3F7 Information Theory and Coding

Howard Mei

December 29, 2020

1 Probability and Entropy

1.1 Discrete Random Variables

- Probability mass function (pmf): $P_X(x) = Pr(X = x)$. $x \in \mathcal{X}$ is a *realisation* of the random variable x
- Cumulative distribution function (cdf): $F_x(a) = Pr(X \leq a) = \sum_{x \leq a} P_X(x)$
- *Expected value*: $\mathbb{E}X = \sum_a a \cdot P_x(a)$
- *Variance*:$Var(X) = \mathbb{E}[(X - \mathbb{E}X)^2] = \mathbb{E}[X^2] - (\mathbb{E}X)^2$.
- $Var(aX) = a^2 \cdot var(X)$
- A function $g(X)$ of rvX is also an rv
- *Expected value* of functions of random variables : $\mathbb{E}[g(X)] = \sum_a g(a) \cdot P_X(a)$

1.2 Jointly distributed random variables

1.2.1 Discrete rvs X,Y

Marginal distributions :

$$P_X(x) = \sum_y P_{XY}(x, y), \qquad P_Y(y) = \sum_x P_{XY}(x, y)$$

Conditional distribution of Y given X :

$$P_{Y|X}(y|x) = \sum_y P_{XY}(x, y), \quad \text{for x such that} \quad P_X(x) > 0$$

1.2.2 Key properties of jointly distributed rvs

- Product rule:

$$\begin{aligned} P_{XYZ} &= P_X P_{Y|X} P_{Z|YX} \\ &= P_Y P_{Z|Y} P_{X|ZY} \\ &= P_Y P_{X|Y} P_{Z|XY} \\ &= P_Z P_{X|Z} P_{Y|XZ} \\ &= P_Z P_{Y|Z} P_{X|YZ} \end{aligned}$$

- Sum rule (marginalization):

$$P_{XY}(x,y) = \sum_z P_{XYZ}(x,y,z)$$

$$P_X(x) = \sum_{y,z} P_{XYZ}(x,y,z) = \sum_y P_{XY}(x,y)$$

These properties extend naturally to multiple jointly distributed rvs $(X_1, ..., X_n)$

1.2.3 Continuous random variables

- Joint density function $f_{XY}(x,y)$
- $Pr(a \leq X \leq b,\ c \leq Y \leq d) = \int_{x=a}^{b} \int_{y=c}^{d} f_{XY}(x,y)\, dxdy$
- For jointly Gaussian rvs, specified by mean vector and covariance matrix
- Conditional density, product and sum rule analogous to discrete case with density replacing pmf and integrals instead of sums

1.2.4 Independence

Discrete random variables $X_1, ..., X_n$ are *statistically independent* if

$$P_{X_1 ... X_n}(x_1, ...x_n) = P_{X_1}(x_1) \cdot P_{X_2}(x_2) \ ... \ P_{X_n}(x_n) \qquad \forall (x_1, ..., x_n)$$

From *product rule* :

$$P_{X_1 ... X_n}(x_1, ..., x_n) = P_{X_1}(x_1) \cdot P_{X_2|X_1}(x_2|x_1) \ ... \ P_{X_n|X_{n-1} ... X_1}(x_n|x_{n-1}, ..., x_1)$$

Therefore, when independent:

$$P_{X_i|\{X_j\}_{j \neq i}} = P_{X_i}$$

Often, *independent* and *identically distributed* (*i.i.d.*) random variables are considered in this course

1.3 Entropy

1.3.1 Define

The entropy of a discrete random variable X with *pmf* $P4$ is

$$H(X) = \sum_x P(x) \cdot \log_2 \frac{1}{P(x)} \quad \text{bits}$$

- $H(X)$ can be written as $\mathbb{E}[\log_2 \frac{1}{P(x)}]$
- $H(X)$ can be seen as the **uncertainty** associated with the rv X.

1.3.2 Properties

Let X be discrete random variable takes M different values with different probability. Then:

- $H(X) \geq 0$
- $H(X) \leq \log M$
- Equiprobable distribution $(\frac{1}{M}, ..., \frac{1}{M})$ has the maximum entropy equal to $\log M$. Seen as equal probability gives maximum uncertainty in the outcome

Proof:

- For any $x \in \chi,\ 0 \leq P(X) \leq 1\ ,\ \frac{1}{P(X)} \geq 1\ ,$ and hence $\log \frac{1}{P(X)} \geq 0\ , H(X) \geq 0$

  Notes when there is a certain probability of $P(X = a) = 1$, $H(X) = 0$ means no uncertainty in outcome.

- *Proof* of $H(X) \leq \log M$

$$\begin{aligned} H(X) - \log M &= \sum_x P(x) log \frac{1}{P(X)} - \sum_x P(x) \log M \\ &= \frac{1}{\ln 2} \sum_x P(x) \ln \frac{1}{MP(X)} \\ &\leq \frac{1}{\ln 2} \sum_x P(x) \left(\frac{1}{MP(x)} - 1 \right) \quad \text{By inequality rule: } lnx \leq (x-1) \\ &= \frac{1}{\ln 2} \left(\sum_x \frac{1}{M} - \sum_x P(x) \right) = 0 \end{aligned}$$

  Hence $H(X) - \log M \leq 0 \qquad H(X) \leq \log M$

- *Proof* of maximum entropy Equiprobable distribution:

$$\begin{aligned} H(X) &= \log M \text{ when} \\ \ln \frac{1}{MP(X)} &= \left(\frac{1}{MP(x)} - 1 \right) \text{ when} \\ MP(x) &= 1 \\ P(x) &= 1/M \end{aligned}$$

1.3.3 Joint and Conditional Entropy

- The *joint* entropy of X, Y is

$$H(X,Y) = \sum_{x,y} P_{XY}(x,y)\, log \frac{1}{P_{XY}(x,y)}$$

- The *conditional* entropy of Y given X is

$$H(Y|X) = \sum_{x,y} P_{XY}(x,y)\, log \frac{1}{P_{Y|X}(y|x)}$$

- Can be seen as the uncertainty for Y is different for different X, $H(Y|X)$ is the average uncertainty in Y given X

$$H(Y|X) = \sum_x P_X(x) \underbrace{\sum_y P_{Y|X}(y|x)\, log \frac{1}{P_{Y|X}(y|x)}}_{\text{H(Y—X=x) another entropy equation}}$$

- $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$
- When X,Y are independent $H(Y|X) = H(Y)$ as uncertainty of Y is not changed if independent.
- When $Y = f(X)$, $H(Y|X) = H(f(X)|X) = 0$ as we can predict any f(X) from X, no uncertainty. However, inversely $H(X|Y) \geq 0$ zeros only when the function is one to one.

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

1.3.4 Joint Entropy of Multiple RVs

- $$H(X_1, ..., X_n) = \sum_{x_1,..,x_n} P_{X_1..X_n}(x_1, .., x_n)\, log \frac{1}{P_{X_1..X_n}(x_1, .., x_n)}$$
- Chain rule of joint entropy

$$\begin{aligned} H(X_1, ..., X_n) &= H(X_1) + H(X_2|x_1) + H(X_n|X_{n-1}, .., X_1) \\ &= \sum_{i=1}^{n} H(X_i|X_{i-1}, .., X_1) \end{aligned}$$

where the conditional entropy

$$H(X_i|X_{i-1}, .., X_1) = \sum_{x_1,..,x_i} P_{X_1..X_i}(x_1, .., x_i)\, log \frac{1}{P_{X_i|X_1,..,X_{i-1}}(x_i|x_1, .., x_{i-1})}$$

- If independent, then

$$H(X_1, ..., X_n) = \sum_{i=1}^{n} H(X_i)$$

- proof of Chain rule

$$P(x_1, ..., x_n) = P_{X_1}(x_1)P(x_2|x_1)...P(x_n|x_{n-1}, ..., x_1) = \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1)$$

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

2 Law of Large Numbers, Typicality, Data Compression

2.1 Estimating Tail probability

- We want to bound the probability of rare events, corresponding to probability mass in the 'tails' of the pmf/density function

  e.g. What is the **bound** for $P(X > 20)$ for average 5 cars/minute without knowing the distribution?

- Markov and Chebyshev inequalities are ways to bound tail probabilities with limited information.
  - Markov is for non-negative rvs and requires only the mean
  - Chebyshev is for general rvs and requires mean and variance

2.1.1 Markov's Inequality

- For a non-negative rv X and any $a > 0$,

$$P(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

- *Proof:*

$$\mathbb{E}[X] = \sum_{r \geq 0} rP(X = r) = \sum_{0 \leq r \leq a} rP(X = r) + \sum_{r \geq a} rP(X = r)$$

$$\text{For } r \geq a, \quad \sum_{r \geq a} rP(X = r) \geq \sum_{r \geq a} aP(X = r)$$

$$\mathbb{E}[X] \geq \sum_{0 \leq r \leq a} rP(X = r) + \sum_{r \geq a} aP(X = r)$$

$$= \sum_{0 \leq r \leq a} rP(X = r) + aP(X \geq a)$$

$$\mathbb{E}[X] \geq aP(X \geq a)$$

2.1.2 Chebyshev's inequality

- Bound the tail probabilities of deviations around the mean
- For any rv X and $a > 0$,

$$P(|X - \mathbb{E}X| \geq a) \leq \frac{Var(X)}{a^2}$$

- *Proof:*

$$P(|X - \mathbb{E}X| \geq a) = P(|X - \mathbb{E}X|^2 \geq a^2)$$

$$\text{Let } Y = |X - \mathbb{E}X|^2$$

$$\text{Apply Markov's,} \qquad P(Y \geq a^2) \leq \frac{\mathbb{E}Y}{a^2} \qquad \text{Note that } \mathbb{E}Y = Var(X)$$

$$P(|X - \mathbb{E}X| \geq a) \leq \frac{Var(X)}{a^2}$$

2.1.3 Weak Law of Large Numbers (WLLN)

- "Empirical average converges to the mean"
- Let X1,X2, be a sequence of i.i.d. rvs with finite mean μ. $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$
  - Informal WLLN: $S_n \to \mu$ as $n \to \infty$
  - Formal WLLN: For any $\epsilon > 0$, $lim_{n\to\infty} P(|S_n - \mu| \geq \epsilon) = 0$
- *Proof:*

  By Chebyshev's inequality:

$$P(|S_n - \mu| \geq \epsilon) \leq \frac{Var(S_n)}{\epsilon^2}$$

  Also:

$$Var(S_n) = 1/n^2 \cdot Var(\sum_i X_i) = 1/n^2 \cdot \sum_{i=1}^{n} Var(X_i) = 1/n^2 \cdot n\sigma^2 = \frac{\sigma^2}{n}$$

  Hence:

$$P(|S_n - \mu| \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2}$$

2.2 Typical Set

- Simple Example: Consider an i.i.d. Bernoulli($\frac{1}{4}$) source.

$$P(X_i = 1) = \frac{1}{4} \quad P(X_i = 0) = \frac{3}{4} \text{ for } i = 1, 2, 3, ...$$

  1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  2. 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0

  - 16 bits sequence
  - Probability of first sequence = $(\frac{3}{4})^1 6$
  - Probability of second sequence = $(\frac{1}{4})^4(\frac{3}{4})^1 2$
  - The first sequence is $\mathbf{3^4}$ more likely than second!!

- **Typical Sequences**:
  - Though less likely than first sequence, the second is more "typical" of the $(\frac{1}{4}, \frac{3}{4})$ source If $X_1, ...X_n$ are chosen i.i.d. Bernoulli(p), then for large n:
  - With high probability, the fraction of ones observed sequence will be close to p by WLLN
  - With high probability the observed sequence will have probability close to $p^{np} * (1-p)^{n(1-p)}$
  - Any number can be written as $2^{\log a}$ Hence,

$$p^{np} * (1-p)^{n(1-p)} = (2^{\log p})^{np}(2^{\log(1-p)})^{n(1-p)} = 2^{-nH_2(p)}$$

  - For large n, with high probability we will observe a typical sequence . Informally, a typical sequence is one whose probability is close to $2^{-nH_2(p)}$

- **Asymptotic Equipartition Property (AEP)**
  The AEP makes this for any i.i.d. discrete source not just Bernoulli sequences.

  - If $X_1, ...X_n$ are chosen i.i.d. P_x, then for any $\epsilon > 0$

$$\lim_{n\to\infty} Pr\left(\left|\frac{-1}{n}\log P_X(X_1, X_2, ..., X_n) - H(X)\right| < \epsilon\right) = 1$$

  - $\frac{-1}{n}\log P_X(X_1, X_2, ..., X_n)$ is a random variable, a fucntion of rvs is a rv
  - AEP says this rv converges in probability to $H(X)$ a constant, as $n \to \infty$
  - Proof:
    * let $Y_i = -\log P_X(X_i)$
    * Functions of independent rvs are also independent rvs
    * WLLN for Y_i's says that for any $\epsilon > 0$ (Note the $<$ sign replaced the $\geq$)

$$\lim_{n\to\infty} Pr(|\frac{1}{n}\sum_i Y_i - \mathbb{E}[Y_i]| < \epsilon) = 1$$

    * Sum of log become log of multiply:

$$\sum_i Y_i = -\log[P_X(X_1)...P_X(X_n)] = -\log P_X(X_1, X_2, ..., X_n)$$

    * $\mathbb{E}[Y_i] = H(X)$

- **The typical set**

  - Definition:
    The typical set $A_{\epsilon,n}$ with respect to P is the set of sequence $x_1, ..., x_n \in \chi^n$ with the property

$$2^{-n(H(X)+\epsilon)} \leq P(x_1, ..., z_n) \leq 2^{-n(H(X)-\epsilon)}$$

    "Sequences with probability concentrated around $2^{-n(H(X)}$"
  - Dependence on n and ϵ A sequence belonging to the typical set is called an ϵ-type sequence

- Properties of the **Typical Set**

  - Property 1:
    * if $X^n = (X_1, ..., X_n)$ is generated i.i.d. P, then

$$Pr(X^n \in A_{\epsilon,n}) \xrightarrow{n\to\infty} 1$$

    * The sequence $X_1, , ..., X_n$ observed us very likely to belong to the typical set.

* Proof:

$$X^n = (X_1, ..., X_n) \Leftrightarrow 2^{-n(H(X)+\epsilon)} \leq P(X^n) \leq 2^{-n(H(X)-\epsilon)}$$
$$\Leftrightarrow H(X) - \epsilon \leq -\frac{1}{n} \log P(X^n) \leq H(X) + \epsilon$$

By AEP:

$$Pr(X^N \in A_{\epsilon,n}) = Pr\left(H(X) - \epsilon \leq -\frac{1}{n} \log P(X^n) \leq H(X) + \epsilon\right) \xrightarrow{n\to\infty} 1$$

- Property 2:
  * Let $|A_{\epsilon,n}|$ denote the number of elements in the typical set $A_{\epsilon,n}$. Then:

$$|A_{\epsilon,n}| \leq 2^{n(H(X)+\epsilon)}$$

  * Proof:

$$\begin{aligned} 1 &= \sum_{x^n \in \mathcal{X}^n} P(x^n) \\ &\geq \sum_{x^n \in A_{\epsilon,n}} P(x^n) \\ &\geq \sum_{x^n \in A_{\epsilon,n}} 2^{-n(H(X)+\epsilon)} \\ &= 2^{-n(H(X)+\epsilon)} |A_{\epsilon,n}| \end{aligned}$$

- Property 3:
  * For sufficiently large n, $|A_{\epsilon,n}| \geq (1-\epsilon)2^{n(H(X)-\epsilon)}$
  * Proof: from Property 1, $Pr(X^n \in A_{\epsilon,n}) \xrightarrow{n\to\infty} 1$
    This means for any $\epsilon > 0$, $Pr(X^n \in A_{\epsilon,n}) > 1 - \epsilon$ for sufficiently large n:

$$\begin{aligned} 1 - \epsilon &< Pr(X^n \in A_{\epsilon,n}) \\ &= \sum_{x^n \in A_{\epsilon,n}} P(x^n) \\ &\leq \sum_{x^n \in A_{\epsilon,n}} 2^{-n(H(X)-\epsilon)} \\ &= 2^{-n(H(X)-\epsilon)} |A_{\epsilon,n}| \end{aligned}$$

- Summary of the properties:
  * Suppose generated $X_1, ..., X_n$ i.i.d. P. With high probability, the sequence will be typical, i.e., its probability is close to $2^{-n(H(X)}$
  * If the pmf P assigns non-zero probabilities to the symbols denoted a,b,c etc., the typical set is essentially the set of sequences whose fraction of a's is close to $P(a)$, fraction of b's is close to $p(b)$, and so on
  * The number of typical sequences is close to $2^{n(H(X)}$

2.3 Compression

GOAL: To compress a source producing symbols $X_1, X_2, ... \in \chi$ that are i.i.d P.

- To each source sequenceX_n, the code assigns a *unique* binary sequence $c(X^n)$
- $c(X^n)$ is called the *codeword* for the source sequence $LetX^n$.
- $l(X^n)$ be the length of the codeword assined to X^n i.e., the number of bits in $c(X^n)$
- The expected code length is defined as:

$$\mathbb{E}[l(X^n)] = \sum_{x^n} P(x^n) l(x^n)$$

2.3.1 A Naive Compression Code

English: 26 chars $/H(X) = 4$ bits

- List all the $|\chi|^n$ possible length n sequences
- Index these as $\{0, 1, ..., |\chi|^n - 1\}$ using $\lceil \log |\chi|^n \rceil$ bits. English - 5n bits
- Expected code length $\mathbb{E}[l(X^n)] = n \log |\chi|$
- Expected number of bits/symbol: $\mathbb{E}[l(X^n)]/n = \log |\chi|$ English - 5 bits per symbol

How can we do better?

