X_3/X_1, X_2) + \cdots + H(X_n/X_1, \cdots, X_n) = \sum_{i=1}^{n} H(X_i/X_{i-1}, \cdots, X_1) \quad (1)$$

Proof:

$$\begin{aligned} P(x_1, \cdots, x_n) &= P(x_1)P(x_2, \cdots, x_n/x_1) \\ &= P(x_1)P(x_2/x_1)P(x_3, \cdots, x_n/x_1, x_2) \\ &= P(x_1)P(x_2/x_1)P(x_3/x_1, x_2)P(x_4, \cdots, x_n/x_1, x_2, x_3) \\ &= P(x_1)P(x_2/x_1)P(x_3/x_1, x_2) \cdots P(x_n/x_1, \cdots, x_{n-1}) \\ &= \prod_{i=1}^{n} P(x_i/x_{i-1}, \cdots, x_1) \end{aligned}$$

We can substitute this value in the definition for joint entropy and expand, giving us the final result.

$$\begin{aligned} H(X_1, \cdots, X_n) &= -\sum_{x_1,\cdots,x_n} P(x_1, \cdots, x_n) \log \prod_{i=1}^{n} P(x_i/x_{i-1}, \cdots, x_1) \\ &= -\sum_{x_1,\cdots,x_n} \sum_{i=1}^{n} P(x_1, \cdots, x_n) \log P(x_i/x_{i-1}, \cdots, x_1) \\ &= -\sum_{i=1}^{n} \sum_{x_1,\cdots,x_n} P(x_1, \cdots, x_n) \log P(x_i/x_{i-1}, \cdots, x_1) \\ &= \sum_{i=1}^{n} H(X_i/X_{i-1}, \cdots, X_1) \end{aligned}$$

2 Mutual Information

We know that the average uncertainity in X is known as entropy of X, and it is given as $H(X)$. If an observer(RX) sees X, the uncertainity in X would become 0 i.e $H(X/X)$.

Hence the reduction in average uncertainity of X achieved by observing X is $H(X) - H(X/X) = H(X)$.

Suppose X, Y are related to each other and the observer knows it's joint probability distribution $P(x, y) = P(X = x, Y = y) \; \forall \; x, y$.

The "reduction in uncertainity of X after observing Y", "information gained about X after observing Y" or "mutual information between X & Y" is:

$$I(X;Y) := H(X) - H(X/Y)$$

By symmetry it follows that:

$$I(X;Y) = H(Y) - H(Y/X) = H(X) + H(Y) - H(X,Y)$$

1. $I(X;Y) \leq \min(H(X), H(Y))$

   Information gained depends on the quantized observation.

2. $I(X;Y) \geq 0$

Proof:

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X/Y) \\
&= -\sum_{x \in supp(P_X)} P(x)\log(P(x) - \left(-\sum_{x,y \in supp P_{X,Y}} P(x,y)\log(x/y) \right) \\
&= -\sum_{x,y} P(x,y)\log P(x) + \sum_{x,y} P(x,y)\log P(x/y) \\
&= \sum_{x,y} P(x,y)\log\frac{P(x/y)}{P(x)} \\
&= \sum_{x}\sum_{y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \\
&= D(P(x,y)||P(x)P(y))
\end{aligned}
$$

As we know that the relative entropy of 2 probability distributions is positive, the proof is complete.

3 Information Theory

- Efficient source representation.(using some random variable or sequence of R.V's)(data compression and source coding)
- High rate and high fidelity.(low probability of error)(channel coding)

3.1 Source Coding

Suppose we have $X \in \{a, b\}$, a binary source with probability distribution P_X. And an observer observes one instance of X, then wants to store(or communicate) it through a noise-free medium, which can carry or store only $\{0, 1\}$(bits).

But the reciever already knows P_X and it so happens that:

$$P_X(b) = 1 \quad P_X(a) = 0$$

Then RX doesn't need to read/recieve the encoded value to know X. It can simply declare the value of X to be b and it will be correct/error-free with probability 1.

We need a 0-length code as code is not required at all.

Suppose we allow a small probability of error $\Rightarrow P(\text{error}) \leq \epsilon, \quad \epsilon \in [0, 1)$.

Then we could have length = 0 & $P(\text{error}) \leq \epsilon$, if we let

$P_X(b) = 1 - \epsilon, P_X(a) = \epsilon$.

Week 4

1 Definitions

- Codeword: String resulting from some particular input/sequence.
- Code: Set of all codewords.

2 Efficient coding of information

We will be dealing with fixed length source code.

1. If we are willing to tolerate some small probability of error, we can conpress the source better.

   We can use a smaller length for representing the source by ignoring symbols which have a very low probability of occurence.

2. Club multiple random variables together.

   Suppose the source is transmitting data as $(X_1, X_2, X_3) = \{a, b\}^3$. Assume that X_1, X_2, X_3 are all independent random variables.

   $$P_{X_1,X_2,X_3}(x_1, x_2, x_3) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3) \quad \forall\, x_1, x_2, x_3 \in \{a, b\}$$

   We know the joint distribution from the individual distribution(called marginal distribution $P_{X_1}(x_1), P_{X_2}(x_2), P_{X_3}(x_3)$).

   $$P_{X_1,X_2,X_3}(x) = p^{n_a(x)}(1-p)^{n_b(x)}$$

   where, x is (x_1, x_2, x_3), $n_a(x)$ is the number of times 'a' occurs in x.

   If we have a 'compression scheme' for one variable from

   $$C_s : \{a, b\} \to \{0, 1\}$$

   We can get a scheme for $\{a, b\}^3$ using the above as:

   $$C'_s : (x_1, x_2, x_3) \to (C_s(x_1), C_s(x_2), C_s(x_3))$$

   $$\{a, b\}^3 \to \{0, 1\}^3$$

   Length of code: 3 bits.

   This code C'_s is as good as the original code C_s.

   In the case of encoding just one source symbol, our possible code lengths were either 0 or 1. Here, we have more choices 0, 1, 2 or 3 length binary strings (vectors or tuples) can be used.

Let,

$$C'_s : \{a, b\}^3 \to \{0, 1\}$$

Length is 1, while normalised length is $\frac{1}{3}$. This would be a good code if, (a, a, a) has a very high probability and the 7 other vectors have small probability.

3. Fixed length source sequences to fixed length codewords.

   Suppose we are allowed to combine multiple symbols and encode them together into some fixed length binary string, then this gives a more 'efficient' source code i.e. smaller normalised length.

   We shall impose some requirements:

   - We are encoding long source strings and can tolerate some small probability of error.
   - We have to use a fixed length source code. (Every n length source string is to be encoded into a l length binary vector/string/tuple. Fixed length meaning l doesn't change with the source string.)

   Assumption: n length random source vector is represented by $(x_1, x_2, \cdots, x_n)$, where x_i is the random variable representing the i^{th} output of the source $\in \{a, b\}$.

$$P_{X_i} = P_X \qquad (P_{X_i}(a) = P_X(a),\ P_{X_i}(b) = P_X(b))$$

   $X'_i s$ have same distribution and independent. In the language of communications, $X_i : i \in 1 \cdots n$ are said to be independent and identically distributed (IID).

   Question: Suppose n is very large, how many $a's$ & $b's$ do we expect to see in the random source sequence $(x_1, \cdots, x_n)$?

   Number of $a's$: np.

   Number of $b's$: $n(1-p)$.

   Number of sequences with such distribution of $a's$ and $b's$: ${}^nC_{np} = \binom{n}{np}$

   Now, for an efficient source code, we will encode only these $\binom{n}{np}$ sequences with unique codewords. And for all other sequences we use a single codeword. We assign each typical sequence a unique codeword of length L. And assign the same L length codeword to all atypical sequences, which is different from the other codewords used. Now,

$$\text{Typical } \to \binom{n}{np} \qquad \text{Atypical } \to 2^n - \binom{n}{np}$$

$$\Rightarrow L \text{ is atleast} = \qquad \log_2 \binom{n}{np} + 1$$

$$\log_2 \binom{n}{np} = \log_2 \left(\frac{n!}{(np)!(n-np)!} \right)$$

$$\begin{aligned}
(n! &\approx n(n-1)\cdots \approx n^n - \text{some poly. with deg } < n) \\
&= \log_2 n! - \log_2 np! - \log_2 (n-np)! \\
&\approx n \log_2 n - np \log_2 np - n(1-p) \log_2 (n(1-p)) - O(n) \\
&\approx n \log_2 n - np \log_2 np - np \log_2 n - n(1-p) log_2 n - n(1-p) log_2 (1-p) \\
&= n \left[p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \right] \\
&\approx nH(X), \text{ where X is the source random variable.}
\end{aligned}$$

Note: O(n) is **much** smaller than the other terms.

$\Rightarrow$ It is sufficient to have length of codewords $= nH(X) + 1$.

Length of codeword per source symbol $= H(X) + \frac{1}{n}$. i.e. We need $H(X)$ bits per source symbol (for a large n).

4. Fixed length source sequences to variable length codewords.

   Say,

   $$X \in \{A, B, C, D\} \qquad A \to 0,\ B \to 1,\ C \to 10,\ D \to 11$$

   In this case we are able to use compression, hence we shall demand zero probability of error here unlike the previous case.

   But we have a problem, consider source generates BA & C, the codeword for both of these is 01. This can be solved by using prefix-free code.

   Prefix-free code: A code is called prefix-free or P.F. code if no codeword in C is a prefix of another codeword in C. So the code defined above was not prefix-free code.

   Examples for P.F code:

   $$\{011, 10, 11, 00\} \tag{1}$$
   $$\{10, 11, 01, 00\} \tag{2}$$
   $$\{0, 10, 11, 110\} \tag{3}$$
   $$\{10, 110, 1110, 11110\} \tag{4}$$

   We know that the nodes of a binary tree without any children are called 'leaves'. Hence, the codewords of a prefix-free code all correspond to the leaves of the tree.

   Suppose $X \in \mathcal{X}, X \sim P_X$ & $|supp(P_X)| = S$ (X can take S possible values).

   Let $l_i : i = 1 \cdots n$, be the length of the binary codewords associated to the i^{th} symbol in supp(P_X).

   $$\text{Expected length of the code } \overline{L} = \sum_n p_i l_i$$

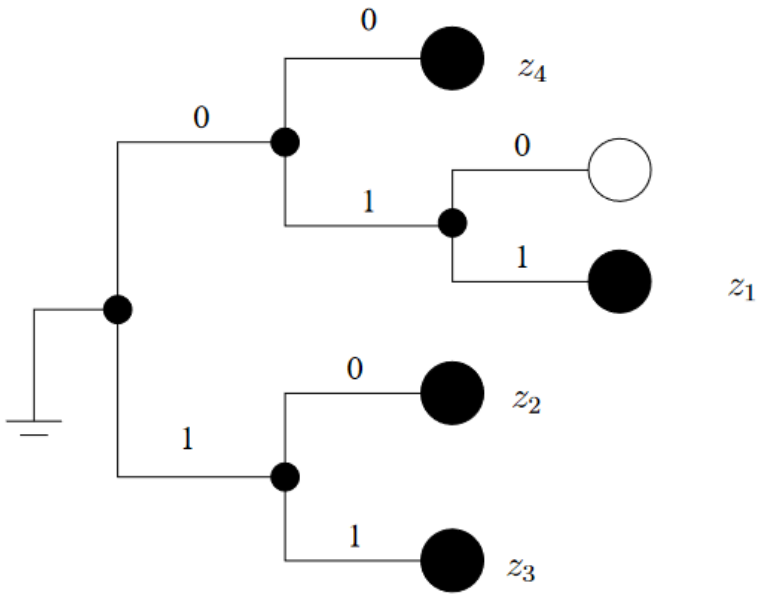


Figure 1: Binary tree of the prefix-free code of example (1)

Let $X \in \{A, B, C, D\}$ & $P_X(A) = \frac{1}{4}$, $P_X(B) = \frac{1}{4}$, $P_X(C) = \frac{3}{8}$, $P_X(D) = \frac{1}{8}$.

- Example (2):

$$\overline{C_2} = \sum_{i=1}^{4} p_i l_i = 2$$

- Example (4):

$$\overline{C_4} = \sum_{i=1}^{4} p_i l_i = \frac{27}{8} \approx 3.8$$

The goal of fixed-variable length source coding is to minimize $\overline{L}$.
The minimum possible $\overline{L}$ among all prefix-free code is denoted as $\overline{L}^*$.
We can show that,

$$H(X) \leq \overline{L}^* \leq H(X) + 1$$

Week 5

1 Some comments on engineering quantitites

1.1 Achievability

There exists some scheme using which we can show that the length of compression, rate or any other quantity of interest happens to be equal to be L or that value which is achievable.

1.2 Converse

No scheme exists which can improve upon some value. (can be upper or lower bound depending on situation)

Eg: Suppose in a running race, the fastest speed a human can run is say 10 m/s. Then

- 'Achievable': There exists some person who can run 10 m/s.
- 'Converse': There exists no person who can run more than 10 m/s.
- 'Matching Converse': No human can run at a speed $10 + \epsilon$, for any $\epsilon > 0$.

2 Channel Coding

Say we are inputting $x \in \mathcal{X}$, $(x_1, \cdots, x_n)$, in a channel and we are getting $y \in \mathcal{Y}$, $(y_1, \cdots, y_n)$, with $\mathcal{Y}$ as output alphabet.

If multiple x_i are mapping to a single y, it signifies a noisy channel as we would be unable to decode accurately.

To make this channel one-one (and therefore ensure correct decoding), we omit some sequences (n-length vectors in $\mathcal{X}^n$) from the set of all transmittable sequences.