2.3.2 Compression via the Typical Set

- Compression scheme:
  - At most $2^{n(H(X)+\epsilon)}$ $\epsilon-$ typical sequences.
  - Index each sequence in $A_{\epsilon,n}$ using $\lceil \log 2^{n(H(X)+\epsilon)} \rceil$ bits. Prefix each of these by a flag bit 0.

$$\text{Bits/typical seq.} \quad = \lceil n(H(X) + \epsilon) \rceil + 1 \leq n(H(X) + \epsilon) + 2$$

  - Index each sequence not in $A_{\epsilon,n}$ using $\lceil \log |\chi|^n \rceil$ bits. Prefix each of these by a flag bit 1.

$$\text{Bits/non-typical seq.} \quad = \lceil n \log |\chi| \rceil + 1 \leq n \log |\chi| + 2$$

- Expected code length:

$$\begin{aligned}
\mathbb{E}[l(X^n)] &= \sum_{x^n} P(x^n) l(x^n) \\
&= \sum_{x^n \in A_{\epsilon,n}} P(x^n) l(x^n) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n) l(x^n) \\
&\leq \sum_{x^n \in A_{\epsilon,n}} P(x^n)(n(H(X) + \epsilon) + 2) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n)(n \log |\chi| + 2) \\
&\leq 1 \cdot n(H(X) + \epsilon) + \epsilon \cdot n \log |\chi| + 2 \left(\sum_{x^n \in A_{\epsilon,n}} P(x^n) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n) \right) \\
&= n(H(X) + \epsilon) + \epsilon n \log |\chi| + 2 \\
&= n(H(X) + \epsilon')
\end{aligned}$$

- Fundamental limit of Compression
  - For n sufficiently large, there exists a code that maps sequences x^n of length n into binary strings such that the mapping is one-to-one and
$$\mathbb{E}[\frac{1}{n}l(X^n)] \leq H(X) + \epsilon$$
  - In fact the expected length of any uniquely decodable code satisfies
$$\mathbb{E}[\frac{1}{n}l(X^n)] \geq H(X)$$
  - Hence entropy is the fundamental limit of loseless compression

3 Prefix-free codes, Kraft inequality, Loseless source coding theorem

3.1 Prefix-free codes

- Definition: A code is called prefix-free or instantaneously decodable if no codeword is the prefix of another.
  1. $\{0, 00, 10\}$? No
  2. $\{0, 10, 11\}$? Yes
  3. $\{00, 010, 011, 0101, 1\}$? No
- Extension Codes and Unique Decodability
  - Given a binary code C for alphabet χ, the extension code C^n is the code applied symbol-by-symbol to strings:
$$C(x_1x_2...x_n) = C(x_1)C(x_2)...C(x_n)$$
  - The extension code C^n is uniquely decodable if for each binary codeword in it, there is only one possible source string can produce it
  - Prefix-free codes are uniquely decodable, but not all uniquely decodable codes are prefix-free
- We will focus only on designing and analysing prefix-free codes as we want fast encoding and decoding algorithms.

3.2 Kraft Inequality

- The number of leaves at a level is 2^l
- $$\sum_{i=1}^{N} 2^{l_{max}-l_i} \leq 2^{l_{max}} \Rightarrow \sum_{i=1}^{N} 2^{-l_i} \leq 1$$
- Derived from: Total number of unusable codes at level max ¡ Total number of codes at level max. While each codeword of length l leads to s set of $2^{l_{max}-l_i}$ unusable leaves at depth $2^{l_{max}}$
- A necessary condition for any prefix-free code with length$\{l_1...l_N\}$
- A sufficient condition, if a set of l length satisfy it, we can always construct a prefix-free code with these length.

3.3 Coding theorem for a random variable

Let X be a random variable taking values in χ with entropy $H(X)$ The expected codeword length L of any binary prefix-free code for X satisfies

$$L \geq H(X)$$

Proof: Denote the probability of symbols as $p_1, p_2, ...$ and the corresponding codeword length as $l_1, l_2, ...$

$$\begin{aligned}
H(X) - L &= \sum_i p_i \log_2(1/p_i) - \sum_i p_i l_i \\
&= \sum_i p_i \log_2(2^{-l_i}/p_i) \\
&= \frac{1}{\ln 2} \sum_i p_i \ln_2(2^{-l_i}/p_i) \\
&\leq^{(a)} \frac{1}{\ln 2} \sum_i p_i \left(2^{-l_i}/p_i - 1\right) \\
&= \frac{1}{\ln 2} \left(\sum_i 2^{-l_i} - \sum_i p_i \right) \\
&\leq^{(b)} \frac{1}{\ln 2}(1-1) = 0
\end{aligned}$$

(a) $\ln(x) \leq x - 1$
(b) Kraft's inequality

3.4 Coding theorem for blocks of source symbols

Instead of assigning codeword to each source symbol, we want to assign codeword to blocks of N source symbols.

Denoting the block by $X^N := (X_1, ..., X_N)$ Hence the Expected length of the code satisfies:

$$E[l(X^N)] \geq H(X^N) \qquad \frac{E[l(X^N)]}{N} \geq \frac{H(X^N)}{N}$$

If X is an iid source, then

$$H(X^N) = NH(X) \rightarrow \frac{E[l(X^N)]}{N} \geq H(X)$$

Remarks:

1. We showed for iid source any prefix-free code has average code length :

$$\frac{E[l(X^N)]}{N} \geq H(X)$$

This is also true for any uniquely decodable code, since all we used is Kraft's inequality which holds for uniquely decodable codes as well.

2. From fundamental limit of compression we have:

$$\mathbb{E}[\frac{1}{N}l(X^N)] \leq H(X) + \epsilon$$

for sufficiently large N

Above together form **Shannon's lossless source coding theorem for iid sources**

- You can construct a uniquely decodable code with expected length arbitrarily close to the entropy (By taking block length N large enough)
- Conversely, you cannot construct a uniquely decodable code with expected length than the source entropy.

4 Shannon-Fano coding, Huffman coding, Arithmetic coding

4.1 Shannon-Fano Coding

- Expected code length $L \geq H(X)$

$$L = \sum_{i=1}^{m} p_i l_i \geq \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$$

  - When can this inequality become an equality?
  - How do we pick lengths to make it as tight as possible

- Since l_i are integers, an obvious way to choose them is:

  -

$$l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$$

  - Note that $x \leq \lceil x \rceil < x + 1$ Therefore

$$L = \sum_i p_i l_i < \sum_i p_i \left(\log_2 \frac{1}{p_i} + 1 \right) = H(X) + 1$$
$$L < H(X) + 1$$

- Verify Shannon-Fano Coding satisfy Kraft inequality

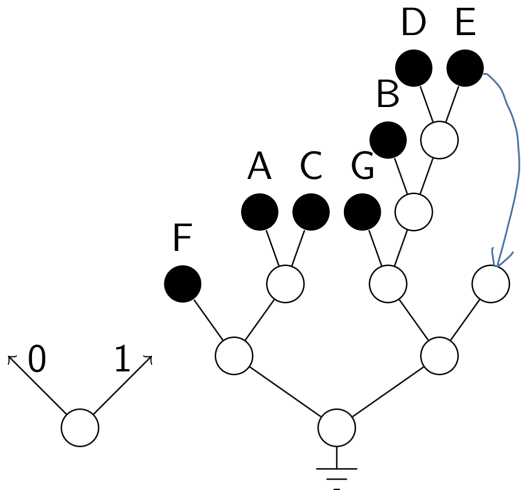
$$\sum_i 2^{l_i} = \sum_i 2^{-\left\lceil \log_2 \frac{1}{p_i} \right\rceil} \leq \sum_i 2^{-\log_2 \frac{1}{p_i}} = \sum_i p_i = 1$$

- Clearly, we can do better by pushing B/D/E to the length two

4.2 Properties of Optimal prefix-free code

1. The lengths are ordered inversely with the probabilities, if $p_j > p_k$, then $l_j \leq l_k$
2. The two last probable symbols have the same length and are on neighboring leaves in the binary tree (i.e. they differ only in the last digit).

x	p_i	$\log(1/p_i)$	ℓ_i
A	0.15	2.73	3
B	0.07	3.83	4
C	0.17	2.55	3
D	0.06	4.05	5
E	0.06	4.05	5
F	0.31	1.68	2
G	0.18	2.47	3



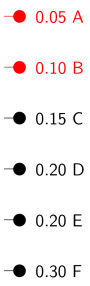
4.3 Huffman Coding

- An algorithm gives an optimal prefix-free code for a given set of probabilities.

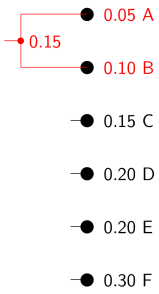
  1. Take the two least probable symbols in the alphabet. these two symbols will be given the longest codewords, which will have equal length, and differ only in the last digit
  2. Combine these two symbols into a single symbol, and repeat.

- A source produces symbols from the set {A,B,C,D,E,F} with probabilities {0.05,0.1,0.15,0.2,0.2,0.3}

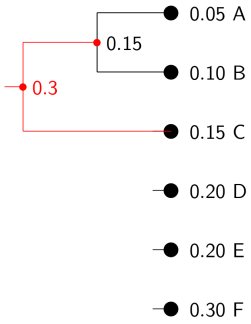
  1. List these symbols in increasing order of their probabilities.



(a) Step 1

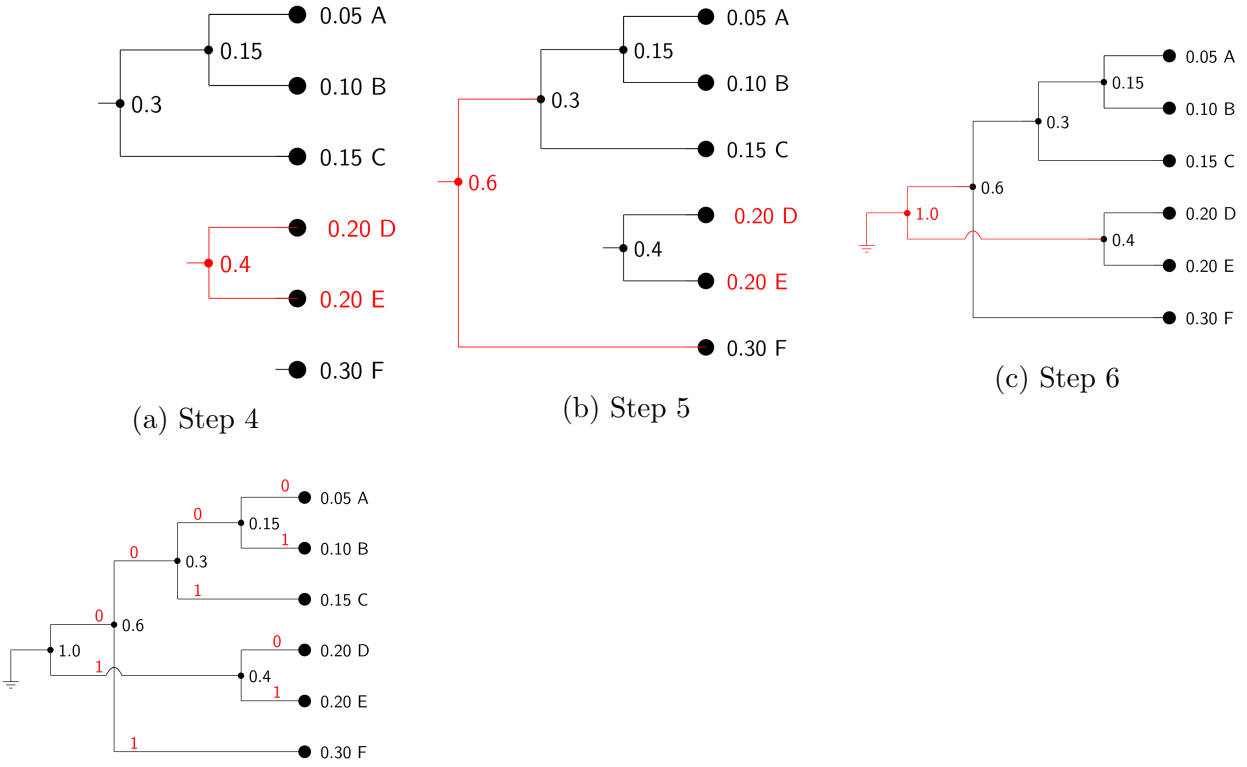


(b) Step 2



(c) Step 3

  2. Combine the two simple with the smallest probabilities, and form a "super-symbol" with the sum of their probabilities.
  3. We now have five symbols with probabilities {0.15,0.15,0.2,0.2,0.3}. Again combine the two symbols with the smallest probabilities...
  4. Among the four remaining symbols, D,E have the smallest probabilities, so combine
  5. Among the three remaining symbols, the smallest probabilities are 0.3,0.3, so combine them...
  6. Finally, combine the remaining two symbols
  7. The symbols are the leaves of a tree. The final step is to assign the codewords to the symbols using the tree.



(a) Step 4 (b) Step 5 (c) Step 6

Figure 3: Assign codes

The Huffman code is:

$$F \to 01, E \to 11, D \to 10, C \to 001, B \to 0001, A \to 0000$$

- Properties of the Huffman Code
  1. For a source with alphabet of size m, the Huffman algorithm requires $m-1$ steps of combining the two smallest probabilities at each stage.
  2. The Huffman code for a given source may not be unique: Swapping 0/1, also 3 same prob can take any two. However, the expected code length should be the same.
- Optimality of Huffman coding: For a given set of probabilities, there is no prefix-free code that has smaller expected length than the Huffman code.
- Huffman codes are optimal for coding a single random variable X, and have expected code length that is less than $H(X) + 1$ But they have some weaknesses:
  - To reduce this overhead of up to 1 bit/symbol, we could design a Huffman code for blocks of k symbols. This would give us an overhead of 1 bit/k symbols, or 1/k bits/symbols.
  - But this comes with the expense of increased complexity. For blocks of k symbols, the binary tree is much larger.
  - These defects are addressed by Arithmetic coding, a scalable algorithm whose expected code length is very close to the source entropy for large sequences. It can also easily deal with non-iid source like text.

4.4 Interval Coding

- Key idea: Each symbol can be represented as an interval inside $[0, 1]$ with length of the interval equal to the symbol probability.

  A source with m symbols with probabilities $\{p_1, ..., p_m\}$ is represented using m intervals

$$[0, p_1), [p_1, p_1 + p_2), ..., [\sum_{i=1}^{m-1} p_i, \sum_{i=1}^{m-1} p_i + p_m).$$

- Example: To represent interval $[0.17, 0.43)$ convert the end-points to binary and find a binary interval lies completely inside this interval. say $[010, 011]$ is the interval of $[0.25, 0.375]$ We choose **010** as the codeword for $[0.17, 0.43)$.

- In general, the binary codeword for a symbol with probability p represented by the interval $[a, a+p)$ can be obtained as follows:

  1. Find the largest *dyadic interval* of the form $[\frac{j}{2^l}, \frac{j+1}{2^l})$ that lies within $[a, a + p)$
  2. Take the binary representation of the lower end-point of the dyadic interval as the codeword.

- Code length: With l code bits, the dyadic intervals have length 2^l. The dyadic interval has to be contained within an interval of length p. Hence

$$2^l \leq p \rightarrow \lceil \log_2(1/p) \rceil \leq l$$

  But sometimes we'll need $l = \lceil \log_2(1/p) \rceil + 1$
  The expected code length can therefore be bounded as

$$L = \sum_i p_i l_i \leq \sum_i p_i (\lceil \log_2(1/p_i) \rceil + 1) < H(X) + 2$$

- Performance: In terms of expected code length, the analysis above shows that interval codes are in general not as good as Huffman codes or even Shannon-Fano codes. But interval coding is the basis for arithmetic coding, which is a powerful technique for long source sequences.

4.5 Arithmetic Coding

- Explain with an example. Consider a source producing symbols $X_1, X_2, ..., X_n$ which are i.i.d., with each X_i taking values in $\{a, b, c\}$ with probabilities $\{0.2, 0.45, 0.35\}$

- Key ideas:

  - Each length n string $(x_1, ..., x_n)$ is represented by a disjoint interval with length equal to the probability of the string.
    - E.g. for $n = 2$, $X_1 = b, X_2 = c$ corresponds to the probability of length $0.45 * 0.35 = 0.1575$ and $X_1 = b, X_2 = a$ of length $0.45 * 0.2 = 0.09$.
  - The interval for $(X_1 = x, X_2 = y)$ is a sub interval of the interval for $X_1 = x$
    - E.g. $X_1 = b$ is the interval $[0.2, 0.65)$ and $(X_1 = b, X_2 = c)$ is the subinterval of length 0.1575 within $[0.2, 0.65)$ To calculate, we rewrite the interval as $\{0, 0.2, 0.65, 0.1\}$ as the order of a,b,c.

* The subinterval becomes $0.2+(0.65-0.2)*0.65 \rightarrow 0.2+(0.65-0.2)*1$ That is $[0.4925, 0.65)$ Also we can verify that $0.65 - 0.4925 = 0.1575$

– Similarly, for any symbols x, y, z, $P(X_1 = x, X_2 = y, X_3 = z)$ is a sub interval of $P(X_1 = x, X_2 = y)$, and so on.

- Decoding: Using binary codeword to sequentially zoom in to the interval, decoding symbols as you go along.

- Expected code length:
Arithmetic coding can be performed in any sequence of length n, so that any sequence $x_1, ..., x_n$ can be represented by an interval of length $p(x_1, ..., x_n)$, which gives a binary codeword of length at most $\lceil \log_2 \frac{1}{p(x_1,...,x_n)} \rceil + 1$.
Therefore the expected code length for length n sequences is bounded as:

$$L_n = \sum_{X_n} p_{x_n} l_{x_n} \leq \sum_{x_n} p_{x_n} (\log_2 (1/p(x_1, ..., x_n)) + 2) = H(X^n) + 2$$

Therefore the expected code length per symbol is

$$\frac{L_n}{n} < \frac{H(X^n)}{n} + \frac{2}{n} = H(X) + \frac{2}{n},$$

Where the last equality hold for *iid* P_X

4.6 Arithmetic coding for non-iid sources

- Consider a source that produces symbols in alphabet $\{a_1, ..., a_m\}$ with a known distribution

$$P(x_1)p(x_2|x_1)...P(x_n|x_1, ..., x_{n-1})$$

The arithmetic coding algorithm can be easily extended to such sources. As before, consider the source with alphabet $\{a, b, c\}$ with probabilities $\{0.2, 0.45, 0.35\}$.
Suppose that three conditional distributions $P(X_2|X_1 = a)$, $P(X_2|X_1 = b)$, $P(X_2|X_1 = c)$ and say

$$P(X_2 = a|X_1 = b) = 0.5, P(X_2 = b|X_1 = b) = 0.3, P(X_2 = c|X_1 = b) = 0.2$$

We do the same thing as before, just change the probability using the conditional probability accordingly.