This subset of transmittable sequences is called as the 'channel code' (or simply code). Denoted generally by $\mathscr{C}$. Note that, $\mathscr{C} \subseteq \mathcal{X}^n$.

Each vector in $\mathscr{C}$ is called a codeword. Number of bits required to represent $|\mathscr{C}|$ codewords

$$= \log_2 |\mathscr{C}| \text{ bits}$$

$$\text{Rate of the code } \mathscr{C} = \frac{\log_2 |\mathscr{C}|}{n} \text{ bits per channel use (bpcu or b/cu)}$$

Intuitively, higher the rate, more the chance of many-one kind of system and higher the chance of error.

2.1 Probabilistically noisy channel

Also called a random channel or random noise.

For $X = x \in \mathcal{X}$, there will be a probability distribution on the output random variable Y. The coditional distribution on Y given $X = x$,

$$P_{Y/X=x} = \{P(Y = y/X = x) : y \in \mathcal{Y}\}$$

These distributions $P_{Y/X}(y/x)$ $\forall x$ completely characterize or describe the random channel.

Now, we need to think about how to calculate $P(error)$. $X_i's$ are given as input to the random channel which are not independent. Then $(Y_1, \cdots, Y_n)$ is given out as output of the random channel, this is then sent into the decoder which then gives out $\hat{X}$, which is an estimate for X.

When $\hat{X} \neq X$, it is called an error event.

$$P(\text{Decoding error}) = P(\hat{X} \neq X)$$

Eg: $\mathcal{X} = \{0, 1\}$, $\mathscr{C} = \{000, 111\}$ instead of all 8 sequences. P(error) decreases but rate of code $= \frac{\log_2 |\mathscr{C}|}{n} = \frac{1}{3}$

Intuitively, it seems like if we want to decrease P(error) we have to increase n and we can expect the error to be close to 0 but this isn't the case.

For any small $\epsilon > 0$, there exists a code $\mathscr{C}$ with P(error) $\leq \epsilon$ & rate of the code $R(\mathscr{C}) = \max_{P_X}(I(X;Y)) - f(\epsilon)$

Note that:

1. I(X;Y) depends on $P_{Y/X=x}$ $\forall$ $x \in \mathcal{X}$.
2. I(X;Y) depends on distribution of P_X and P_Y.
3. X isn't a natural source upon which we have no control but it is the output of some encoding which encodes the 'raw source'.

   So $P_X(x)$ is generally assumed to be controllable in the mathematical framework of information theory.

The quantity $\max_{P_X}(I(X;Y))$ is called the 'Channel capacity', denoted by C.

3 Channel coding theorem

No matter what we do, we can't get a code with rate $> C$ (channel capacity) and expect a small probability of error.

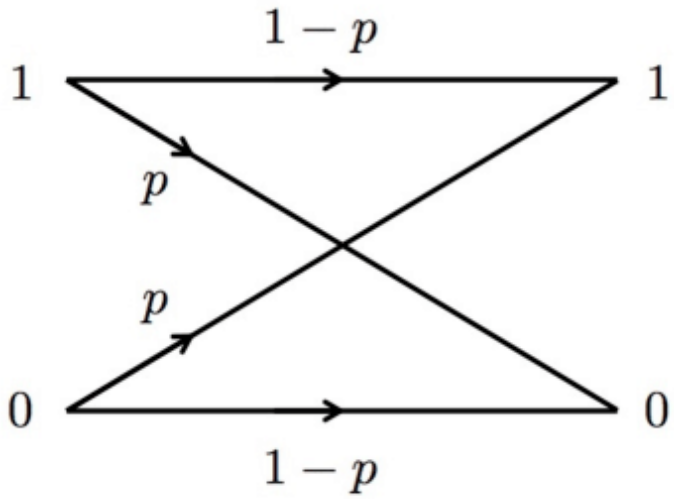
Note: To make the rate very close to C, we have to use a very high value of code length n.

3.1 Binary symmetric channel

$$\mathcal{X} = \{0, 1\} = \mathcal{Y}$$

A binary symmetric channel with crossover probability p, denoted by BSC(p), is a channel with binary input and binary output and probability of error p . That is, if X is the transmitted random variable and Y the received variable, then the channel is characterized by the conditional probabilities:

$$\begin{aligned} P(Y = 0/X = 0) &= 1 - p \\ P(Y = 0/X = 1) &= p \\ P(Y = 1/X = 0) &= p \\ P(Y = 1/X = 1) &= 1 - p \end{aligned}$$



We want to find channel capacity of this channel,

$$C = \max_{P_X}(I(X;Y))$$

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y/X) \\ &= H(Y) - \sum_{x \in \{0,1\}} P_X(x) H(Y/X = x) \\ &= H(Y) - \sum_{x \in \{0,1\}} P_X(x) H_2(p) \\ &= H(Y) - H_2(p) \end{aligned}$$

As we know $max(H(Y)) \leq Y$,

$$C_{BSC} = 1 - H_2(p)$$

where $H_2(p)$ is the binary entropy function defined by,

$$H_2(x) = x \log_2 \frac{1}{x} + (1-x) \log_2 \frac{1}{1-x}$$

Hence if P_x is a uniform distribution, the channel capacity can be 1.

We will now show the converse for the BSC(p) capacity. i.e. we will show that no matter what we do, we can't get a rate higher than $1 - H_2(p)$.

Let C be the code which has rate close to capcity.

$$R = \frac{\log_2 |C|}{n} \Rightarrow |C| = 2^{nR}$$

Suppose $\underline{c} \in C$ is transmitted, then

$$C = (c_1, \cdots, c_n) \rightarrow \text{BSC(p)} \rightarrow (y_1, \cdots, y_n)$$

There are $\sim np$ positions in $\underline{c}$ which are flipped to get $\mathcal{Y}$. (as the channel is independently acting on each bit)

We can expect any sequence in the set $S(\underline{c})$ as the output sequence.

$$S(\underline{c}) = \{\mathcal{Y} \in \{0,1\}^n : d_H(\underline{c}, \mathcal{Y}) = np\}$$

d_H is the Hamming distance between $\mathcal{Y}$ & $\underline{c}$. i.e number of positions in $\underline{c}$ which we have to flip.

Now around every codeword we draw this 'Hamming ball' of radius np.

We want $S(\underline{c}1)$ and $S(\underline{c}2)$ to be close to empty to prevent many-one mapping.

Because the code C has small probability of error, this means that the balls around the codewords in C are non-intersecting.

$$\Rightarrow |C| \leq \frac{2^n}{|S(C)|}$$

$|S(C)|$ is the number of vectors in any ball and the same for any $\underline{c} \in C = \binom{n}{np}$.

$$\Rightarrow |C| \leq \frac{2^n}{\binom{n}{np}}$$

$$\log_2 |C| \leq n - \log_2 \binom{n}{np} \leq n - nH_2(p)$$

(as we have previously seen)

$$\Rightarrow R = \frac{\log_2 |C|}{n} \leq 1 - H_2(p)$$

$$\Rightarrow R \leq C_{BSC}$$

We have studied the result of Shannon's noisy coding channel theorem for the particular case of BSC. But Shannon's achievability result wasn't constructive (i.e. doesn't identify a code which works but rather shows the existence of one such code).

Such a class of channels is calles discrete memoryless channels (without feedback).

It took around 50 years to come up with candidate constructions which have rate close to capacity and small probability of error. This process is called coding theory.

Week 6

1 Random variable source coding

Let $\underline{c}(x)$ be the codeword assigned to $x \in \mathcal{X}$.

$l(x)$ be the length of codeword assigned to x.

Here we are coding a single random variable and all codewords are binary strings. (Fixed-variable length source coding)

$$\mathscr{C} = \{\underline{c}(x) : x \in \mathcal{X}\}$$

$$L_{\mathscr{C}} = \sum_{x \in \mathcal{X}} p(x) l(x)$$

Our goal is to design a code which has minimum $L_{\mathscr{C}}$, L^*

$$L^* = \min_{\mathscr{C}} L_{\mathscr{C}}$$

1.1 Kraft inequality

Let $\mathscr{C}$ be any prefix-free (binary) code. Then,

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$$

Proof: We know that any prefix-free code can be represented using a binary tree which has 'leaves' as the codeword. In the binary tree corresponding to the code, the depth of the tree would be the length of the largest codeword in the code. ($l_{\max} = max_{x \in \mathcal{X}} l(x)$)

Suppose there is a codeword $\underline{c}$ of length l represented by a node at depth l($l \leq l_{\max}$).

Then $\underline{c}$ has $2^{l_{\max} - l}$ successors at level $l_{\max}$. Also, none of these are codewords as the code is prefix-free. Now,

$$\sum_{\underline{X} \in \mathcal{X}} 2^{l_{\max} - l(\underline{X})} \leq 2^{l_{\max}} \tag{1}$$

This is true if distinct codewords $\underline{c1}$, $\underline{c2}$ don't have common successors at $l_{\max}$ level. Any successor of $\underline{c1}$ at $l_{\max}$ has $\underline{c1}$ as a prefix. Similarly in the other case as well.

Suppose $l(\underline{c1}) \leq l(\underline{c2})$, so there can be a common successor $\underline{v}$ at $l_{\max}$.

Then, $\underline{v}$ has first $l(\underline{c1})$ places as $\underline{c1}$ and first $l(\underline{c2})$ places as $\underline{c2}$.

$\Rightarrow$ c1 should be a prefix of c2 which isn't true as $\mathscr{C}$ is a prefix-free code. Hence, no pair of codewords in $\mathscr{C}$ have any common successors.

Hence (1) is true. And dividing both sides by $2^{l_{\max}}$ gives us the Kraft inequality.

1.2 Lemma

$$L^* \geq H(X)$$

Any prefix-free code for X has average length of at least $H(X)$.

Proof:

$$L - H(X) = \sum_{x \in \mathcal{X}} p(x) l(x) - \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}$$

We know,

$$D(p||q) = \sum_{x \in supp(P_X)} p(x) \log \frac{p(x)}{q(x)}$$

Let

$$q(x) := \frac{2^{-l(x)}}{\sum_{x \in \mathcal{X}} 2^{-l(x)}}$$

$$\begin{aligned}
D(p_X||q_X) &= \sum_{x \in supp(P_X)} p(x) \log \frac{p(x)}{\frac{2^{-l(x)}}{\sum_{x' \in \mathcal{X}} 2^{-l(x')}}} \\
&= -\sum p(x) \log \frac{1}{p(x)} + \sum p(x) \log \frac{1}{\frac{2^{-l(x)}}{\sum_{x' \in \mathcal{X}} 2^{-l(x')}}} \\
&= -H(X) + \sum_{x \in supp(p_x)} p(x) \log 2^{l(x)} + \sum_{x \in supp(p_x)} p(x) \log \sum_{x'} 2^{-l(x')} \\
&= -H(X) + L - \epsilon \qquad \text{(from Kraft's)} \\
&\leq -H(X) + L
\end{aligned}$$

$$\Rightarrow L_{\mathscr{C}} - H(X) \geq 0$$

$$L^* \geq H(X)$$

Note: Equality happens only iff $p_x = q_x$ and ϵ is zero.

1.3 Lemma

Suppose we have a random variable $X \in \{x_1, x_2, \cdots, x_k\}$ and positive integers such that $\sum_{i=1}^{k} 2^{-l_i} \leq 1$.

Then there exists a prefix free code for X with codeword lengths $l_1, l_2, \cdots, l_k$.

Proof: We can construct a binary tree with leaves at depths $l_1, l_2, cdots, l_k$ such that none of these nodes are successors of each other, i.e. they are leaves of some binary tree (valid p-f code).

Assume that $l_1 \leq l_2 \leq \cdots \leq l_k$, without loss in generality , for any $i \leq k$,

$$\sum_{1}^{i-1} 2^{-l_i} < 1 \tag{2}$$

Imagine we take the full binary tree upto level l_k. At each step of the algorithm we intend to pick one available (undeleted) node from the above tree at level l_i and delete all it's successors from the tree. Repeat this process for $i = 1, \cdots, k$, we will then have a prefix-free tree.

We have to show that at each step $i = 1, \cdots, k$, there is atleast one node left undeleted at depth i. We shall use observation (2).

Clearly at step 1, there is a node at l_1, ($l_1 \geq 1$).

After $i-1$ steps, assume that we have picked nodes at level $l_1, \cdots, l_{i-1}$ and appropriately deleted. We want to show that there is a node at level i.

Total nodes at level l_k which aren't in the tree after $(i-1)^{th}$ step

$$= \sum_{j=1}^{i-1} 2^{l_k - l_j}$$

Hence, number of nodes remaining at level l_k

$$= 2^{l_k}(1 - \sum_{j=1}^{i-1} 2^{-l_j})$$

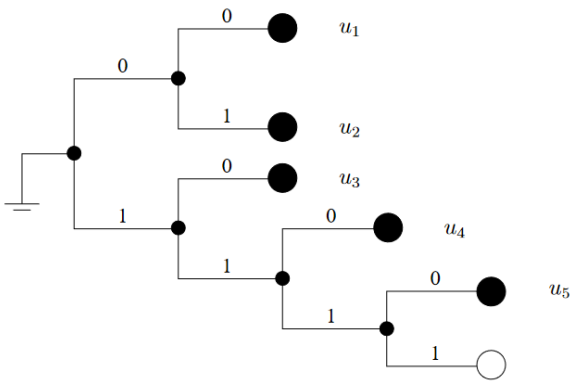
$\Rightarrow$ Number of nodes remaining at l_k in tree at after $(i-1)^{th}$ step > 1.