- Coding Algorithm:

  – Let source alphabet $\mathcal{A}$ be $\{a_1, ..., a_m\}$
  – We are given a source sequence $x_1, x_2, ..., x_n$ where each $x_i \in \mathcal{A}$
  – Both encoder and decoder know the conditional distributions $P_{X_1}, P_{X_2|X_1}, ..., P_{X_n|X_1,...,X_{n-1}}$.
  – The encoding algorithm computes the interval using the following lower and upper cumulative probabilities. For $i = 1, ...m$ and for $k - 1, ..., n$ we define

$$L_k(a_i|x_1, ..., x_{k-1}) = \sum_{i'}^{i-1} P(X_k = a_{i'}|X_1 = x_1, ..., X_{k-1} = x_{k-1})$$

$$U_k(a_i|x_1, ..., x_{k-1}) = \sum_{i'}^{i} P(X_k = a_{i'}|X_1 = x_1, ..., X_{k-1} = x_{k-1})$$

- Finite precision issues: The arithmetic encoding algorithm above assumes an infinite precision computer:
  - As n grows, the length of the interval corresponding to a sequence $(x_1, ..., x_n)$ shrinks
  - Hence the number of digits needed to accurately store the values lo and hi grows with n.

Summary:

1. Arithmetic coding can achieve compression very close to the source entropy, with complexity scaling linearly with the length of the sequence.
2. It does require you to know the conditional distribution. But this approach fits well with machine learning techniques that can mine huge quantities of text/speech/video data to build good probabilistic models.
3. Note that the assumed distribution of the source doesn't need to be true one, it only needs to be the same at both encoder and decoder.
4. Arithmetic coding works even when you generate the source conditional distribution on the fly, based on what has been observed so far. Given the generating rule, the decoder will also generate required conditional distributions as it reconstructs the sequence.

5 Relative Entropy Mutual Information

5.1 Relative Entropy

5.1.1 Define

The ***Relative Entropy*** or the *Kullback-Leibler(KL)* divergence between two pmfs P and Q is:

$$D(P||Q) = \sum_{x \in \chi} P(x) \log \frac{P(x)}{Q(x)}$$

- P and Q are defined on the same alphabet χ
- Measure of distance between distributions P and Q
- Not a true distance. For example: $D(P||Q) \neq D(Q||P)$
- if $P = Q$ Then $D(P||Q) = 0$
- Example: If $P = Bern(p)$ and $Q = Bern(q)$ for $p, q \in [0, , 1]$,

$$D(P||Q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

- Relative Entropy is always non-negative:

$$D(P||Q) \geq 0 \text{With equality if and only if } P = Q$$

Proof: Using $\log_2 a = \frac{\ln a}{\ln 2}$

$$-D(P||Q) = \frac{1}{\ln 2}\sum_{x\in\mathcal{X}} P(x)\ln\frac{Q(x)}{P(x)}$$
$$\leq \frac{1}{\ln 2}\sum_{x\in\mathcal{X}} P(x)\,(fracQ(x)P(x)) \qquad = \frac{1}{\ln 2}(1-1) = 0$$

Again, we use our favorite inequality $\ln x \leq (x-1)$ with equality iff x=1

- We look at applications in compression and hypothesis testing.

5.1.2 Redundancy in Source Coding

Often the True distribution of the source is unknown, and we have to work with estimated source distribution for compression.

- Suppose true pmf of a rv is $P = \{p_1, ..., p_m\}$, estimated pmf is $\hat{P} = \{\hat{p}_1, ..., \hat{p}_m\}$
- $\hat{P}$ is used to design a compression code whose code length l_i satisfy

$$l_i = \log(1/\hat{p}_i) \qquad \text{for } i = 1, ..., m.$$

The code is optimal for the distribution $\hat{p}$

- How far are we from the optimal of the true distribution? The Average code Length:

$$\begin{aligned} L = \sum_i p_i l_i &= \sum_i p_i \log\frac{1}{\hat{p}_i} \\ &= \sum_i p_i \log\frac{p_i}{p_i\hat{p}_i} \\ &= \sum_i p_i \log\frac{1}{p_i} + \sum_i p_i \log\frac{p_i}{\hat{p}_i} \\ &= H(P) + D(P||\hat{p}_i) \quad \text{bits/symbol} \end{aligned}$$

- Therefore $D(P||\hat{p}_i)$ is the Price you pay in bits/symbol –Redundancy– for designing the code with estimated distribution rather than true distribution.
- Example: Suppose we know that the true distribution P is one from a set of distributions $\mathcal{P}$. For a ternary source that came from the set of distributions of the form $\mathcal{P} = \{\gamma, 0.2, 0.8-\gamma\}$for $\gamma \in (0, 0.8)$.

  Then design a code using a distribution $\hat{P}$ that minimizes the *worst-case* redundancy over this class of distribution $\mathcal{P}$, i.e.,

$$\text{Choose } \hat{P} \text{ to minimize } \max_{P\in\mathcal{P}} D(P||\hat{P})$$

$$R^* = \min_{\hat{P}}\max_{P\in\mathcal{P}} D(P||\hat{P})$$

is called *minimax* redundancy

5.1.3 Hypothesis Testing

We have data $X_1, ..., X_n$, and the knowledge that one of the following is true.

$$H_0 : X_1, ..., X_n \sim \text{i.i.d. } P$$

$$H_1 : X_1, ..., X_n \sim \text{i.i.d. } Q$$

Where H_0 is called the *null hypothesis*

Some definitions

- A decision rule or a test is a function that maps the data $(X_1, ..., X_n)$ to a binary decision (0 for H_0, 1 for H_1).
- With any decision rule, two kinds of errors can be made:
  Type I error: When H_0 is true, but the decision rule chooses H_1.
  Type II error: When H_1 is true, but the decision rule chooses H_0.
- Given the data $X_1, ..., X_n$, the *likelihood ratio*(LR) is defined as

  $$\frac{Q(X_1, ...X_n)}{P(X_1, ...X_n)}$$

  and the normalized *log-likelihood ratio*(LLR) is

  $$\frac{1}{n} \log \frac{Q(X_1, ...X_n)}{P(X_1, ...X_n)}$$

- The optimum decision rule is a likelihood-ratio thresholding rule,i.e.,
  For some threshold T:
  Choose H_1 if $\frac{Q(X_1,...X_n)}{P(X_1,...X_n)} \leq T$: otherwise choose H_0
  Equivalently for LLR
  Choose H_1 if $\frac{1}{n} \log \frac{Q(X_1,...X_n)}{P(X_1,...X_n)} \leq t$: otherwise choose H_0
  The intuition is that Q need to comparably larger than P to be valid.
  Note that both tests are equivalent by setting $T = 2^{nt}$
- Increase threshold t, the condition to choose H_1 becomes more stringent; Hence type-I error prob. increases, type-II error prob. decreases.
- If we have the optimality of thresholding tests, then:
  If a thresholding test has probability of Type-I error α and prob. of type-II error β. Then any test that lower α must increase β

Examples: Under $H_0 : X_1, ..., X_n \sim$ i.i.d. Bern(p); $H_1 : X_1, ..., X_n \sim$ i.i.d. Bern(q) (For Bern, 1=p or q, 0= (1-p) or (1-q)). Denote number of ones as k_n, we have:

$$\begin{aligned} LLR(X_1, ...X_n) &= \frac{1}{n} \log \frac{Q(X_1, ...X_n)}{P(X_1, ...X_n)} \\ &= \frac{1}{n} \log \frac{q^{k_n}(1-q)^{n-k_n}}{p^{k_n}(1-p)^{n-k_n}} \\ &= \frac{k_n}{n} \log \frac{q}{p} + \left(1 - \frac{k_n}{n}\right) \log \frac{1-q}{1-p} \end{aligned}$$

Question: What is the behaviour of the LLR as n gets large when: a) H_0 is true; b) H_1 is true

- H_0: $X_i \sim$ i.i.d. Bern(p)

$$\frac{k_n}{n} \to p \quad LLR \to p \log \frac{q}{p} + (1-p) \log \frac{1-q}{1-p} = -D(P||Q)$$

  - When the true distribution is P, the likelihood of the data under the Q distribution(Wrong) (for large n) is

$$Q(X_1, ...X_n) = P(X_1, ...X_n) 2^{-nD(P||Q)}$$

- H_1: $X_i \sim$ i.i.d. Bern(q)

$$\frac{k_n}{n} \to q \quad LLR \to q \log \frac{q}{p} + (1-q) \log \frac{1-q}{1-p} = D(Q||P)$$

  - When the true distribution is q, the likelihood of the data under the P distribution(Wrong) (for large n) is

$$P(X_1, ...X_n) = Q(X_1, ...X_n) 2^{-nD(Q||P)}$$

- The likelihood of the data under the wrong distribution is **exponentially smaller** than the likelihood under the true distribution.
- The exponent is controlled by the relative entropy, but note the asymmetry between $D(P||Q)$ and $(Q||P)$

5.2 Mutual Information

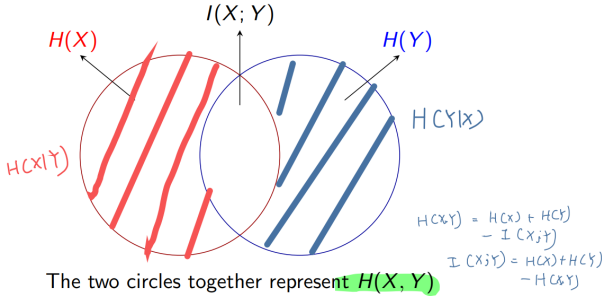
- Consider two random variables X and Y with joint pmf P_{XY}. The mutual information between X and Y is defined as

$$I(X;Y) = H(X) - H(X|Y) \quad \text{bits}$$

- **Reduction in the uncertainty of X when you observe Y**
- **Property 1**

$$\begin{aligned} I(X;Y) &= H(X) - H(Y) - H(X,Y) \\ &= H(X) + H(Y) - (H(X) + H(Y|X)) \\ &= H(Y) - H(Y|X) \\ I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

X says as much about Y as Y says about X



- I(X;Y) when X and Y are independent: 0 - No reduction in uncertainty
- I(X;Y) when Y=X: H(X) - Full reduction in uncertainty
- I(X:Y) when Y=f(X): H(X) - H(X—f(x))

- **Property 2**

$$I(X;Y) = D(P_{XY}||P_X P_Y)$$

The relative entropy between the joint pmf and the product of the marginals
Proof:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= -\sum_x P_X(x) \log P_X(x) + \sum_{x,y} \log P_{X|Y}(x|y) \\ &= \sum_{x,y} \log \frac{P_{X|Y}(x|y)}{P_X(x)} \\ &= \sum_{x,y} \log \frac{P_{X|Y}(x|y) P_Y(y)}{P_X(x) P_Y(y)} \\ &= D(P_{XY}||P_X P_Y) \end{aligned}$$

- **Property 3**

$$I(X;Y) \geq 0$$
$$\rightarrow H(X|Y) \leq H(X), \quad H(Y|X) \leq H(Y)$$

Knowing another random variable Y can only reduce the average uncertainty in X

Proof: from Property 2 and $D(P||Q) \geq 0$

5.2.1 Conditional Mutual Information

Given X, Y, Z jointly distributed according to P_{XYZ}, the conditional mutual information $I(X:Y|Z)$ is defined as:

$$I([X:Y]||Z) := H(X|Z) - H(X|Y,Z).$$

5.2.2 Chain Rule

For any sequence of random variables: $X_1, X_2, ..., X_n$ jointly distributed with Y,

$$I(X_1, X_2, ..., X_m; Y) = \sum_{i=1}^{n} I(X_i; Y|X_{i-1}, X_{i-2}, ..., X_1),$$

By applying the conditional mutual information rule, for each i:

$$I(X_i; Y|X_{i-1}, X_{i-2}, ..., X_1) = H(X_i|X_{i-1}, X_{i-2}, ..., X_1) - H(X_i|Y, X_{i-1}, X_{i-2}, ..., X_1)$$

6 Discrete Channels and Channel Capacity

6.1 Binary Channel

6.1.1 Binary Symmetric Channel(BSC)

From Fig 4,

$$P(Y=0|X=0)=1-p, \qquad P(Y=1|X=1)=1-p$$
$$P(Y=1|X=0)=p, \qquad P(Y=1|X=0)=p$$

- p is the "Crossover probability"; the channel is BSC(p)
- We need to find a good design of error-correcting codes for the BSC.

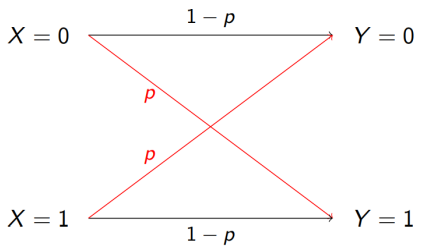


Figure 4: Binary Symmetric Channel

Example: Repetition Code

- For a BSC of p=0.1, use (1,3) repetition code:

(1, 3) Repetition Code

Data:	0	1	1	0	0 . . .
Coded bits:	000	111	111	000	000 . . .
Received bits:	001	101	111	011	000 . . .
Decoded bits:	0	1	1	1	0 . . .

Figure 5: (1,3)Repetition coding for BSC

- Data bit wrongly decoded if channel flips two or more bits
- Probability of decoding error: $\binom{3}{2}(0.1)^2(0.9)+\binom{3}{3}(0.1)^3=0.028$
- Data rate $=\frac{1}{3}$ bits/transmission If we apply a (1,9) or even longer repetition code:
- Probability of error $=0.0009$ or goes to 0, but...
- Data rate $=\frac{1}{9}$ or $\frac{1}{n}$, which also goes to 0

6.2 Discrete Memory less Channel

- A DMC is a system consisting of an input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$ and a set of transition probabilities

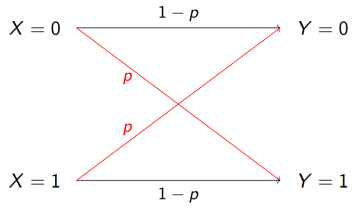
$$P_{Y|X}(b|a) = Pr(Y = b|X = a) \quad \text{for all } a \in \mathcal{X} \text{ and } b \in \mathcal{Y}$$

- Memoryless means that the channel acts independently on each input. For each time instant $k = 1, 2, ...$

$$Pr(Y_k = y|X_k = x, X_{k-1}, ..., X_1, Y_{k-1}, ..., Y_1) = P_{Y|X}(y|x)$$

Given all the past, the current output depends only on the current input

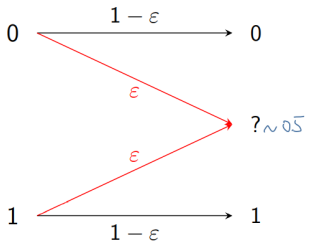
- Example:Binary Symmetric Channel



  - A DMC can be described by a transition probability matrix, e.g.

		Y	
	$P_{Y\|X}$	0	1
X	0	$1-p$	p
	1	p	$1-p$

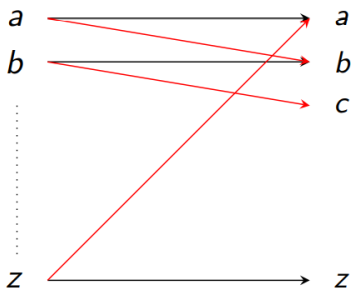
- Example: Binary Erasure Channel (BEC):



		Y		
	$P_{Y\|X}$	0	?	1
X	0	$1-\varepsilon$	ε	0
	1	0	ε	$1-\varepsilon$

  - Erasure can model packet loss in networks
  - When the demodulator thinks the output symbol is too noisy, it can declare an erasure

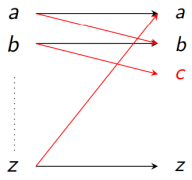
- Noisy Keyboard Channel: A useful toy example
  - Input alphabet of 26

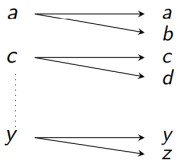


– Each channel input is either received unchanged or transformed into next symbol:

$$P(Y = a|X = a) = P(Y = b|X = a) = \frac{1}{2},$$
$$P(Y = b|X = b) = P(Y = c|X = b) = \frac{1}{2},$$
$$\vdots$$
$$P(Y = z|X = z) = P(Y = a|X = z) = \frac{1}{2}$$

To achieve error-free over this channel:





– Use only symbols a,c,e,...,y

– Noiselessly convey one of 13 symbols per transmission

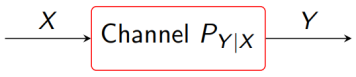
– $\Longrightarrow$ Transmission rate $= \log 13$ bits/transmission

– This is in fact the maximum possible rate

For a general DMC:

- Construct a set of input sequences which have **Non-Intersecting** sets of output sequences with high probability
- These input sequences are non-confusable as to the inputs a,c,e... in the keyboard channel.

6.3 Channel Capacity



The channel capacity of a discrete memoryless channel is defined as

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

- $P_{Y|X}$ is defined by the DMC. Each input distribution P_X yields a differeent joint distribution on (X,Y) given by $P_X P_{Y|X}$
- C is the maximum transmission rate over the DMC for arbitrarily small probability of error

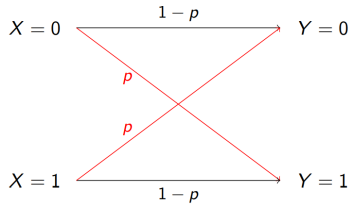
Example 1: Noiseless Binary Channel

$$I(X;Y) = H(X) - H(X|Y) = H(X) \quad (H(X|Y) = 0)$$

- Channel capacity is maximize mutual information, so here maximize H(X) $\rightarrow P_X = (\frac{1}{2}, \frac{1}{2})$
- Therefore

$$C = \max_{P_X} I(X;Y) = \max_{P_X} H(X) = 1\text{bit/transmission}$$

Example 2: BSC



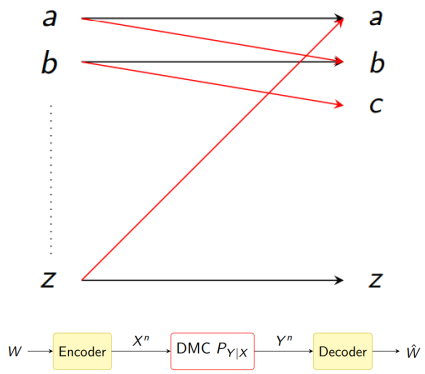
$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - \sum_{x \in 0,1} P_X(x) H(Y|X = x) \\ &= H(Y) - \sum_{x \in 0,1} P_X(x) H_2(P) \\ &= H(Y) - H_2(P) \\ &\leq 1 - H_2(P) \end{aligned}$$

- $H(Y|X = x) = H_2(P)$ since $H(Y|X = 0) = H((Y|X = 1) = H\{1 - p, p\} = H\{p, 1 - p\}$
- Maximum value of H(Y) = 1 is attained when $P_X = (0.5, 0.5) \rightarrow C = 1 - H_2(p)$
- For p = 0.1, C = 0.531 bits/transmission

Example 3 : Noisy Keyboard Channel

$$I(X;Y) = H(Y) - H(Y|X) \qquad = H(Y) - 1 \leq \log(26) - 1 = \log(13)$$

- what P_X maximises $H(Y)$? We have two choices!