$\Rightarrow$ At least one survivor node should be present at level l_i also. So we can pick a node for the i^{th} step from level l_i also. This completes the proof.

Remark: We construct the tree from smallest length to largest length codewords. Contrast this with optimal source coding we shall see Hoffman codes later.

1.4 Example

Example 2.3: Construct a binary prefix-free code with lengths $w_1 = 2, w_2 = 2, w_3 = 2, w_4 = 3, w_5 = 4$.
Since $\sum_{i=1}^{5} 2^{-w_i} = 1/4 + 1/4 + 1/4 + 1/8 + 1/16 = 15/16 < 1$, we know such a prefix-free code exists.
We "grow" such a code to obtain



U	Z
u_1	00
u_2	01
u_3	10
u_4	110
u_5	1110

1.5

Now, suppose that the source random variable $X \sim P_X$, we want to obtain a collection of integers $l_1, \cdots, l_k$ such that Kraft inequality is satisfied, then we know how to get the code for X.

$$\sum_{i=1}^{k} 2^{-l} \leq 1$$

- Suppose all codewords are of the same length,

  $$k2^{-l} \leq 1 \Rightarrow l \geq \log k$$

  Hence we can pick $l = \lceil \log k \rceil$ (ceil function). But there is no guarantee that this code is 'good', it may not have small average length.

- We know average length $= \sum p_i l_i$.

  Then we will choose small l_i for larger p_i. (while taking care that it satisfies Kraft inequality)

2 Shannon-Fano code

We fix,

$$l_i = \lceil \log_2 \frac{1}{p_i} \rceil \tag{3}$$

where p_i is the probability of X taking the i^{th} value in $\mathcal{X}$.
Clearly $l_i \geq 1$.

$$\sum_{i=1}^{k} 2^{-l_i} = \sum_{i=1}^{k} 2^{-\lceil \log_2 \frac{1}{p_i} \rceil} =\leq \sum_{i=1}^{k} 2^{-\log_2 \frac{1}{p_i}}$$
$$= \sum_{i=1}^{k} p_i = 1$$

The lengths given by (3) satisfy Kraft inequality. We can use the tree-pruning algorithm (see sec 1.3) to get a prefix-free code for X.

This code obtained is called as the Shannon-Fano code.

Now,

$$\begin{aligned} L_{\text{Shannon-Fano}} &= \sum_{i=1}^{k} p_i \lceil \log_2 \frac{1}{p_i} \rceil \\ &< \sum_{i=1}^{k} p_i \left(\log_2 \frac{1}{p_i} + 1 \right) \\ &< H(X) + 1 \end{aligned}$$

But S-F code is not always an optimal length prefix-free code.

2.1 Example for Shannon-Fano code

$$X \in \mathcal{X} = \{x_1, x_2, x_3, x_4\}$$

$$P_X(x_i) = \begin{cases} 1/4, & \text{if } i = 1 \\ 1/2, & \text{if } i = 2 \\ 1/9, & \text{if } i = 3 \\ 5/36, & \text{if } i = 4 \end{cases}$$

Lengths satisying the Kraft's inequality for the Shannon-Fano code

$$l_i = \lceil \log_2 \frac{1}{P_X(x_i)} \rceil = \begin{cases} 2, & \text{if } i = 1 \\ 1, & \text{if } i = 2 \\ 4, & \text{if } i = 3 \\ 3, & \text{if } i = 4 \end{cases}$$

$$\bar{L} = \sum_{i=1}^{4} p_i l_i = \frac{57}{36}$$

Obtaining the Shannon-Fano code:

- Arrange lengths in ascending order: $l_2 \leq l_1 \leq l_4 \leq l_3$. Hence we pick off a node at depth 2, then 1, 4 and at 3 and delete all the successors.

- Now we can choose the codewords from the binary tree, say

$$x_1 \to 01 \qquad x_2 \to 1 \qquad x_3 \to 0010 \qquad x_4 \to 000$$

Hence, the prefix-free Shannon-Fano code is $\{01, 1, 0010, 000\}$.

3 Huffman coding

We will show an optimal code construction that has smaller average length compared to the Shannon-Fano code.

3.1 Lemma's

Let us see some intuitive lemma's about an optimal code for a random variable X:

Assume that $X \in \mathcal{X} = \{x_1, \cdots, x_k\}$.
$P_X(x_i) = p_i, i = 1, \cdots, k$, without loss in generality, let $p_1 \geq \cdots \geq p_k$.

1. Consider that $l_1, \cdots, l_k$ are the lengths of the codewords associated to the messages $x_i, i = 1, \cdots, k$ respectively in any optimal code for X.

   Then,

$$l_1 \leq \cdots \leq l_k \tag{4}$$

   Proof: Suppose $\mathscr{C}$ is an optimal code in which $\exists$ some distinct $i, j \in \{1, \cdots, k\}$ such that $p_i > p_j$ but $l_i > l_j$. (Assumption of the contrary)

   We shall show that there is another code $\mathscr{C}'$ which has smaller average length $L_{\mathscr{C}'}$ than $L_{\mathscr{C}}$.

   Cosider the code $\mathscr{C}'$ in which the codewords for x_i & x_j are swapped between those in $\mathscr{C}$.

$$\underline{c}_{\mathscr{C}'}(x_j) = \underline{c}_{\mathscr{C}}(x_i) \qquad \underline{c}_{\mathscr{C}'}(x_i) = \underline{c}_{\mathscr{C}}(x_j)$$

   Now in $\mathscr{C}$,

$$l_{i,\mathscr{C}} := \text{length of codeword of } x_i \text{ in } \mathscr{C} = l_j$$
$$l_{i,\mathscr{C}'} = l_i$$

   So, $l_{i,\mathscr{C}'}$ is having the same codewords as $\mathscr{C}$ and hence is also prefix-free.

$$\bar{L}_{\mathscr{C}'} = \sum_{k=\{1,\cdots,k\}\setminus\{i,j\}} p_k l_k + p_i l_i + p_j l_j$$

$$\bar{L}_{\mathscr{C}} = \sum_{k=\{1,\cdots,k\}} p_k l_k$$

$$\bar{L}_{\mathscr{C}'} - \bar{L}_{\mathscr{C}} = p_i(l_j - l_i) + p_j(l_i - l_j) = (p_i - p_j)(l_j - l_i)$$

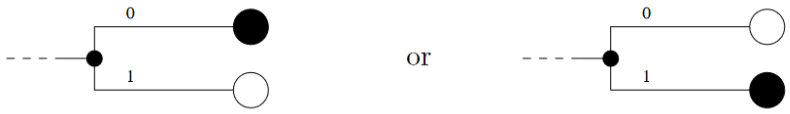
We have $p_i > p_j$, but $l_i > l_j$,

$$\Rightarrow \bar{L}_{\mathscr{C}'} - \bar{L}_{\mathscr{C}} < 0 \Rightarrow \bar{L}_{\mathscr{C}'} < \bar{L}_{\mathscr{C}}$$

$\Rightarrow \mathscr{C}$ is not optimal, thus this a contradiction. (4) has to happen for an optimal code.