1. $P_X = (1/26, ..., 1/26)$
2. $P_X = (1/13, 0, ..., 1/13, 0)$ [We used before]
3. Both yield $P_Y = (1/26, 1/26, ..., 1/26)$

$$C = \max_{P_X} I(X;Y) = \log(26) - 1 = \log(13) \text{ bits/transmission}$$

7 Channel Coding Theorem

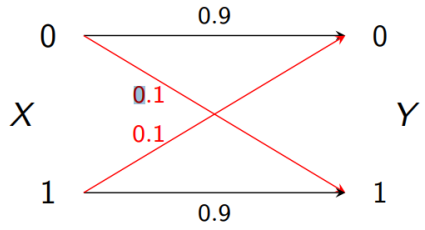
Definition of a Channel Code

- We use the channel n times to transmit k information bits.
- Each sequence of k-bits as indexing a "message" W in the set $\{1, ..., 2^k\}$
  k bits $\Leftrightarrow$ 2^k messages
- The rate R of the code is $R = \frac{k}{n}$ bits/transmission. Then the total number of messages is $2^k = 2^{nR}$.

An (n,k) channel code of rate R for the channel $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ consists of:

1. A set of messages: $\{1, ..., 2^k = 2^{nR}\}$
2. an encoding function X^n: $\{1, ..., 2^k\} \to \mathcal{X}^n$ that assigns a **codeword** to each message. The set of codewords $\{X^n(1), ..., X^n(2^{nR})\}$ is called the **codebook**
3. A decoding function g: $\mathcal{Y}^n \to \{1, ..., 2^{nR}\}$, which produces a guess of the transmitted message for each received vector

7.1 Preview of Channel Coding Theorem



For BSC(0.1):

- For input sequence X^n, the output Y^n is generated as

$$Y_i = X_i \oplus E_i \quad \text{for } i = 1, ..., n$$

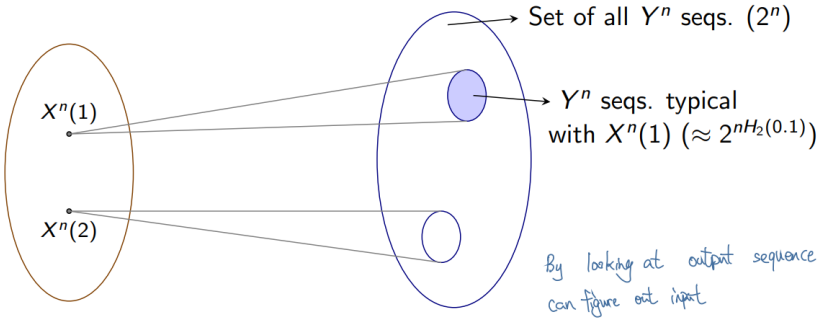
- Note:

$$X_i \oplus 0 = X_i \qquad X_i \oplus 1 = \bar{X}_i$$

- $E_1, ..., E_n$ i.i.d $\sim$ Bern(0.1) is the sequence of errors introduced by the channel
- For large n, the number of ones in $(E_1, ..., E_n) \simeq 0.1n$ (AEP)

How big is the set of Y^n sequences "typical" with any given X^n?

$$2^{nH_2(0.1)}$$



- the high-probability set of typical Y^n Sequences for a given $X^n(1)$ is much smaller than 2^n
- pick $X^n(2)$ far enough away from $X^n(1)$.
- Then the typical set of Y^n's for $X^n(2)$ is non-intersecting with the typical set for $X^n(1)$. This could again carried out for $X^n(3)$....
- Number of distinct messages we can transmit = max.number of non-intersecting sets .

$$2^{nR} \simeq \frac{2^n}{2^{nH_2(0.1)}} \Rightarrow \text{ Rate R} \simeq 1 - H_20.1) = \text{Channel Capacity}$$

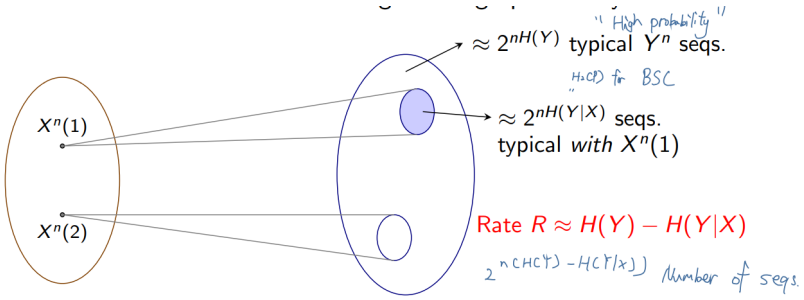
7.2 General DMC

Fix an input pmf P_X. Together with the channel $P_{Y|X}$, this gives

$$P_{XY} = P_X P_{Y|X} P_Y = \sum_X P_X P_{Y|X}$$

If $X^n(1), X^n(2), ..., X^n(2^{nR})$ are generated i.i.d $\sim P_X$, then:

- When $X_n(j)$ is transmitted, the set of highly likely Y_n's has approximately $2^{nH(Y|X)}$ sequences for each $j \in \{1, ..., 2^{nR}\}$
- These sets are non-intersecting with high probability



7.3 Joint typicality

Example: Let(X^n, Y^n) be drawn i.i.d. according to the following joint pmf:

		Y	
	P_{XY}	0	1
X	0	0.4	0.1
	1	0.1	0.4

$$Pr(X_n = x^n, Y^n = y^n) = \prod_{i-1}^{n} P_{XY}(x_i, y_i), \quad \text{for all } (x^n, y^n).$$

For large n,

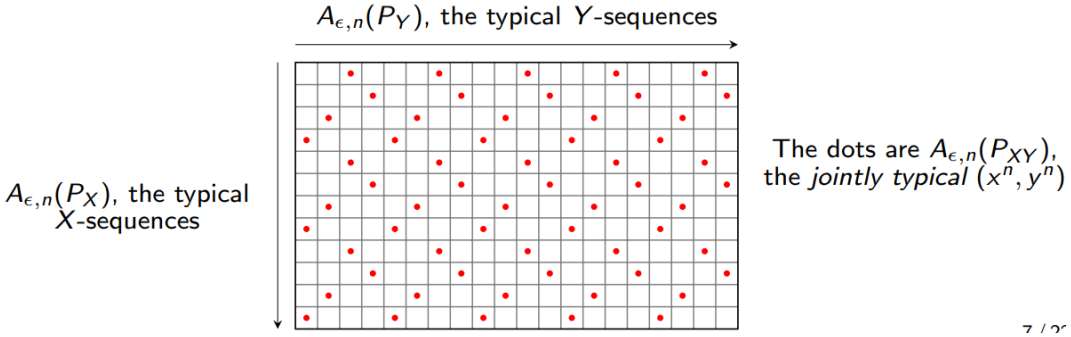
- X^n and Y^n will each have approximately 50% ones.
- The number of (X_i, Y_i) pairs that are $(0,0), (0,1), (1,0), (1,1)$ will be close to $.4n, .1n, .1n, .4n$.

7.4 Joint Typical Set

The set $A_{\epsilon,n}$ pf *jointly typical* sequences $\{(x^n, y^n)\}$ with respect to a joint pmf P_XY is defined as

$$\begin{aligned} A_{\epsilon,n} = \{&(x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n \text{ such that} \\ &| -\frac{1}{n} \log P_X(x^n) - H(X)| < \epsilon \\ &| -\frac{1}{n} \log P_Y(y^n) - H(Y)| < \epsilon \\ &| -\frac{1}{n} \log P_{XY}(x^n, y^n) - H(X,Y)| < \epsilon\} \end{aligned}$$

Where $P_{XY}(x^n, y^n) = \prod_{i=1}^{N} P_{XY}(X_i, y_i)$



7.5 Joint AEP

AEP: *Asymptotic equipartition property*
Let (X_n, Y_n) be a pair of sequences drawn i.i.d. according to P_{XY}, i.e.,

$$Pr(X^n = x^n, Y^n = y^n) = \prod_{i=1}^{n} P_{XY}(X_i, y_i), \quad \text{for all } (x^n, y^n)$$

Then for any $\epsilon > 0$:

1. $Pr((X^n, Y^n) \in A_{\epsilon,n}) \to 1$ as $n \to \infty$
2. $|A_{\epsilon,n}| \leq 2^{n(H(X,Y)+\epsilon)}$ (Size of typical set)
3. If $(\tilde{X}^n, \tilde{Y}^n)$ are a pair of sequences drawn i.i.d according to $P_X P_Y$ [i.e. $\tilde{X}^n, \tilde{Y}^n$ are independent with the same marginals as P_{XY}], then

$$Pr((\tilde{X}^n, \tilde{Y}^n) \in A_{\epsilon,n}) \leq 2^{-n(I(X;Y)-3\epsilon)}$$

7.6 Proof of AEP

7.6.1 Claim 1:

When (X_n, Y_n) drawn i.i.d. according to P_{XY},

$$Pr\left(|-\frac{1}{n}\log P_X(x^n) - H(X)| < \epsilon\right) \to 1 \text{ as } n \to \infty$$
$$Pr\left(|-\frac{1}{n}\log P_Y(y^n) - H(Y)| < \epsilon\right) \to 1 \text{ as } n \to \infty$$
$$Pr\left(|-\frac{1}{n}\log P_{XY}(x^n, y^n) - H(X,Y)| < \epsilon\right) \to 1 \text{ as } n \to \infty$$

The proof of the above is very similar to that of the AEP in Handout 2 and follows from the WLLN. Thus $Pr((X^n, Y^n) \in A_{\epsilon,n}) \to 1$ as $n \to \infty$

7.6.2 Claim 2:

We have

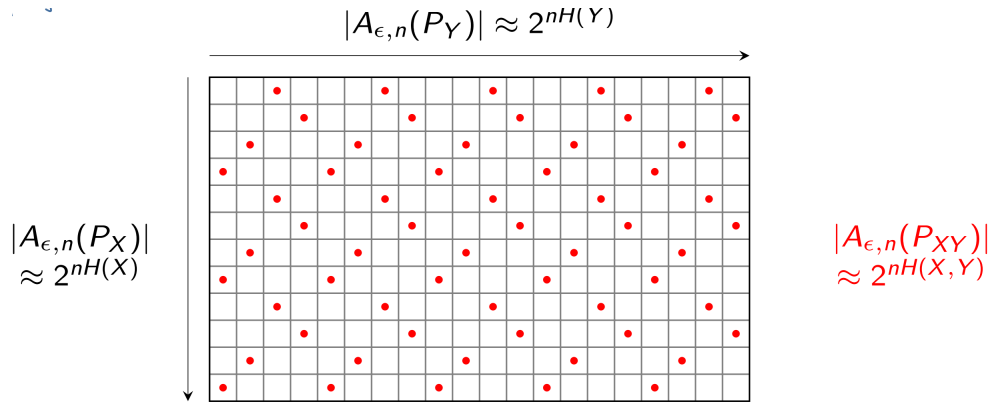
$$\begin{aligned}
1 &= \sum_{x^n, y^n} P_{XY}(x^n, y^n) \\
&\geq \sum_{(x^n,y^n)\in A_{\epsilon,n}} P_{XY}(x^n, y^n) \\
&\geq \sum_{(x^n,y^n)\in A_{\epsilon,n}} 2^{-n(H(X,Y)+\epsilon)} \\
&= 2^{-n(H(X,Y)+\epsilon)}|A_{\epsilon,n}|
\end{aligned}$$

Hence $|A_{\epsilon,n}| \leq 2^{n(H(X,Y)+\epsilon)}$.

7.6.3 Claim 3:

If $(\tilde{X}^n, \tilde{Y}^n)$ are independent but have the same marginals as X^n and Y^n, then

$$\begin{aligned}
Pr((\tilde{X}^n, \tilde{Y}^n) \in A_{\epsilon,n}) &= \sum_{(x^n,y^n)\in A_{\epsilon,n}} P_X(x^n)P_Y(y^n) \\
&\leq 2^{n(H(X,Y)+\epsilon)} \cdot 2^{-n(H(X)-\epsilon)} \cdot 2^{-n(H(Y)-\epsilon)} \\
&= 2^{-n(I(X;Y)-3\epsilon)}
\end{aligned}$$



7.7 The probability of Error of a code

The *maximal* probability of error of the code is defined as

$$\max_{j \in \{1,\ldots,2^{nR}\}} Pr(\hat{W} \neq j | W = j)$$

The *average* probability of error of the code is

$$\frac{1}{2^{nR}} \sum_{j=1}^{2^{nR}} Pr(\hat{W} \neq j | W = j)$$

W and $\hat{W}$ denote the transmitted, and decoded messages respectively.

7.8 The Channel Coding Theorem

For a DMC with capacity C, all rates less than C are achievable..

1. Fix $R < C$ and pick any $\epsilon > 0$. Then, for all sufficiently large n there exists a length-n code of rate R with maximal probability of error less than ϵ

2. Conversely, any sequence of length-n codes of rate R with average/maximal probability of error $P_e^{(n)} \to 0$ as $n \to \infty$ must have $R \leq C$.

7.8.1 Proof of the coding theorem

Prove the achievability of all rates $R < C$. (the first part).
Codebook Generation:

- Fix rate $R < C$ and input pmf P_X. We generate each of the 2^{nR} codewords independently according to the distribution

$$Pr(X_n(k) = (x_1, ..., x_n)) = \prod_{i=1}^{n} P_X(x_i) \quad \text{for } k = 1, ..., 2^{nR}.$$

- Codebook $\mathcal{B}$ can be interpreted as a $2^{nR} \times n$ matrix:

$$\mathcal{B} = \begin{bmatrix} X_1(1) & X_2(1) & \cdots & X_n(1) \\ X_1(2) & X_2(2) & \cdots & X_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ X_1(2^{nR}) & X_2(2^{nR}) & \cdots & X_n(2^{nR}) \end{bmatrix} = \begin{bmatrix} \text{Length n codeword - Message 1} \\ \text{Length n codeword - Message 2} \\ \vdots \\ \text{Length n codeword - Message } 2^{nR} \end{bmatrix}$$

- Each entry in the matrix is chosen i.i.d according to P_X. The probability that we generate a particular codebook $x^n(1), ..., x^N(2^{nR})$ is $\prod_{w=1}^{2^{nR}} \prod_{i=1}^{n} P_X(x_i(w))$

7.8.2 Encoding

1. A codebook $\mathcal{B}$ is generated as described previously. The codebook is revealed to both sender and receiver, who also know the channel transition matrix $P_{Y|X}$.
2. To transmit message W, the encoder sends $X^n(W)$ over the channel
3. The receiver receives a sequence Y^n generated according to

$$\prod_{i=1}^{n} P_{Y|X}(Y_i|X_i(W)) \tag{1}$$

4. From Y^n, the receiver has to guess which message was sent.
    - Assuming a uniform prior on the messages, the optimal decoding rule is max-likelihood decoding: decode the message $\hat{W}$ that maximises Eq. 1
    - But *joint typical* decoding is used for easier analyze.

7.8.3 Joint Typicality Decoder

The decoder declares that the message $\hat{W}$ was sent if both the following conditions are satisfied:

- $(X^n(\hat{W}), Y^n)$ is jointly typical with respect to $P_X P_{Y|X}$.
- There exists no other message $W\prime \neq \hat{W}$ such that $(X^n(\hat{W}), Y^n)$ is jointly typical.

If no such $\hat{W}$ is found or there is more than one such, an error is declared.

7.8.4 Analyze the probability of error

- The average probability of error for a given codebook $\mathcal{B}$ is

$$\frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} Pr(\hat{W} \neq w | \mathcal{B}, W = w) \tag{2}$$

- Analysing this for a specific codebook is hard. So the average of Eq. 2 over all codebooks is calculated:

$$\bar{P}_e = \frac{1}{2^{nR}} \sum_{\mathcal{B}} \sum_{w=1}^{2^{nR}} Pr(\hat{W} \neq w | \mathcal{B}, W = w) Pr(\mathcal{B}).$$

$$\bar{P}_e = \frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} \sum_{\mathcal{B}} Pr(\hat{W} \neq w | \mathcal{B}, W = w) Pr(\mathcal{B}).$$

- Recall $Pr(\mathcal{B})$ is the probability corresponding to picking each symbol of $\mathcal{B}$ i.i.d. $\sim P_X$.
- Since all the messages are equally likely, we can assume that the first message is the transmitted one. Thus

$$\bar{P}_e = \sum_{\mathcal{B}} Pr(\hat{W} \neq 1 | \mathcal{B}, W = 1) Pr(\mathcal{B}).$$

7.8.5 Error Analysis

Assuming W=1 was transmitted, there are two sources of error:

1. $X^n(1)$ is not jointly typical with the output Y^n.
2. $X^n(k)$ is jointly typical with the output Y^n for some $k \neq 1$.

- Let E_k be the event that $X_n(k)$ and Y^n are jointly typical.
- Then:

$$\begin{aligned} \bar{P}_e =& P(E_1^C \cup E_2 \cup ... \cup E_{2^nR}) \\ \leq& P(E_1^C) + P(E_2) + ... + P(E_{2^nR}) \\ & \text{(Union Bound)} \end{aligned}$$

1) Showing $P(E_1^C)$ is small:

- $X^n(1)$ i.i.d. $\sim P_X$.
- The channel generates Y^n symbols by symbol according to $P_{Y|X}(Y_i|X_i(1))$ for $i = 1, ..., n$
- Therefore $X^n(1), Y^n$ is generated i.i.d $\sim P_X P_{Y|X}$
- Joint AEP implies that $P(E_1^C) \leq \epsilon$ for sufficiently large n

2) Showing $P(E_2) + ... + P(E_{2^nR})$ is small:
For $k \neq 1$:

- $X^n(k)$ was generated independently from $X^n(1)$, and Y^n is obtained by passing $X^n(1)$ through the channel.
- Hence $X^n(k)$ and Y^n are independent for $k \neq 1$.
- Further, $X^n(k)$ is i.i.d. $\sim P_X$, and Y^n is i.i.d. $\sim P_Y$.