2. Consider the tree representation of a code which is optimal code. In such a tree, it should be true that either each node is a codeword or it has at least 2 successors which are codewords. i.e. There are no unused leaves in the tree of an optimal code.

   Proof: Let $\mathscr{C}$ be the optimal code. Let us consider the tree corresponding to this, suppose it has an unused leaf. Because the code is optimal, these unused leaves must be at maximum depth in the tree. Then, for at least one value x_i of $\mathscr{C}$, we have the situation



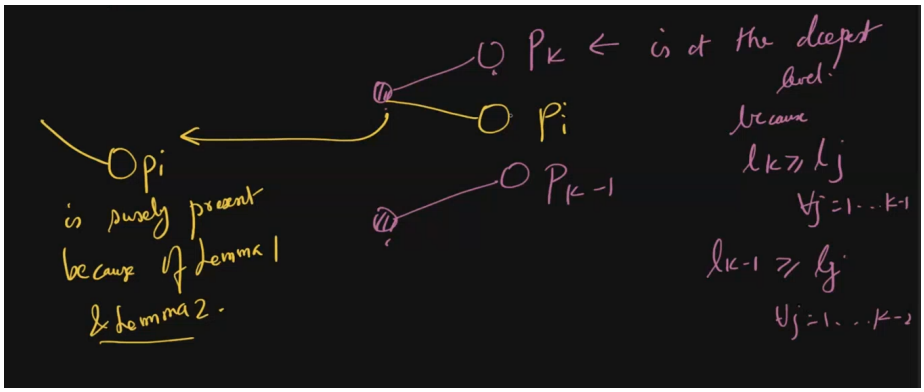
   In either case, we can delete the last digit of the codeword for x_i (without changing the other codewords) and still have a prefix-free code. But the new code has smaller $\bar{L}$ and thus the original code could not have been optimal.

3. There is an optimal prefix-free code for random variable X such that the codewords associated to the two smallest probability symbols are siblings (share the same parent). i.e. two least likely codewords differ only in their last digit.

   Proof: Let $\mathscr{C}$ be some optimal code for random variable. Suppose $\mathscr{C}$ already satifies the property specified above, then we are done.

   Suppose $\mathscr{C}$ doesn't satify the property, we know:

$$l_{k-1} \leq l_k \qquad as \qquad p_{k-1} \geq p_k$$



Note that $l_i = l_k$ and $i \neq k-1$.

$$\Rightarrow l_i \leq l_{k-1} \leq l_k = l_i \qquad \text{from Lemma-1}$$

$$\Rightarrow l_i = l_{k-1} = l_k$$

Interchange the codewords for the i^{th} and $(k-1)^{th}$ symbols and get a new code $\mathscr{C}'$.

$$\Rightarrow \bar{L}_{\mathscr{C}} = \bar{L}_{\mathscr{C}'}$$

Now $\mathscr{C}'$ satisfies the property in Lemma 3.

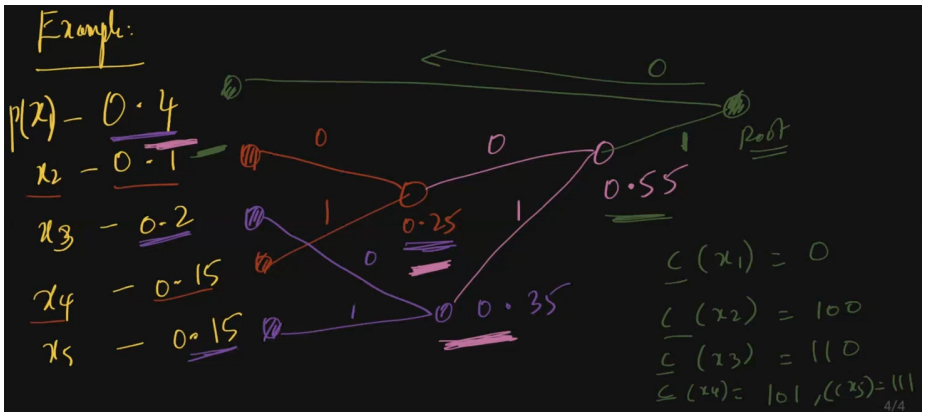
3.2 Huffman coding

Say we have p_X, assume that $p_1 \geq \cdots \geq p_k$. Unlike Shannon-Fano code, here we are building the tree in reverse. We pick p_{k-1} and p_k as they have the least 2 probabilities.

These are assigned to siblings in the tree and now the parent of them has probability $p_{k-1} + p_k$.

Now we can form a new distribution using these $k-1$ probabilities (symbols) i.e. $p_1, \cdots, p_{k-2}, p_{k-1} + p_k$. Now, if $\{p_{k-2}, p_{k-1} + p_k\}$ are the least 2 probabilities, a new node is assigned to p_{k-2}. We repeat the process until all codewords are assigned.

3.3 Example


Example:
p(x) - 0.4
x2 - 0.1
x3 - 0.2
x4 - 0.15
x5 - 0.15
0
0
1
0
1
0
1
0.25
0.35
0.55
Root
C(x1) = 0
C(x2) = 100
C(x3) = 110
C(x4) = 101, C(x5) = 111
4/4

Week 8

1 Achievability in channel coding theorem for BSC

For any $\epsilon > 0$, there exists a sequence of codes $\mathscr{C}$(one for each x) with rate $R : 1 - H_2(p) - \epsilon$ and P(error)$\to 0$ as $x \to \infty$.

Proof: We will use a 'random' code. Entire argument is for a fixed value of n. We will assume n is large (so that number of flips is np).

We will pick a code $\mathscr{C}_n$ of rate $R = 1 - H_2(p) - \epsilon$. So we want $|\mathscr{C}_n| = 2^{nR}$. (such that nR is an integer)

1.1 Random code generation

- Pick each codeword in $\mathscr{C}_n$ from $\{0,1\}^n$ uniformly at random. Then P(codeword is specific n-length sequence)$= \frac{1}{2}^n$
- Repeat this process 2^{nR} times.

We get,

$$|\mathscr{C}_n| = 2^{nR}$$

Now we want to prove,

$$P(\hat{\underline{c}} \neq \underline{c}) \leq 2^{-n\delta} \qquad \forall \qquad \underline{c} \in \mathscr{C}_n, \delta > 0$$

For getting a handle on the probability of error, we have to first define the decoding function (i.e. how is estimate $\hat{\underline{c}}$ calculated from a particular recieved vector).

Let the decoding function be denoted by,

$$D : \{0,1\}^n \to \mathscr{C}_n$$

For any $y \in \{0,1\}^n$,

$$D(y) := \text{argmax}_{\underline{c}' \in \mathscr{C}_n} P(y|\underline{c}')$$

D(y) is the estimate $\hat{\underline{c}}$ of the code when recieved vector is y. This is called the "Maximum likelihood decoding rule".