- From the Joint AEP, the probability that $X_n(k)$ and Y_n are jointly typical according to P_{XY} is $\leq 2^{-n(I(X;Y)-3\epsilon)}$

$$\Rightarrow P(E_2) + ... + P(E_{2^nR}) \leq (2^{nR} - 1)2^{-n(I(X;Y)-3\epsilon)}$$

Putting the two parts together:

$$\begin{aligned}\bar{P}_e &\leq P(E_1^C) + P(E_2) + ... + P(E_{2^{nR}}) \\ &\leq \epsilon + 2^{nR}2^{-n(I(X;Y)-3\epsilon)} \\ &\leq^{(a)} \epsilon + \epsilon\end{aligned}$$

(a) is true when $R < I(X;Y) - 3\epsilon$ and n is sufficiently large.

Therefore combining 1) and 2):

$$\bar{P}_e = \frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} \sum_{\mathcal{B}} Pr(\hat{W} \neq w | \mathcal{B}, W = w) Pr(\mathcal{B}) \leq 2\epsilon$$

Final Steps:

1. Choose P_X to be one that maximises $I(X;Y)$.
2. Get rid of average over codebooks: As $\bar{P}_e \leq 2\epsilon$, there exists at least one codebook $\mathcal{B}*$ with

$$P_e(\mathcal{B}^*) = \frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} Pr(\hat{W} \neq w | \mathcal{B}^*, W = w) Pr(\mathcal{B}) \leq 2\epsilon$$

3. Throw away the worst half of the codewords in $\mathcal{B}^*$: For $\mathcal{B}^*$, the probability of error averaged over all messages is $\leq 2\epsilon$. Thus the probability of error must be $\leq 4\epsilon$ for at least halt the messages.

7.8.6 The Final Code

- The number of codewords in this improved version of $\mathcal{B}^*$ is $2^{nR}/2$. Its rate is

$$\frac{\log(2^{nR}/2)}{n} = R - \frac{1}{n}.$$

- Since R is any rate less than $C - 3\epsilon$, we have shown the existence of a code with rate

$$C - 4\epsilon - \frac{1}{n}$$

whose maximal probability of error satisfies

$$\max_W Pr(\hat{W} \neq w | W = w) \leq 4\epsilon$$

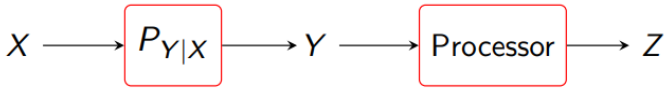
- Since $\epsilon > 0$ is an arbitrary constant, we have shown that for any $R < C$, for sufficiently large n there exists a code with arbitrarily small maximal error probability.
- This proves the first part of channel coding theorem, the converse would be proved in next chapter.

7.9 Summary

- Allow an arbitrarily small but non-zero probability of error
- Use the channel many times in succession, so that the law of large numbers comes into effect
- Random Coding: Calculate the average probability of error over a random choice of codebooks, which can then be used to show the existence of at least one good code

8 Data Processing, Fano's Inequality

8.1 Data Processing and Mutual Information



Random variables, X,Y,Z are said to form a **Markov chain** if their joint pmf can be written as

$$P_{XYZ} = P_X P_{Y|X} P_{Z|Y}$$

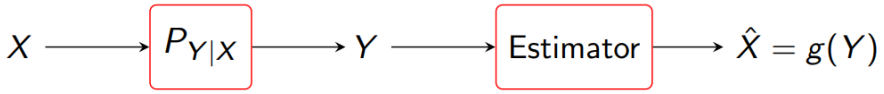
In other words, the conditional distribution of Z given (X,Y) depends only on Y, i.e., $P_{Z|XY} = P_{Z|Y}$. Example of Markov chains:

1. Y is a noisy version of X, and $Z = f(Y)$ is an estimator of X based only on Y
2. The output of the $X \to Y$ channel is fed into the $Y \to Z$ channel.

Data-Processing Inequality
If X-Y-Z form a Markov chain, then $I(X;Y) \geq I(X;Z)$.

Processing the data Y cannot increase the information about X

8.2 Fano's Inequality



- We want to estimate X by observing a correlated random variable Y
- The probability of error of an estimator $\hat{X} = g(Y)$ is $P_e = Pr(\hat{X} \neq X)$
- Given X takes values in the set $\mathcal{X}$, we wish to bound P_e

Fano's Inequality
For any estimator $\hat{X}$ such that $X - Y - \hat{X}$, the probability of error $P_e = Pr(\hat{X} \neq X)$ satisfies

$$1 + P_e \log |\mathcal{X}| \geq H(X|\hat{X}) \geq H(X|Y) \text{ or } P_e \geq \frac{H(X|Y) - 1}{\log |\mathcal{X}|}$$

No matter how good the estimator is, the probability of error can not be lower than some value

8.2.1 Proof of Fano

- Define an error random variable

$$E = \begin{cases} 1 & \text{if } \hat{X} \neq X \\ 0 & \text{if } \hat{X} = X \end{cases}$$

- Use chain rule to expand $H(E, X|\hat{X})$ in two ways:

$$\begin{aligned} H(E, X|\hat{X}) &= H(X|\hat{X}) + H(E|X, \hat{X}) \\ &= H(E|\hat{X}) + H(X|\hat{X}, E) \end{aligned} \tag{3}$$

- Claims:

  1. $H(E|X, \hat{X}) = 0$. (E is a function of ...)
  2. $H(E|\hat{X}) \leq H(E) = H_2(P_e)$. (Conditioning can only reduce entropy)
  3. $H(X|\hat{X}, E) \leq P_e \log |\mathcal{X}|$ because

$$\begin{aligned} H(X|\hat{X}, E) &= Pr(E = 0)H(X|\hat{X}, E = 0) + Pr(E = 1)H(X|\hat{X}, E = 1) \\ &\leq (1 - P_e)0 + P_e \log |\mathcal{X}| \end{aligned}$$

Use these claims in Eq.3

$$H(X|\hat{X}) = H(E|\hat{X}) + H(X|\hat{X}, E) \leq H_2(P_e) + P_e \log |\mathcal{X}|$$

Note that $H_2(P_e) \leq 1$. Therefore

$$H(X|\hat{X}) \leq 1 + P_e \log |\mathcal{X}|.$$

For the other side, the data-processing inequality given:

$$I(X; Y) = H(X) - H(X|Y) \geq I(X; \hat{X}) = H(X) - H(X|\hat{X})$$

Thus $H(X|\hat{X}) \geq H(X|Y)$.

8.2.2 Apply Fano's inequality into Channel Coding

- Consider a length n, rate R channel code, i.e., $2^n R$ codewords
- $\hat{W}$ is a guess of W based on Y^n
- W uniformly distributed in $\{1, 2, ..., 3^n R\}$
- $P_e = Pr(\hat{W} \neq W) = \frac{1}{2^{nR}} \sum_{k=1}^{2^{nR}} Pr(\hat{W} \neq k|W = k)$
- Apply Fano's Inequality:

$$H(W|\hat{W}) \leq 1 + P_e \log |W| \leq 1 + P_e \log 2^{nR} = 1 + P_e nR$$

Above will be used to show that any sequence of $(2^{nR}, n)$ codes with $P_e \to 0$ must have $R \leq C$.

8.2.3 Lemma/Helping theorem for channel coding converse

Let Y^n be the result of passing a sequence X^n through a DMC of channel capacity C. Then

$$I(X^n;Y^n) \leq nC$$

Regardless of the distribution of X^n. Proof:

$$\begin{aligned}
I(X^n;Y^n) &= H(Y^n) - H(Y^n|X^n) \\
&= H(Y^n) - \sum_{i=1}^{n} H(Y_i|Y_{i-1},...,Y_1,X^n) \\
&=^{(a)} H(Y^n) - \sum_{i=1}^{n} H(Y_i|X_i) \\
&\leq^{(b)} \sum_{i=1}^{n} H(Y_i) - \sum_{i=1}^{n} H(Y_i|X_i) \\
&= \sum_{i=1}^{n} I(X_i;Y_i) \\
&\leq^{(c)} nC.
\end{aligned}$$

Justification for step (a)-(c):

(a) The channel is memoryless. This meas that given $X-i, Y_i$ is conditionally independent of everything else.

(b)

$$\begin{aligned}
H(Y^n) &= H(Y_1) + H(Y_2|Y_1) + ... + H(Y_n|Y_{n-1},...,Y_1) \\
&\leq H(Y_1) + H(Y_2) + ... + H(Y_n)
\end{aligned}$$

as conditioning can only reduce entropy

(c) From the definition of capacity, C is the maximum of $I(X;Y)$ over all joint pmfs over (X,Y) where $P_{Y|X}$ is fixed by the channel.

8.3 Prove of Channel Coding Converse

Consider any length n, rate R channel code, i.e., $(2^{nR}, n)$ channel code with average probability of error P_e. We have:

$$\begin{aligned}
nR &=^{(a)} H(W) \\
&=^{(b)} H(W|\hat{W}) + I(W:\hat{W}) \\
&\leq^{(c)} 1 + P_e nR + I(W;\hat{W}) \\
&\leq^{(d)} 1 + P_e nR + I(X^n;Y^n) \\
&\leq^{(e)} 1 + P_e nR + nC.
\end{aligned}$$

This implies:

$$P_e \geq 1 - \frac{C}{R} - \frac{1}{nR}$$

Thus, unless $R \leq C$, P_e is bounded away from 0 as $n \to \infty$

$$n \to \infty \quad P_e \geq 1 - \frac{c}{R}$$

If $R \geq C$, error would be large.
Justification for steps (a)-(e):

(a) W is uniform over $\{1, ..., 2^{nR}\}$

(b) $I(W; \hat{W}) = H(W) - H(W|\hat{W})$

(c) Fano applied to $H(W|\hat{W})$ See Chapter 6

(d) Data processing inequality applied to $W - X^n - Y^n - \hat{W}$

(e) From previous Lemma

8.4 Summary

C is a sharp threshold:

- For all rates $R < C$, there exists a sequence of $(2^{nR}, n)$ codes whose $P_e \to 0$
- For rates $R > C$, no sequence of $(2^{nR}, n)$ codes whose $P_e \to 0$

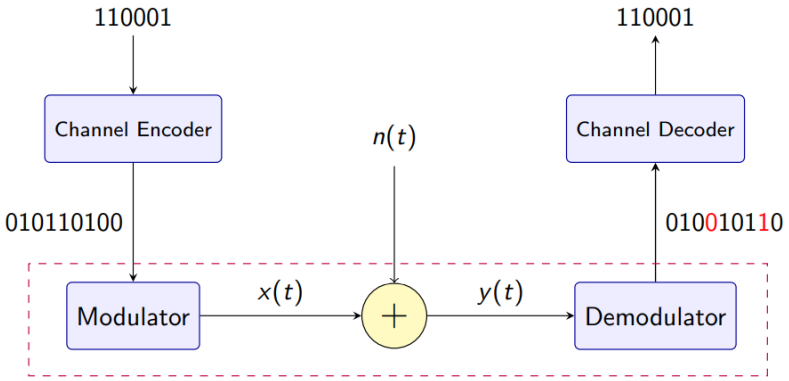
However, channel coding theorem doe not give practical technique to communicate reliably at any rate $R < C$

1. Joint typical decoding is too complex to be feasible
2. An $2^n R \times n$ codebook is too large to store

In the next chapters, compact codebook representation and fast encoding/decoding algorithms are introduced.

9 Information theory with continuous RVs

9.1 The Additive White Gaussian Noise(AWGN) Channel



$$Y(t) = X(t) + N(t)$$

Assumptions on the channel:

1. Input $X(t)$ is *power-limited* to P:

$$\Rightarrow \text{Average energy over time T must be } \leq P \text{ for large T}$$

   - Battery power is limited; may also need to limit interference with other users

2. $X(t)$ is *band-limited* to W:

$$\Rightarrow \text{Fourier Transform of } X(t) \text{ must be zero outside } [-W, W]$$

   - B.W. is a scarce resource

3. Noise N(t) is a random process assumed to be white Gaussian

   - Good approximation to sum of many independent zero-mean random variables(Central Limit Theorem) (Know mean and variance but not exact noise)
   - Gaussian noise is in a sense the 'worst' among noise rvs of a given variance
   - Easy to analyse

By Shannon-Nyquist sampling theorem, it can be shown that the *continuous-time* channel

$$Y(t) = X(t) + N(t)$$

with power constraint P and bandwidth constraint W is equivalent to the following *discrete-time* channel
Discrete-time AWGN Channel:

$$Y_k = X_k + Z_k, \quad k = 1, 2, ...,$$

- Average power constraint P on input $\Rightarrow$

$$\frac{1}{n}\sum_{k=1}^{n} X_k^2 \leq P$$

- Z_k are i.i.d. Gaussian with mean 0, variance σ^2 (denoted by $\mathcal{N}(0, \sigma^2)$)
- Need to compute the capacity of this channel in bits/transmission

9.2 Information Theory for Continuous Alphabets

- In the AWGN channel, X_k, Z_k, Y_k are all real-valued random variables
- Define entropy for rvs in a continuous space, **Differential Entropy**

$$h(X) = \int_{-\infty}^{\infty} f_X(u) \log \frac{1}{f_X(u)}\, du$$

9.2.1 Differential Entropy

Example: Let X be uniformly distributed in the interval $[0, a]$. Its differential entropy is:

$$h(X) = \int_0^a f_X(u) \log \frac{1}{f_X(u)}\, du = \int_0^a \frac{1}{a} \log a\, du = \log a$$

- For $a = 1$, $h(X) = 0$
- For $0 < a < 1$, $h(X)$ is negative!
- For any random variable X, we may interpret $h(X)$ as uncertainty relative to that of a $Unif[0, 1]$ random variable. $Unif[0, 1]$ rv is a baseline which has *0 differential entropy.*

Example: $h(X)$ for a Gaussian rv $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$\phi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$\begin{aligned}
h(X) =& -\int \phi(x) \frac{\ln \phi(x)}{\ln 2}\, dx \\
=& -\frac{1}{\ln 2} \int \phi(x) \left[-\frac{(x-\mu)^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \\
=& \frac{1}{\ln 2} \left[\frac{Var(X)}{2\sigma^2} + \frac{1}{2} \ln 2\pi\sigma^2 \right] \\
=& \frac{1}{\ln 2} \left[\frac{1}{2} + \frac{1}{2} \ln 2\pi\sigma^2 \right] \\
=& \frac{1}{2} \log 2\pi e\sigma^2
\end{aligned}$$

9.2.2 Differential Entropy of Multiple RVs

The joint differential entropy of (X, Y) with joint density f_{XY} is

$$h(X, Y) = \int f_{XY}(u, v) \log \frac{1}{f_{XY}(u, v)}\, du\, dv$$

The conditional differential entropy of X given Y is

$$h(X|Y) = \int f_{XY}(u, v) \log \frac{1}{f_{X|Y}(u|v)}\, du\, dv$$

Joint differential entropy chain rule:

$$h(X, Y) = h(X) + h(Y|X)$$

Mutual information is similar to the discrete case:

$$I(X; Y) = h(X) - h(X|Y) = h(X) + h(Y) - h(X, Y) = h(Y) - h(Y|X)$$

Relative entropy between two densities f and g is

$$D(f||g) = \int f(u) \log \frac{f(u)}{g(u)}\, du$$

- $D(f||g) \geq 0$ still holds.
- Since $I(X;Y) = D(f_{XY}||f_X f_Y)$, mutual information is always non-negative
- Chain rules for differential entropy and mutual information hold, and are analogous to the discrete case.

To summarize, the main difference is that differential entropy can be negative and should be interpreted as uncertainty relative to a baseline, but other quantities and properties should have the usual interpretations.

9.3 Capacity of the discrete-time AWGN Channel

$$Y_k = X_k + Z_k, \quad k = 1, 2, ... \tag{4}$$

Input power constraint is

$$\frac{1}{n}\sum_{k=1}^{n} X_k^2 \leq P$$

Z_k i.i.d. Gaussian $\sim \mathcal{N}(0, \sigma^2)$

- The capacity of the AWGN channel with power constraint P is $C = \max I(X;Y)$, where the maximum is over all inputs pdfs f_X for which $\mathbb{E}X^2 \leq P$
- Compute C explicitly first
- We'll then show C is the supremum of achievable rates with arbitrarily low probability of error for the channel in Eq. 4

9.3.1 Computing the AWGN Capacity

$$\begin{aligned} I(X;Y) =& h(Y) - h(Y|X) \\ =& h(Y) - h(X+Z|X) \\ =& h(Y) - h(Z|X) \\ =& h(Y) - h(Z) \end{aligned}$$

Last Equality since Z is independent of X

- $Z \sim \mathcal{N}(0, \sigma^2)$.Hence $h(Z) = \frac{1}{2}\log 2\pi e\sigma^2$
- Note,
$$\mathbb{E}Y^2 = \mathbb{E}(X+Z)^2 = \mathbb{E}X^2 + 2\mathbb{E}(XZ) + \mathbb{E}Z^2 = \mathbb{E}X^2 + \sigma^2$$
(X,Z are independent $\Rightarrow \mathbb{E}(XZ) = \mathbb{E}X\mathbb{E}Z = 0$)
- Therefore $\mathbb{E}X^2 \leq P$ implies $\mathbb{E}Y^2 \leq P + \sigma^2$ (With equality when $\mathbb{E}X^2 = P$)

KEY FACT: Among all random variables Y with $\mathbb{E}Y^2 \leq (P + \sigma^2)$, the maximum differential entropy is achieved when Y is Gaussian $\mathcal{N}(0, P + \sigma^2)$

Since $Y = X + Z$ and $\mathbb{E}Y^2 \leq (P + \sigma^2)$, the key fact implies:

$$h(Y) \leq \frac{1}{2}\log 2\pi e(P + \sigma^2)$$

with equality if X is distributed as $\mathcal{N}(0, P)$. For this X:

$$\begin{aligned} I(X;Y) &= h(Y) - \frac{1}{2}\log 2\pi e\sigma^2 \\ &= \frac{1}{2}\log 2\pi e(P+\sigma^2) - \frac{1}{2}\log 2\pi e\sigma^2 \\ &= \frac{1}{2}\log\left(1+\frac{P}{\sigma^2}\right) \end{aligned}$$

Therefore the capacity of the discrete-time AWGN channel with input constraint P and noise variance σ^2 is

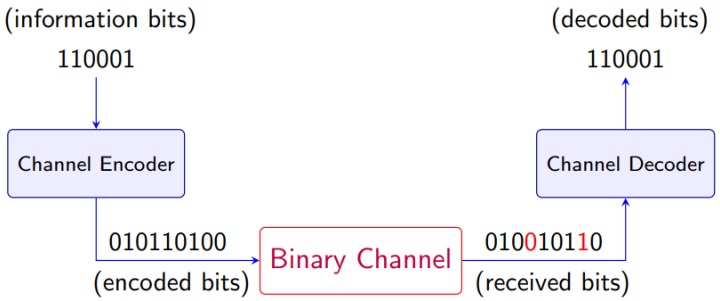
$$C = \frac{1}{2}\log\left(1+\frac{P}{\sigma^2}\right) \quad \text{bits/transmission}$$

- C depends only on the *signal-to-noise ratio* (snr) $\frac{P}{\sigma^2}$
- If the channel has(baseband) bandwidth W, it can be used $2W$ times per second. Therefore, the capacity in bits/s is

$$2W \cdot \frac{1}{2}\log\left(1+\frac{P}{\sigma^2}\right) = W\log\left(1+\frac{P}{\sigma^2}\right)$$

In the rest of the course, fixed modulation scheme is assumed and focus on designing good binary codes for the binary erasure channel and the binary symmetric channel.

10 Binary Linear Block Codes



- **Channel Encoder**: Adds redundancy to the source bots in a *controlled* manner
- **Channel Decoder**: Recovers the source bits from the received bits by exploiting the redundancy

10.1 Block Code

An (n, k) binary block code maps every block of k data bits into a length n binary codeword

- Add redundancy $\Rightarrow n > k$
- The **rate** R of the code is $R = \frac{k}{n}$
- Assuming the codewords are distinct, the size of the code, i.e., the number of codewords, is $M = 2^k$

Examples:

1) $(n, 1)$ repetition code:

$$0 \longrightarrow 00\cdots0 \qquad 1 \longrightarrow 11\cdots1$$

2) An$(n = 5, k = 2)$ block code:

$$\begin{aligned} 00 &\longrightarrow 10101 \\ 01 &\longrightarrow 10010 \\ 10 &\longrightarrow 01110 \\ 11 &\longrightarrow 11111 \end{aligned}$$

3) Single parity check $(K + 1, k)$ code:

   Given k data bits, to form the codeword, add a one $(K + 1)$th parity bit which is the binary (modulo-two) sum of the k data bits.

   E.g. with $k = 4$, this gives a (5,4) code with 16 codewords:

$$\begin{aligned} 0000 &\longrightarrow 00000 \\ 0001 &\longrightarrow 00011 \\ 0010 &\longrightarrow 00101 \\ 0011 &\longrightarrow 00110 \end{aligned}$$

4) To prove the achievability in the channel coding theorem, a $(n, k = nR)$ block code by generating the codeword entries i.i.d. $\sim P_X$ is generated.

   This gives a rate-optimal code with very low probability of decoding-error, but computationally infeasible to implement.

What makes a a good code?

- rate $R = k/n$ as high as possible
- Low probability of decoding error
- Computationally efficient ways to encode ans decode.

compare **Repetition** code vs **Single-parity**code on a BSC:

1) The $(n, 1)$ repetition code with n odd:
   - Decode: If the received word contains a majority of zeros, declare data bit to be 0; else declare 1.
   - Can reliably correct up to $\lfloor \frac{n}{2} \rfloor$ errors in the received word.
   - But rate is only 1/n
2) The $(K + 1, k)$ Single-parity code
   - Has rate $k/(k + 1)$, which $\to 1$ as $k \to$ big
   - But can only detect one error and cannot correct it
   - E.g. with a $(5, 4)$ single-parity code, suppose the data bits were $(0, 0, 0, 1)$, received bits are $(0, 1, 0, 1, 1)$. There has been an error, but cannot correct the error.

10.2 Hamming distance of a code

- The *Hamming distance* d($\underline{x}, \underline{y}$) between two binary sequences $\underline{x}$,$\underline{y}$ of length n is the number of positions in which $\underline{x}$ and $\underline{y}$ differs.
- Let $\mathcal{B}$ be a code with codewords $\{\underline{c}_1, ..., \underline{c}_M\}$. Then the minimum distance d_{min} is the smallest Hamming distance between any pair of codewords. That is,

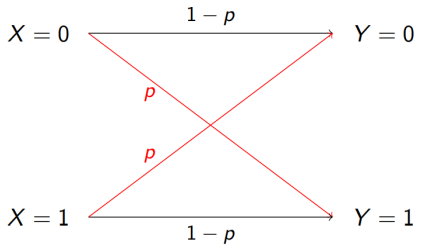
$$d_{min} = \min_{i \neq j} d(\underline{c}_i, \underline{c}_j)$$

10.3 Optimal decoding of a block code

- Given a block code $\mathcal{B}$ with codewords $\{\underline{c}_1, ..., \underline{c}_M\}$, the optimal decoder is the one that minimises the probability of the decoding error.
- For any channel described by $P_{Y|X}$, if the input bits/messages corresponding to the codewords are equally likely, then the optimal decoder given the received length-n sequence $\underline{y}$ is the max.-likelihood decoder given by:

$$Decode\, \hat{\underline{c}} = \arg \max_{\underline{c} \in \{\underline{c}_1, ..., \underline{M}\}} Pr(\underline{y}|\underline{c})$$

10.3.1 Optimal decoding on the BSC(p)



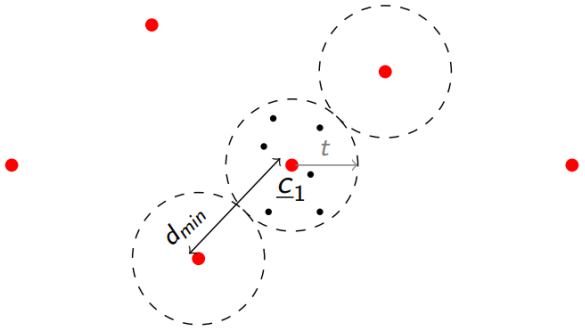
Given the codeword $\underline{c}$ was transmitted and $\underline{y}$ was received, the number of channel errors is $d(\underline{y}, \underline{c})$ and the number of correctly received bits is $n - d(\underline{y}, \underline{c})$. Hence

$$Pr(\underline{y}|\underline{c}) = p^{d(\underline{y},\underline{c})}(1-p)^{n-d(\underline{y},\underline{c})} = (1-p)^n \left(\frac{p}{1-p}\right)^{d(\underline{y},\underline{c})}$$

Thus for $p < 0.5$, the optimal decoding rule is:

$$Decode\, \hat{\underline{c}} = \arg \min_{\underline{c} \in \{\underline{c}_1, ..., \underline{M}\}} d(\underline{y}, \underline{c})$$

i.e. the optimal decoder for BSC picks the codeword closest in Hamming distance to $\underline{y}$.



The big red dots represent the codewords, the little ones are the other length n binary sequences.(Potentially be received sequences)

Draw spheres of radius t centred on each codeword. Sphere centred at $\underline{c}_i$ contains all the binary sequence $\underline{r}$ such that $d(\underline{c}_i, \underline{r}) \leq t$

- If fewer than t errors occur, the received word $\underline{r}$ will lie within the sphere of the transmitted codeword.
- Further, the spheres won't overlap if $2t < d_{min}$, or $2t \leq d_{min} - 1$. Therefore:

We can successfully correct any pattern of t errors if $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor$

10.4 Linear Block Codes

10.4.1 Definition

A (n, k) linear block code (LBC) is defined in terms of k length-n binary vectors, say $\underline{g}_1, ..., \underline{g}_k$. A Sequence of k data bits, say, $\underline{x} = (x_1, ..., x_k)$ is mapped to a length-n codeword $\underline{c}$ as follows.

$$\underline{c} = x_1\underline{g}_1 + ... + x_k\underline{g}_k$$

This can be compactly written as

$$\underline{c} = \underline{x}G, \quad \text{where } G = \begin{bmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \vdots \\ \underline{g}_k \end{bmatrix}$$

- The $k \times n$ matrix G is called a **Generator matrix** of the code
- k is code dimension, n is the block dimension
- Matrix map $1 \times k$ row vector $\underline{x}$ to $1 \times n$ row vector $\underline{c}$ (We would have 2^k of $\mathcal{X}$)
- This matrix multiplication is over the binary field:

$$x + x = 0, \quad x + \bar{x} = 1, \quad x \cdot 0 = 0, \quad x \cdot 1 = x$$

10.4.2 Example of LBC

Find the dimension and code corresponding to each of the following generator matrices.

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Each of the matrices defines a $(4, 2)$ LBC. Hence the code dimension $k = 2$.

G_1

$\underline{x}$		$\underline{c}$
0 0	$\longrightarrow$	0 0 0 0
0 1	$\longrightarrow$	0 1 1 1
1 0	$\longrightarrow$	1 0 0 1
1 1	$\longrightarrow$	1 1 1 0

G_2

$\underline{x}$		$\underline{c}$
0 0	$\longrightarrow$	0 0 0 0
0 1	$\longrightarrow$	0 1 1 1
1 0	$\longrightarrow$	1 1 1 0
1 1	$\longrightarrow$	1 0 0 1

G_1, G_2 yield same set of codewords C but different mappings $\Rightarrow$

The generator matrix for a code is not unique

Systematic generator matrices for a code is preferred.

$$G = [I_k \quad | \quad P]$$

where I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix.

10.4.3 Systematic Generator Matrices

For a systematic $G = [I_k \mid P]$, the codeword corresponding to length-k data vector $\underline{x}$ is

$$\underline{c} = \underline{x}G = \underline{x} \cdot [I_k \mid P] = [\underline{x} \mid \underline{x}P]$$

The length-n codeword consists of the k data bits $\underline{x}$, followed by $(n-k)$ parity bits $\underline{x}P$.

10.4.4 Converting to Systematic Form

Given any generator matrix, systematic version can be found:

1. Use elementary row operations: swap rows and/or replace any row by a sum of that row and any other rows.

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{R_1 = R_1 + R_2} G_{sys} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

   If G_{sys} is obtained from G_2 via elementary row operations, then they have the same set of codewords; only the mappings are different.

2. To bring G_1 into systematic form, may also need to swap columns. This leads to rearranging the components of the codewords.

10.5 LBCs as Subspaces

- Let C be an (n, k) LBC with codewords $\{c_0, ..., c_{M-1}\}$
- C is a subspace of $\{0, 1\}^n$, i.e., it is closed under vector addition and scalar multiplication.
- Proof: We need to show:

  1) For any codewords $\underline{c}_i, \underline{c}_j \in C$, $\underline{c}_i + \underline{c}_j \in C$
     - Let $\underline{c}_i = a_1\underline{g}_1 + \cdots + a_k\underline{g}_k$, and $\underline{c}_j = b_1\underline{g}_1 + \cdots + b_k\underline{g}_k$. Then

$$\underline{c}_i + \underline{c}_j = (a_1 + b_1)\underline{g}_1 + \cdots + (a_k + b_k)\underline{g}_k$$

       Thus $\underline{c}_i + \underline{c}_j$ is the codeword corresponding to the data sequence $\underline{x} = (a_1 + b_1, ..., a_k + b_k)$, and belong to C.
     - For $\underline{c} \in C$ and $\underline{c} + \underline{c} = \underline{0}$. This implies all-zero codeword $\underline{0}$ always belongs to C.
  2) If $\underline{c} \in C$, then $0 \cdot \underline{c}$ and $1 \cdot \underline{c}$ are in C.
     - Proved by $0 \cdot \underline{c} = \underline{0}$ and $1 \cdot \underline{c} = \underline{c}$

- Basis, dimension, orthogonality that are in $\mathbb{R}^n$ can be carried over to $\{0,1\}^n$.

$$\text{Vectors } \underline{u}, \underline{v} \in \{0,1\}^n \text{ are said to be orthogonal of } \underline{u}\underline{v}^T = 0$$

- However, a vector in $\{0,1\}^n$ can be orthogonal to itself, different from $\mathbb{R}^n$
- It helps to think of C being a k-dimensional subspace of $\{0,1\}^n$ as similar to the plane $x+y+z=0$ being a two-dimensional subspace of $\mathbb{R}^3$

Summary of properties of a (n,k) linear block code:

1. The code is a k-dimensional subspace of vectors from $\{0,1\}^n$
2. The rows of G,$\{\underline{g}_1, ..., \underline{g}_k\}$ form a basis for C. Each codeword is a linear combination of basis vectors: $x_1\underline{g}_1 + ... + x_k\underline{g}_k$.
3. The sum of any two codewords is also a codeword; the all-zero vector $\underline{0}$ is always a codeword.

10.6 The Parity Check Matrix

10.6.1 Definition

- The orthogonal complement of C, denoted $C^\perp$ is defined as the set of all vectors in $\{0,1\}^n$ that are orthogonal to each vector in C.
- $C^\perp$ is a subspace: if $\underline{u}, \underline{v} \in C^\perp$, then for any $\underline{c} \in C$, $\underline{c}\underline{u}^T = 0$ and $\underline{c}\underline{v}^T = 0 \Rightarrow \underline{c}(\underline{u}+\underline{u})^T = 0 \Rightarrow (\underline{u}+\underline{u}) \in C^\perp$
- Since C has dimension k, can be shown that $C^\perp$ has dimension $(n-k)$.
- Thus we can find a basis $\{\underline{h}_1, ..., \underline{h}_{n-k}\}$ for $C^\perp$, expressed as

$$H = \begin{bmatrix} \underline{h}_1 \\ \underline{h}_2 \\ \vdots \\ \underline{h}_{n-k} \end{bmatrix}$$

- The $(n-k) \times n$ matrix H is called the **parity check matrix**
- Each codeword $\underline{c} \in C$ is orthogonal to each row of H, i.e.,

$$\underline{c}H^T = \underline{0} \quad \Longrightarrow \quad \underline{x}GH^T = \underline{0} \text{ for all } \underline{x} \quad \Longrightarrow \quad GH^T = 0$$

- Any H gives $GH^T = 0$ is a valid parity check Matrix

10.6.2 Finding H

- If we have systematic $G = [I_k \mid P]$, then take

$$H = \left[P^T \mid I_{n-k}\right]$$

- P is $k \times (n-k)$. P^T is $(n-k) \times k$, and

$$GH^T = [I_k \mid P] \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} = P + P = 0$$

- Example: The $(7, 4)$ Hamming code has parity check matrix

$$H = \begin{bmatrix} 1\,1\,0\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0 \\ 1\,0\,1\,1\,0\,0\,1 \end{bmatrix}$$

  From this, we can obtain the generator matrix:

$$G = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,0\,1 \end{bmatrix}$$

The codewords satisfy $\underline{c}H^T = \underline{0}$. For the Hamming code, $\underline{c} = (c_1, ..., c_7)$, and a codeword bits must satisfy:

$$\begin{aligned} c_1 + c_2 + c_4 + c_5 &= 0 \\ C_1 + c_2 + c_3 + c_6 &= 0 \\ c_1 + c_3 + c_4 + c_7 &= 0 \end{aligned}$$

These equations obtained from $\underline{c}H^T = 0$, are called the *parity-check equations* of the code.

- The parity matrix H is a complete description of the linear code, just like the generator matrix G. Given H you can find G. Given G you can find H.
- THe codeowrds are precisely the binary sequences $\underline{c} \in \{0, 1\}^n$ that satisfy

$$\underline{c}H^T = 0$$

  i.e., the binary dot-product of a code-word with each row of H is 0.

10.7 Minimum Distance of an LBC

- The Hamming distance between two binary vectors $\underline{u}, \underline{v}$ can be expressed as

$$d(\underline{u}, \underline{v}) = wt(\underline{u} + \underline{v}),$$

  where wt refers the Hamming *weight*(i.e., the number of ones) of the vector.

- Since the minimum distance of a code is

$$d_{min} = \min_{i \neq j} d(\underline{c}_i, \underline{c}_j) = \min_{i \neq j} wt(\underline{c}_i + \underline{c}_j)$$

- For any two codewords $\underline{c}_i, \underline{c}_j$ of an LBC, note that $\underline{c}_i + \underline{c}_j$ is also a codeword, say, $\underline{c}_k$. Therefore

$$d_{min} = \min_{i \neq j} d(\underline{c}_i, \underline{c}_j) = \min_{\underline{c}_k \neq 0} wt(\underline{c}_k)$$

The minimum distance of an LBC equals the Hamming weight among the non-zero codewords.

Another useful property:

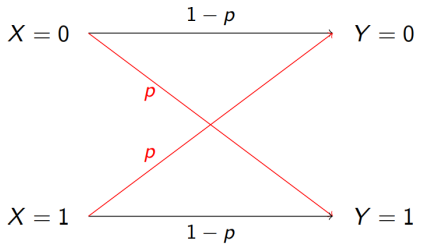
Let C be an linear block code with parity check matrix H.The minimum distance of C is the smallest number of columns of H that sum to $\underline{0}$.

Example: Determine the dimension, d_{min}, and the guaranteed error correcting capability of an LBC with

$$H = \begin{bmatrix} 1\,0\,1\,1\,0\,0 \\ 1\,1\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,1 \end{bmatrix}$$

- Since $H : (n-k) \times n$ and $n - k = 3, n = 6$. Hence the dimension $k = 3$, and there are $2^3 = 8$ codewords.
- For d_{min}, observe that no two columns sum up to $\underline{0}$(only if there is identical columns) so try 3: $col_1 + col_4 + col_5 = \underline{0} \Rightarrow d_{min} = 3$.
- For BSC: can correct any pattern of up to t errors if $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor \Rightarrow$ the above code can correct all patterns of 1 error.