To show that $P(\hat{\underline{c}} \neq \underline{c})$ is small, we will show that $P(D(y) \neq \underline{c})$ is small where y is the random vector when $\underline{c}$ is transmitted.

For the above decoder, we want to show $P(\hat{\underline{c}} \neq \underline{c}) \leq 2^{-n\delta}$, $\delta > 0$.

For any specific $\underline{c}$, we want to find an upper limit for $P(\hat{\underline{c}} \neq \underline{c})$. This occurs when $\underline{c}$ is not the closest codeword to y.

By law of large numbers,

$$d_H(y, \underline{c}) \approx np$$

$$B(y, np) = \{\underline{x} \in \{0,1\}^n : d_H(\underline{x}, \underline{y}) \leq np\}$$

If there are other codewords within this ball B(y,np), the decoder can make an error. Else it will not make an error.

$$\begin{aligned} P(\hat{\underline{c}} \neq \underline{c}) &\leq P(\underline{c}' \in B(y, np) : \hat{\underline{c}} \neq \underline{c}) \\ &\leq \frac{|B(y, np)|}{2^n} \\ &\leq \frac{\sum_{i=0}^{np} \binom{n}{i}}{2^n} = \frac{\binom{n}{np} + \sum_{i=0}^{np-1} \binom{n}{i}}{2^n} \\ &\approx \frac{2^{nH_2(p)}}{2^n} = 2^{-n(1-H_2(p))} \end{aligned}$$

As n grows large, this value will be dominated by the first term of value $\binom{n}{np}$.

$$\Rightarrow P(\hat{\underline{c}} \neq \underline{c}) \leq 2^{-n(1-H_2(p))} \tag{1}$$

(we assume p < 0.5, otherwise we can change the channel to BSC(p') where p'= 1-p)

We would like to show this result (1) for all codewords rather than specific codewords. We want,

$$P\left(\bigcup_{\underline{c} \in \mathscr{C}} (\hat{\underline{c}} \neq \underline{c})\right) \leq 2^{-n\delta} \qquad \delta > 0$$

This is known as the union bound.

1.2 Union bound

We know,

$$\begin{aligned} P\left(\bigcup_{\underline{c} \in \mathscr{C}_n} (\hat{\underline{c}} \neq \underline{c})\right) &\leq \sum_{\underline{c} \in \mathscr{C}_n} P(\hat{\underline{c}} \neq \underline{c}) \\ &\leq P\left(\bigcup_{\underline{c} \in \mathscr{C}} (\hat{\underline{c}} \neq \underline{c})\right) \leq \sum_{\underline{c} \in \mathscr{C}_n} 2^{-n(1-H_2(p))} \\ &\leq 2^{nR} 2^{-n(1-H_2(p))} \\ &\leq 2^{-n(1-H_2(p)-R)} \\ &\leq 2^{-n\varepsilon} \end{aligned}$$

$$\Rightarrow P(\hat{\underline{c}} \neq \underline{c}) \leq 2^{-n\epsilon} \quad \forall \underline{c} \in \mathscr{C}_n$$

Hence proved.

In practice using random codes and minimum distance/likelihood decoder (MDD/MLD) for BSC is very complex (complexity of decoder/encoder is very high). Hence, we use structured codes which have low encoding/decoding performance, mainly linear codes.

2 Linear codes

The random code construction is not really useful for implementation as:

1. we could end up with a bad code due to the random construction.
2. Encoding and decoding complexity is very large.

So we want codes which are good in rate and P(error) and also have reasonable encoding/decoding complexity. An important class of codes having above properties are linear codes.

We will look at some simple examples of liner codes for binary channel with worst-case/bounded error model. Construction of codes which are useful for implementation is dealt with in coding theory.

2.1 Worst case/bounded error model for binary channel

Let t, n be some integers such that t $\leq$ n.

We input some $x \in \{0,1\}^n$ to a binary channel and we get $y \in \{0,1\}^n$. $d_H(y,x) \leq t \Rightarrow$ There are atmost t positions where recieved vector y is different from transmitted vector x.

For this channel, we want to design a code $\mathscr{C} \subseteq \{0,1\}^n$ (such that all upto t errors are corrected).

2.1.1 Example

Now, suppose $= \{0,1\}^n$, t=1.

Let us construct a situation in which the decoder will surely make an error in decoding.

Suppose $\underline{c} = (1, \cdots, 1) \in \mathscr{C}$ was transmitted.

Suppose $y = (0, 1, \cdots, 1)$, $y \in$. (we will assume min Hamming distance decoder $\hat{\underline{c}} = argmin_{\mathscr{C}} d_H(y, \underline{c})$)

Hence iff $\underline{c} = y$,

$$min_{\underline{c} \in \mathscr{C}} d_H(y, \underline{c}) = 0$$

$\Rightarrow$ Decoding error has happened as $\hat{\underline{c}} \neq \underline{c}$ (estimate and transmitted are not same).

So correcting any $t \geq 1$ error requires us to pick proper subsets of $\{0,1\}^n$.

But we also want to pick large subsets of $\{0,1\}^n$ as the code because we want to maximize $R = \frac{\log |\mathscr{C}|}{n}$ bits/channel use.

But picking large $\mathscr{C}$, codewords are closer in Hamming distance, which means that it is more likely to create decoding errors.

2.2 Lemma

Let $\mathscr{C} \subseteq \{0,1\}^n$ be chosen. Define,

$$d_{min}(\mathscr{C}) = min_{c,c' \in \mathscr{C} \& c' \neq c} d_H(c, c')$$

$\mathscr{C}$ can be correct upto t errors if and only if $d_{min}(\mathscr{C}) \geq 2t + 1$.
Proof:
If part: Given: $\mathscr{C}$ can correct any t errors.
To prove: $d_{min}(\mathscr{C}) \geq 2t + 1$.
Given statement implies any for any c, c' $\in \mathscr{C}$.

$$B_t(c) = \text{Hamming ball of radius t} := \{x \in \{0,1\}^n : d_H(x, c) \leq t\}$$

Then,

$$B_t(c) \bigcap B_t(c') = \phi$$

$$\Rightarrow d_H(c, c') > 2t \qquad \forall \qquad c, c' \in \mathscr{C} \quad c \neq c'$$

This can be proved by contradiction.

2.3 Terminology

- Size of code = $|\mathscr{C}|$.
- Length of code (Block length) = n.

Lemma above relates the error correcting capability of the code with the minimum distance, minimum disctance calculation has nothing to do with the channel.

Suppose code has minimum distance of d, then it can be used on a channel for correcting upto $\lfloor \frac{d-1}{2} \rfloor$.

This says that code design can be theoretically done independent done of the channel and it's performance can be tested based on it's minimum distance.