10.8 Performance of Block Codes over a BSC(p)



Since all error patterns of weight up to $\lfloor \frac{d_{min}-1}{2} \rfloor$ will be corrected. Hence prob. of correctly decoding the codeword is at least the probability that the channel flips $\lfloor \frac{d_{min}-1}{2} \rfloor$ or fewer bits.

$$\Rightarrow P_{\text{correct}} \geq \sum_{w=0}^{\lfloor (d_{\min}-1)/2 \rfloor} \binom{n}{w} p^W (1-p)^{n-w}$$

$$\Rightarrow P_{\text{error}} \leq 1 - \sum_{w=0}^{\lfloor (d_{\min}-1)/2 \rfloor} \binom{n}{w} p^W (1-p)^{n-w}$$

10.9 A caveat about minimums-distance

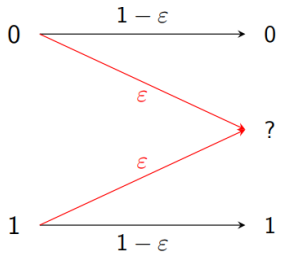
- The above bound on P_e may be pessimistic because $d_{\min}$ only gives the guaranteed error correction capability of the code.
- There may still be many error patterns of weight $> \lfloor \frac{d_{\min}-1}{2} \rfloor$ that can be corrected : you only cannot guarantee that every error pattern of weight t will be corrected if $t > \lfloor \frac{d_{\min}-1}{2} \rfloor$.

The discussion of pros and cons of minimum-distance as a design metric for codes is subtle which is not examinable. The important fact is that:

$d_{\min}$ gives you the guaranteed error correction capability of a code, but the code may be able to correct many more error patterns.

11 Low Density Parity Check(LDPC) Codes for the Binary Erasure Channel

11.1 The Binary Erasure Channel (BEC)



- Capacity of BEC is $(1 - \epsilon)$
- We want to construct linear block codes with rate close to $(1-\epsilon)$ that have fast decoding algorithms and low probability of decoding error.
- Besides being practically important (erasures models packet losses in internet), designing good codes for the BEC gives insight into designing codes for general binary input channels.

11.2 Linear Codes for the BEC

Consider the code defined by the foll. 10×16 parity check matrix. Since $n = 16$ and $n - k = 10$, $k = 6$ and the rate of the code is 3/8.

$$H = \begin{bmatrix} 1&1&1&0&0&0&0&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&1&0&0&0&0 \\ 0&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0 \\ 1&0&1&1&1&0&0&0&1&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0 \\ 1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0 \\ 1&1&0&1&0&0&1&0&1&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1 \end{bmatrix}$$

$$\text{Received } \underline{y} = \begin{bmatrix} x & x & x & x & x & x & x & x & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

To recover the eight erased bits (denoted by a,b,...,h): a codeword $\underline{C}$ has to satisfy $\underline{C}H^T = 0$
$\Rightarrow$ Set binary dot-product of $\underline{y}$ with each row of H to 0

Contribution to dot products from known bits of $\underline{y}$ can be computed and moved to RHS. These columns are greyed out:

$$\begin{bmatrix} 1&1&1&0&0&0&0&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&1&0&0&0&0 \\ 0&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0 \\ 1&0&1&1&1&0&0&0&1&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0 \\ 1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0 \\ 1&1&0&1&0&0&1&0&1&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Received } \underline{y} = \begin{bmatrix} a & b & c & d & e & f & g & h & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

We need to solve 8 binary variables from 10 linear equations.

- Possible if there are at least 8 independent rows in the effective $10 \times 8 matrix$ (Entries in dark)
- Can solve the system of equations by performing elementary row operations to get the effective matrix in lower/upper triangular form (By Gaussian elimination)

Gaussian elimination yields:

$$\left[\begin{array}{cccccccc} 1&1&1&0&0&0&0&0\\ 0&1&1&1&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ \hline 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{array}\right] = \left[\begin{array}{c} 0\\0\\0\\1\\0\\1\\1\\0\\ \hline 0\\0 \end{array}\right]$$

$$[a\ b\ c\ d\ e\ f\ g\ h]$$

Back-substitution yields:

$$[a\ b\ c\ d\ e\ f\ g\ h] = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 0]$$

Information bits can be recovered directly if the code is systematic (this one isn't)

- The above matrix-inversion/Gaussian-elimination decoder is optimal for the BEC.
- If the pattern of erasures is such that you cannot solve the resulting system of equations, you cannot solve it using any other decoder either.
- Complexity of the above decoder scales like n^3 with block length (due to Gaussian elimination) - not too bad, but can we do better?

Next, faster iterative decoder that does opportunistic back-substitution is introduced.

11.3 Iterative Decoding

Look for a row which has exactly one 1 among the erased locations.

Let's consider a different code defined by the following 10×16 parity check matrix. $\underline{y}$f has eight erased locations as before:

$$H = \left[\begin{array}{cccccccccccccccc} 0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1\\ 0&0&0&\mathbf{1}&0&1&1&0&0&1&0&0&0&0&0&0\\ 0&0&1&0&1&0&0&0&1&0&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\ 1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\ 0&1&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\ 0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0\\ 0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&1&0&0&1&1&0&0\\ 1&0&0&1&0&0&0&1&0&0&1&0&0&0&0&0 \end{array}\right]$$

$$\text{Received } \underline{y} = \left[\begin{array}{cccccccccccccccc} 1&a&b&c&d&1&0&0&0&1&e&1&f&g&h&1 \end{array}\right]$$

Setting binary dot-product of $\underline{y}$ with second row to 0 gives $c = 0$

$$H = \begin{bmatrix} 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,\mathbf{1}\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

$$\underline{y} = \begin{bmatrix} 1\ a\ b\ \mathbf{0}\ d\ 1\ 0\ 0\ 0\ 1\ e\ 1\ f\ g\ h\ 1 \end{bmatrix}$$

Setting binary dot-product of $\underline{y}$ with fourth row to 0 gives $g = 1$

$$H = \begin{bmatrix} 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,\mathbf{1}\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

$$\underline{y} = \begin{bmatrix} 1\ a\ b\ 1\ d\ 1\ 0\ 0\ 0\ 1\ e\ 1\ f\ 1\ h\ 1 \end{bmatrix}$$

Setting binary dot-product of $\underline{y}$ with fifth row to 0 gives $e = 1$

$$H = \begin{bmatrix} 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,\mathbf{1}\,0 \\ 0\,0\,\mathbf{1}\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

$$\underline{y} = \begin{bmatrix} 1\ a\ b\ 0\ d\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ f\ 1\ h\ 1 \end{bmatrix}$$

Setting binary dot-product of $\underline{y}$ with seventh row to 0 gives $h = 0$ Setting binary dot-product of $\underline{y}$ with eighth row to 0 gives $b = 1$

$$H = \begin{bmatrix} 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,\mathbf{1}\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,\mathbf{1}\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

$$\underline{y} = \begin{bmatrix} 1\ a\ 1\ 0\ d\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ f\ 1\ 0\ 1 \end{bmatrix}$$

Third and sixth rows then give $d = 0$, $a = 1$, respectively

$$H = \begin{bmatrix} 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,\mathbf{1}\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,\mathbf{1}\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,\mathbf{1}\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

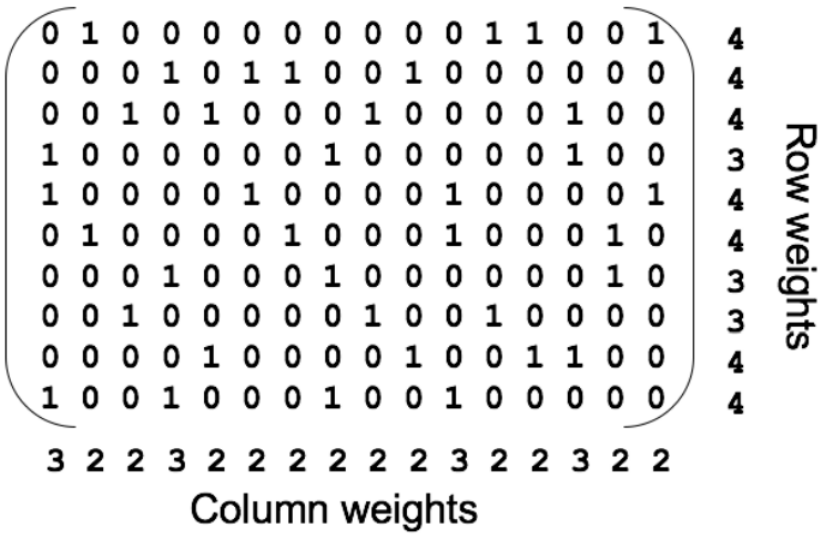
$$\underline{y} = \begin{bmatrix} 1\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,f\,1\,0\,1 \end{bmatrix}$$

Finally, we decide $f = 1$

- Here we were lucky. However, iterative decoding won't always work.
- what property must parity-check matrix have for iterative decoding to succeed with high probability?
  - When the effective matrix can be triangulated by just row swaps, i.e., no linear combinations of rows needed for Gaussian elimination.
- For this to happen for most erasure patterns, the parity check matrix needs to have a low density of ones.

11.4 Low-Density Parity Check (LDPC) Matrix

The parity check matrix in our example:

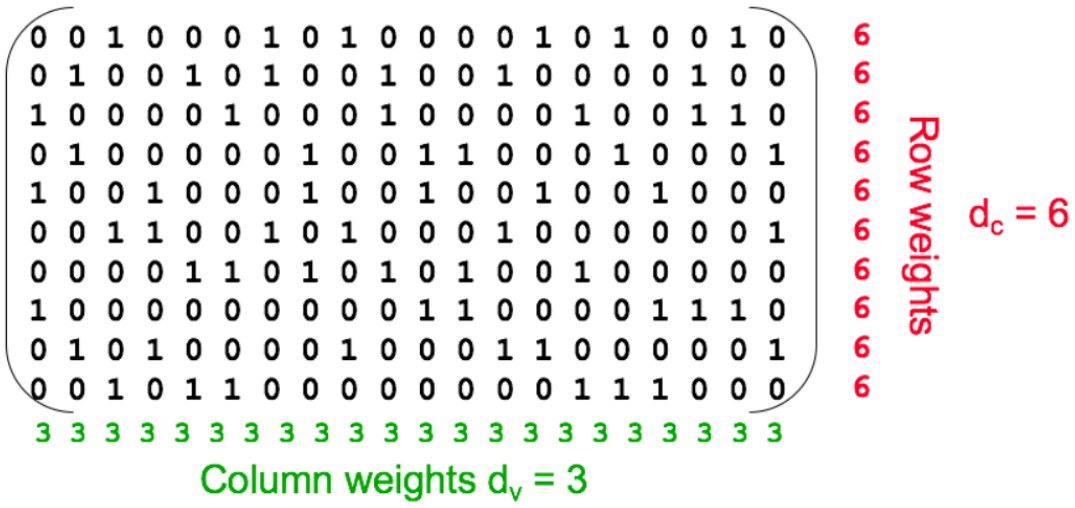


We define:

- Weight of a row/columns: number of ones in the row/columns
- This is an *irregular* LDPC code. In a regular LDPC code, all rows have the same weight, and all columns have the same weight.

11.4.1 Regular LDPC codes

Example of regular 10×20 parity check matrix:



- In a regular (n, k) LDPC code:
  - Each of the n codeword bits is involved in d_v parity check equations ('v' $\rightarrow$ variable)
    - Each variable involves in exact d_v (3) parity check equation.
  - Each of the (n-k) parity check equations involves d_c code bits ('c' $\rightarrow$ check)
    - Each row perform checks on d_c (6) variables
- Number of ones in regular parity check matrix:

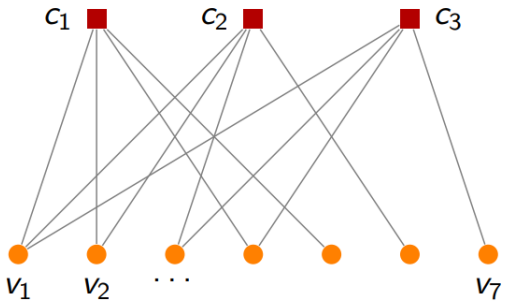
$$d_v n = d_c(n-k)$$

  - In our example: $d_v n = 3 \cdot 20 = 60 = d_c(n-k) = 6 \cdot 10 = 60$
- The design rate of a regular LDPC code is $\frac{k}{n} = 1 - \frac{d_v}{d_c}$
  - The design rate is the true code rate if the rows of the parity check matrix are linearly independent.
  - Otherwise, if there are redundant rows, which can be removed from the parity check matrix, thus making the true code rate larger than the design rate.
- To describe a class of regular LDPC code, it suffices to specify d_v and d_c
- For irregular codes, we need to specify the weight distributions on the columns and rows.
- To analyse the iterative decoder and design good LDPC matrices, it is useful to represent into a factor graph.

11.5 Factor graph of a linear code

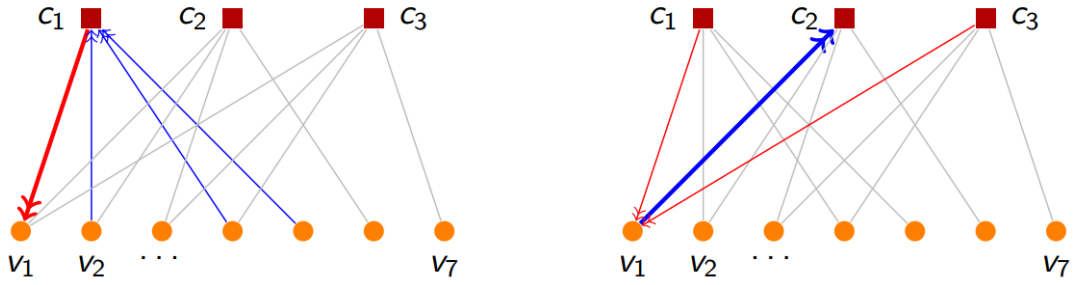
Example: $(n = 7, k = 4)$ Hamming code:

$$H = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \end{matrix}$$



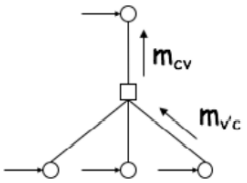
- Each column j of H is a variable node v_j (represented by a circle)
- Each row i of H is a check/constraint node c_i (represented by a square)
- A one in entry (i, j) of H means that variable node j is connected to check node i, i.e., code bit j is involved in the parity check equation descried by the ith row.

11.6 Iterative Decoding as Message Passing

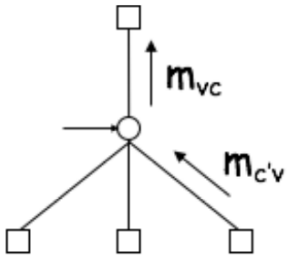


- Message passing is a class if iterative algorithms, where in each step:
  1) Each variable node v sends a message to each check node c that it is connected to
  2) Each check node c sends a message to each variable node v that it is connected to.
- These messages are denoted by $\{m_{vc}\}$ and $\{m_{cv}\}$, respectively, for $v \in \{v_1, v_2, ..., \}$ and $c \in \{c_1, c_2, ..., \}$
  - The message m_{cv} is a function of messages $\{m_{v'c}\}$ coming in from variable nodes v' connected to c (excluding v(self))
  - The message m_{vc} is a function of messages $\{m_{c'v}\}$ coming in from variable nodes c' connected to v (excluding c(self))
- The Figure above illustrates this rule for $m_{c_1 v_1}$ and $m_{v_1 c_2}$.
- The message passing rules for decoding on the BEC are as follows.
  - In the first step, m_{vc} is the channel output corresponding to bit v: (0,1, or ?), and all check nodes c send $m_{cv} =?$

– In each subsequent step: (Remember ◯– variable □– check)



$$m_{cv} = \begin{cases} \sum_{v'} m_{v'c} \pmod 2 & \text{if no } m_{v'c} = ? \\ ? & \text{if at least one incoming } m_{v'c} = ? \end{cases}$$

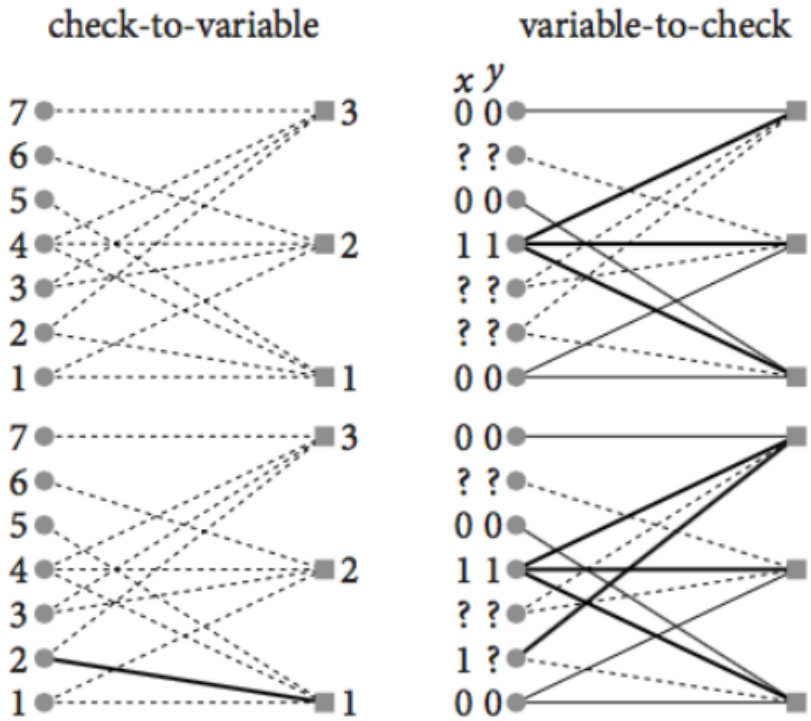


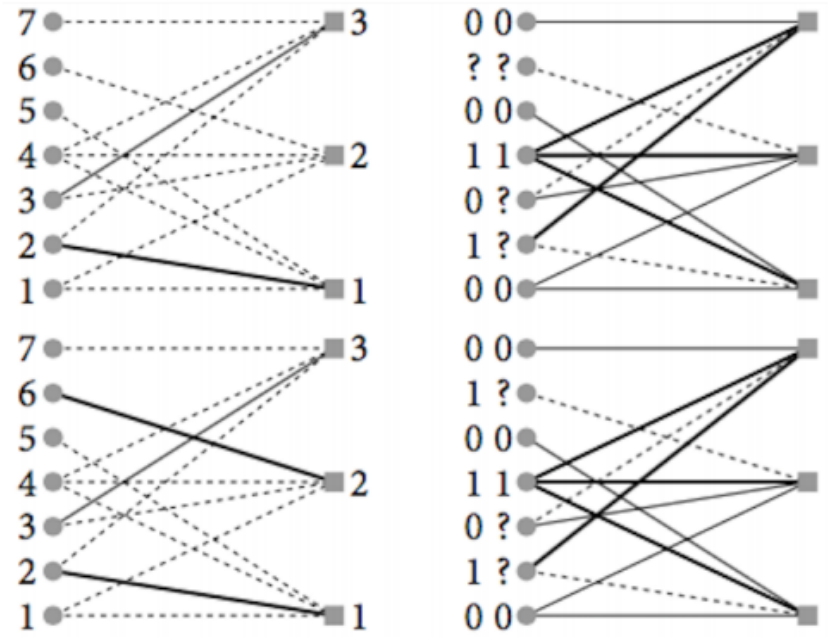
$$m_{vc} = \begin{cases} x \in \{0,1\} & \text{if } v = x \text{ OR at least one incoming } m_{c'v} = x \\ ? & \text{if } v = ? \text{ and ALL incoming } m_{c'v} = ? \end{cases}$$

– Check-to-Variable is more strict, any ? from from other incoming variable-to-check message would result in a ?