2.4 Hamming Bound

Hamming bound is the upper bound on the size of code based on a given minimum distance.

Lemma: Let $\mathscr{C}$ be any code with $d_{min}(\mathscr{C}) = d$.
Then,

$$|\mathscr{C}| \leq \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i}} \qquad t = \lfloor \frac{d-1}{2} \rfloor$$

Proof follows as we can pick atmost one codeword per ball.

2.5 Linear codes (over $\mathbb{F}_2$)

$$\mathbb{F}_2 \to (\{0,1\}, +, \cdot)$$

$$+ \to XOR$$

Definition: A linear code over $\mathbb{F}_2$ of length n is subset $\mathscr{C} \subseteq {\mathbb{F}_2}^n$ and also a subspace of the vector space ${\mathbb{F}_2}^n$.

$$\Rightarrow \forall a, b \in \mathbb{F}_2 \quad c_1, c_2 \in \mathscr{C}$$

$$ac_1 + bc_2 \in \mathscr{C}$$

Since only non-trivial values of a, b above are a=1 and b=1.

$\Leftrightarrow \mathscr{C}$ is a subspace of ${\mathbb{F}_2}^n$ iff $\forall c_1, c_2 \in \mathscr{C}$, we have $c_1 + c_2 \in \mathscr{C}$.

Week 9

1 Lemma

If $\mathscr{C}$ is a linear code, then

$$d_{min}(\mathscr{C}) = \min_{\underline{c} \neq 0} w_H(\underline{c})$$

where, Hamming weight, $w_H(\underline{c})$ = number of non zero positions in $\underline{c}$.

Proof:
By definition,

$$\begin{aligned} d_{min}(\mathscr{C}) &= \min(d_H(\underline{c}_1, \underline{c}_2)) && \underline{c}_1, \underline{c}_2 \in \mathscr{C}; \underline{c}_1 \neq \underline{c}_2 \\ \min(d_H(\underline{c}_1, \underline{c}_2)) &= \min(w_H(\underline{c}_1 - \underline{c}_2) && \underline{c}_1, \underline{c}_2 \in \mathscr{C}; \underline{c}_1 \neq \underline{c}_2 \\ &= \min(w_H(\underline{c})) && \underline{c} \neq 0; \underline{c} \in \mathscr{C} \end{aligned}$$

Hence proved.

2 Examples

We know that every subspace of a vector space has a basis, i.e a set of linearly independent vectors from the subspace which span the subspace.

1. Suppose $\mathscr{C} = \mathbb{F}_2^n$,
   - Then any set of n linearly independent vectors from $\mathbb{F}_2^n$ will be a basis of $\mathscr{C}$.
   - In particular we can choose the standard basis, $\underline{c}_1 = (1, 0, \cdots, 0), \underline{c}_2 = (0, 1, 0, \cdots, 0), \cdots, \underline{c}_n = (0, \cdots, 0, 1)$
2. Suppose $\mathscr{C} = \{(0, \cdots, 0), (1, \cdots, 1)\}$
   - As this code is closed under addition, this is a valid linear code.
   - The basis for $\mathscr{C}$ will be $\{(1, \cdots, 1)\}$.
   - This code encodes 1 bit.
3. Suppose $B = \{g_1, \cdots, g_k\}$, $k < n$ are a set of linearly independent vectors in $\mathbb{F}_2^n$. What is linear code $\mathscr{C}$ for which B is a basis?
   - Set of all linear combinations of vectors in B, i.e.

$$\mathscr{C} = \text{span}(B) = \left\{ \sum_{i=1}^{k} \alpha_i g_i : \alpha_i \in \mathbb{F}_2 \right\}$$

- $|\mathscr{C}| = 2^k$, k is called the dimension of the subspace.
  Hence, $k = \log_2 |\mathscr{C}|$.
- Rate of the code $= k/n$.
- This code encodes k bits.
- Encoding is a linear operator, hence implementation is simple.

$$(\alpha_1, \alpha_2, \cdots, \alpha_k) \xrightarrow{\text{encoded}} \sum_{i=1}^{k} \alpha_i g_i$$

$$(\alpha_1, \alpha_2, \cdots, \alpha_k) \xrightarrow{\text{linear}} (\alpha_1, \alpha_2, \cdots, \alpha_k)_{1\times k} G_{k\times n}$$

where, $G_{k\times n} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{pmatrix}$

2.1 Generator matrix

Pick any collection of k linearly independent from $\mathbb{F}_2^n$ $\{g_1, \cdots, g_k\}$.

$$G_{k\times n} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{pmatrix}$$

Rowspace(G) = span(rows of G) = k dimensional subspace of $\mathbb{F}_2^n$

$$d_{min}(\mathscr{C}) = \min_{\underline{c} \neq 0} w_H(\underline{c})$$

Encoding is the operation of mapping $2^n R$ length messages to the n-length codewords in a unique manner. It is the mapping from k-length vectors over $\mathbb{F}_2$ to $\mathscr{C}$.

For linear codes, we can do this encoding as a linear mapping. Encoding operation for linear codes requires polynomial in n, unlike non-linear codes require exponential complexity.

2.2 Example

- Repetition code (eg. 2):

$$G = [1, 1, \cdots, 1]_{1\times n}$$

$$\text{Rowspace}(G) = \{(0, 0, \cdots, 0), (1, 1, \cdots, 1)\}$$

$$d_{min}(\mathscr{C}) = n \qquad dim(\mathscr{C}) = 1 \qquad R = \frac{k}{n} = \frac{1}{n}$$

How can we implement minimum distance decoding more efficiently?

$$\hat{\underline{c}} = argmin_{\underline{c} \in \mathscr{C}} d_H(y, \underline{c})$$

For n=5: Suppose $y = (1\,1\,1\,0\,0)$, then minimum distance decoder output is $\hat{c} = (1\,1\,1\,1\,1)$.

$$MDD(y) = \begin{cases} \underline{0} = (0, \cdots, 0) & w_H(y) < \frac{n}{2} \\ \underline{1} = (1, \cdots, 1) & w_H(y) > \frac{n}{2} \end{cases}$$

This is the majority decoding rule.

3 Binary Hamming code

This is a class of codes, we take up a particular example, let:

$$n = 2^r - 1 \qquad k = 2^r - 1 - r \qquad d = 3 \quad \forall \quad r \geq 3$$

$$\text{if } r = 3 \Rightarrow \qquad n = 7, k = 4, d = 3$$

$$G_{4\times 7} = \left[I_4 : \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \right]$$

(we are appending I_4 with 3 columns to the right)

Note: The 4 rows of G are linearly independent vectors of $\mathbb{F}_2^7$. Rank(G) = number of linearly independent vectors in rows or columns = 4.

$\mathscr{C}$=Rowspace(G) is a 4-dim linear code. Rate= 4/7.

$|\mathscr{C}| = 2^k = 2^4 = 16$

$d_{min}(\mathscr{C}) = 3$