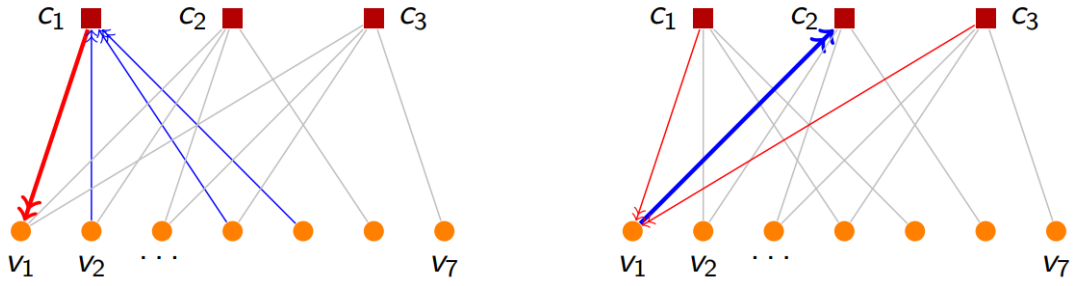
– Variable-to-check is less strict, only when all of the incoming and the self variable is ?, the outgoing message is ?

- Message Passing for the $(7, 4)$ Hamming code





- Message-passing decoding of the $(7, 4)$ Hamming code with the received word $(0, ?, ?, 1, , ?, 0)$.
- A 0 message is indicated as thin line, 1 - thick line, ? - dashed line.
- The four rows corresponding to iterations 0 to 3.
- After the first iteration we recover $x_2 = 1$
- After the second, $x_3 = 0$
- After the third, $x_6 = 1$
- The recovered codeword is $(0, 1, 0, 1, 0, 1, 0)$.



- In summary, each step of the message-passing decoder:
  - m_{vc}: variable node v tells check node c its best guess of its own value (Based on the other incoming messages).
  - m_{cv}: check node c tells variable node v what it thinks the value of v should be (Based on the other incoming messages).
- The complexity of the message passing decoder

$$\propto \underbrace{(\#\text{ edges in the graph })}_{n\bar{d}_v}(\#\text{ of iterations })$$

- $n\bar{d}_v$ is the number of ones. Therefore

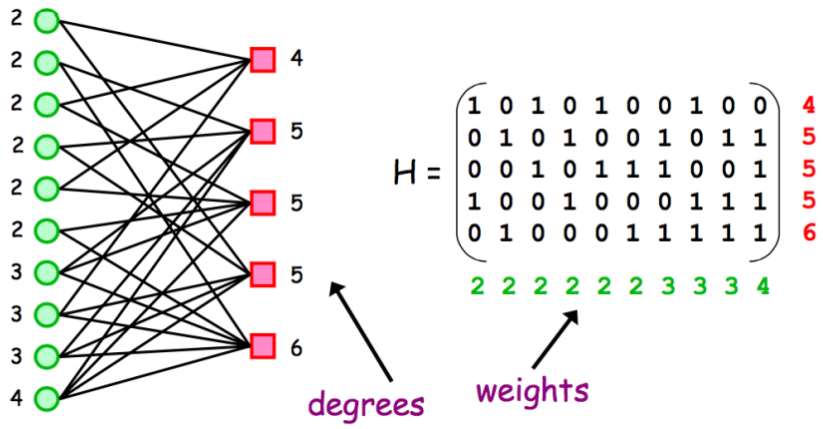
$$\text{Low density of ones in } H \longrightarrow \text{low complexity of decoder}$$

11.7 Designing good LDPC codes

- Question:
  1. Given an h, how do we predict its erasure-correcting performance with message passing decoding?
  2. How do we design H matrices that have good decoding performance?
- Simulation is a way, but it is computationally intensive. Some theoretical insight is needed.
  - The standard way to construct an LDPC code is to first choose a degree distribution that specifies the distribution of weights on the nodes/edges of the graph
  - Then fix a (large) code length n, and pick an H with this degree distribution, either at random or through some deterministic construction

11.8 Degree distributions

11.8.1 Node Perspective



- Define from the *Node perspective*
  - L_i: Fraction of *left* (variable) nodes of degree i, i.e., the fraction of columns in H with weight i.
    - For the above H, $L_2 = 6/10$ $L_3 = 3/10$ $L_4 = 1/10$
  - R_i: Fraction of *right* (check) nodes of degree i, i.e., the fraction of rows in H with weight i.
    - For the above H, $R_4 = 1/5$ $L_5 = 3/5$ $L_6 = 1/5$
- Node-perspective polynomials (easy for calculating average degree):

$$L(x) = \sum_{i=1}^{d_{v,\max}} L_i x_i, \quad R(x) = \sum_{i=1}^{d_{c,\max}} R_i x_i$$

- For the above code with $(n = 10, k = 5)$,

$$L(x) = \frac{3}{5}x^2 + \frac{3}{10}x^3 + \frac{1}{10}x^4, \quad R(x) = \frac{1}{5}x^4 + \frac{3}{5}x^5 + \frac{1}{5}x^6.$$

  - The average degree of a variable node is

$$\bar{d}_v = \sum_{i=1}^{d_{v,\max}} iL_i = L'(1).$$

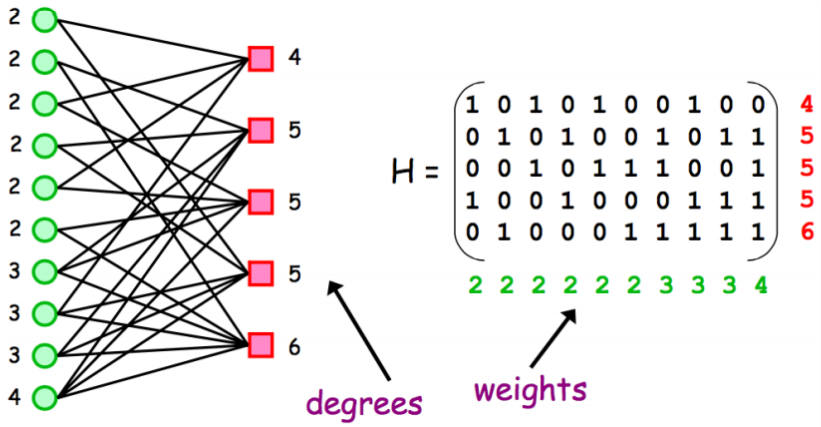
  - The average degree of a check node is

$$\bar{d}_c = \sum_{i=1}^{d_{c,\max}} iR_i = L'(1).$$

  - The number of edges in the graph, equivalent to the number of ones in H is $\bar{d}_v n = \bar{d}_c(n-k)$

Hence the design rate of the code is $\frac{k}{n} = 1 - (\bar{d}_v/\bar{d}_c)$

11.8.2 Edge Perspective



- Define from the *Edge perspective*
  - λ_i: Fraction of edges connected to variable nodes of degree i, i.e., the fraction of ones in H in columns of weight i.
    - For the example above, $\lambda_2 = 12/25$, $\lambda_3 = 9/25$, $\lambda_4 = 4/25$
  - ρ_i: Fraction of edges connected to check nodes of degree i, i.e., the fraction of ones in H in rows of weight i.
    - For the example above, $\rho_4 = 4/25$, $\rho_5 = 15/25$, $\rho_6 = 6/25$
- Edge-perspective polynomials:

$$\lambda(x) = \sum_{i=1}^{d_{v,\max}} \lambda_i x^{i-1}, \quad \rho(x) = \sum_{i=1}^{d_{c,\max}} \rho_i x^{i-1}$$

- Average node degree:

$$\bar{d}_v = \left(\int_0^1 \lambda(x)\,dx\right)^{-1} \bar{d}_c = \left(\int_0^1 \rho(x)\,dx\right)^{-1}$$

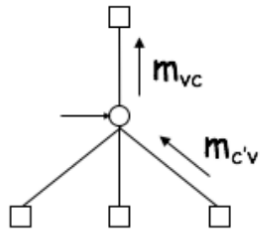
- Hence, the design rate:

$$\frac{k}{n} = 1 - (\bar{d}_v/\bar{d}_c) = 1 - \left(\int_0^1 \rho(x)\,dx \Big/ \int_0^1 \lambda(x)\,dx\right)$$

We then analyse the erasure-correcting performance (number of erasures could be corrected) of code drawn uniformly at random from the ensemble with a given $\lambda(x), \rho(x)$. The code length n is assumed to be large.

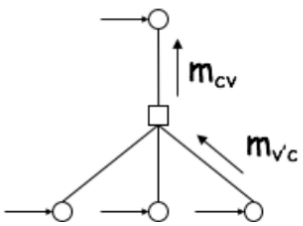
11.9 Density Evolution

- Density evolution is a technique to predict the decoding performance of codes with a given $\lambda(x), \rho(x)$ for large n. Note that fixing $\lambda(x), \rho(x)$ together also determine the rate $\frac{k}{n} = 1-(\bar{d}_v/\bar{d}_c)$.
- First consider the class of a *regular* LDPC codes. We would like to understand how regular codes with a given (d_v, d_c) perform.
- E.g. $d_v = 3$, $d_c = 6$, i.e., $\lambda(x) = x^2, \rho(x) = x^5$
  - Let p_t denotes the probability that an outgoing $v \to c$ message (along an edge picked uniformly at random) is an erasure ('?') in step t. On a BEC with erasure probability ϵ, for $t = 0$, $p_0 = \epsilon$
  - Let q_t denotes the probability that an outgoing $c \to v$ message is a '?' in step t. For $t = 0$, $q_0 = 1$ since first $c \to v$ messages are always erasures
- For $t \geq 1$ assuming all the incoming messages at each variable/check node are **Independent**, we have:



The probability that the variable AND *all* the $d_v - 1$ incoming messages being erased is:

$$p_t = \varepsilon(q_{t-1})^{d_v-1}.$$



The probability that *at least* one of the $d_c - 1$ incoming messages is erased is:

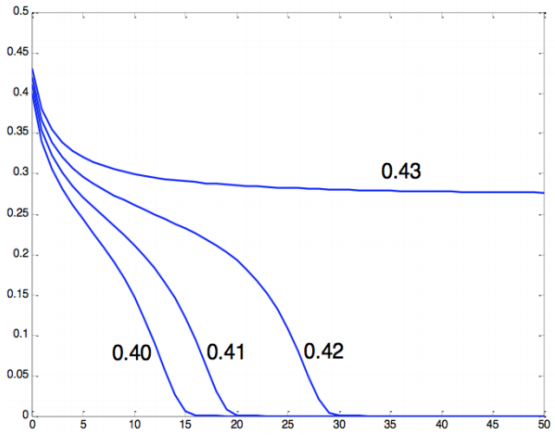
$$q_t = 1 - (1 - p_t)^{d_c-1}.$$

- Combining the two equations, we get:

$$p_t = \epsilon(1 - (1 - p_{t-1})^{d_c-1})^{d_v-1}$$

The *density evolution* recursion predicts the fraction of erased bits at the end of each step t. We initialise the recursion with $p_0 = \epsilon$. The target is to minimize p_t after iterations.

- Example: Rate 1/2 code with $d_v = 3$, $d_c = 6$. The following graph shows p_t vs. t for different ϵ : $\epsilon = 0.40, 0.41, 0.42, 0.43$



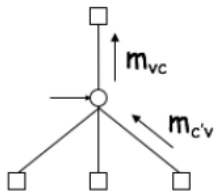
- Since p_t is a fraction of erased bits after t steps of message passing, we want to find the maximum ϵ for which p_t gets close to 0 with growing t.
  - For the $(d_v = 3, d_c = 6)$ LDPC ensemble, using density evolution this *threshold* is found to be $\epsilon^{MP} = 0.4294$.
  - The **Shannon limit**, i.e, the max.possible ϵ for reliable decoding with any rate R code, is $\epsilon^* = 1 - R = 0.5$
- A key assumption in density evolution is that the incoming messages at each node are independent.
  - This is strictly true only if there are no cycles(loops) in the factor graph, i.e., the graph is a tree.
  - A practical LDPC code is almost never completely cycle-free. But the independence assumption is usually close enough got large n as long as the factor graph does not have short cycles.

11.10 Irregular codes

Irregular LDPC ensembles give us more flexibility in the code design, and can be optimized to get reliable message passing decoding ϵ^{MP} closer to Shannon limit ϵ^*

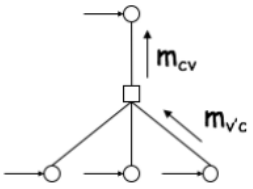
- Consider an LDPC ensemble with $\lambda(x) = \sum_i \lambda_i x^{i-1}$ and $\rho(x) = \sum_i \rho_i x_{i-1}$. Recall that:
  - λ_i is the fraction of edges connected to degree-i variable nodes.
  - ρ_i is the fraction of edges connected to degree-i check nodes.
- Similarly for irregular $(\lambda(x), \rho(x))$ ensembles:
  - p_t denotes the probability that an outgoing $v \to c$ message (along an edge picked uniformly at random) is an erasure ('?') in step t. On a BEC, start with $p_0 = \epsilon$.
  - q_t denotes the probability that an outgoing $c \to v$ message is a '?' in step t. Start with $q_0 = 1$.

- For $t \geq 1$ assuming all the incoming messages are **Independent**:



The probability that an outgoing message from a variable node with deg. i is erased is $p_{t,i} = \varepsilon q_{t-1}^{i-1}$. Hence the probability that a randomly picked edge has $m_{vc} =?$ is

$$p_t = \sum_i \lambda_i p_{t,i} = \varepsilon \sum_i \lambda_i q_{t-1}^{i-1} = \varepsilon \lambda(q_{t-1}).$$



The probability that an outgoing message from check node with deg. j is erased is $q_{t,j} = 1 - (1 - p_t)^{j-1}$. Hence the probability that a randomly picked edge has $m_{cv} =?$ is

$$q_t = \sum_j \rho_j q_{t,j} = 1 - \sum_j \rho_j (1 - p_t)^{j-1} = 1 - \rho(1 - p_t).$$

Combining the two equations, we get the density evolution equation for a $\lambda(x), \rho(x)$ ensemble:

$$p_t = \epsilon \lambda(1 - \rho(1 - p_{t-1})).$$

- For a given rate R, we want to get the maixmum possible threshold ϵ^{MP} for which $p_t \to 0$.
  - Need to optimize over $\lambda(x), \rho(x)$ such that $R = 1 - (\int \rho / \int \lambda)$. which can be difficult and is not examinable.

11.11 Summary

- For the BEC, the optimal decoder for a linear code solves the system of binary linear equations obtained from $\underline{y} H^T = 0$
- Low density parity check matrices enable fast iterative decoding.
- LDPC codes can be represented in terms of a bipartite factor graph with variable nodes on one side and check nodes on the other.
- The iterative decoder passes messages back and forth along the edges of the graph. The messages represent iteratively refined estimates of what the code-bits are.
- An LDPC code is characterized by its degree distributions $\lambda(x), \rho(x)$.
- Density evolution lets us predict the decoding performance of a given ensemble for large n. This can be used to optimize the degree distributions.

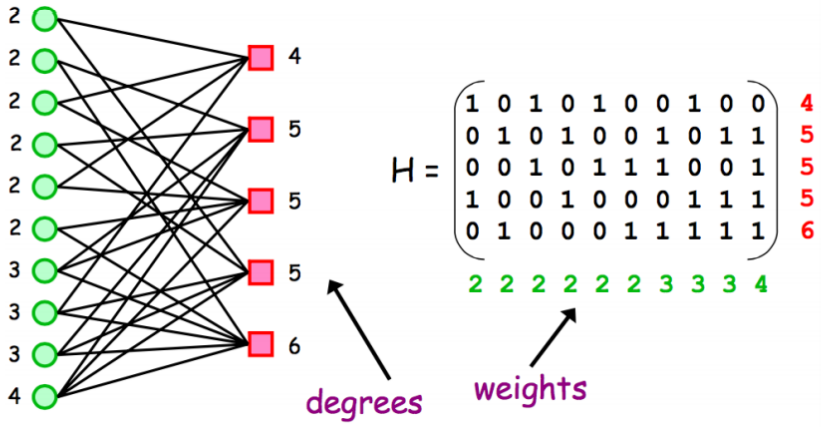
12 LDPC Codes for the general binary input Channel

Three channels will be covered:

1. **BEC**(ϵ): covered before
2. **BSC**(p)
3. **B-AWGN** channel: $Y = X + N$, where the input $X \in \{+1, -1\}$ and $N \sim \mathcal{N}(0, \sigma^2)$ is additive white Gaussian noise.

   For the B-AWGN channel, the input is generated from a binary (0/1) codeword as follows: map each 0 code-bit to $X = +1$, and each 1 code-bit to $X = -1$.

12.1 Set-up



- LDPC code above could be used, and the information bits are mapped to codeword $\underline{c} = (c_1, ..., c_n)$
- The codeword $\underline{c}$ is then transmitted over the channel. (For B-AWGN, $c_i = 0 \rightarrow X_i = +1$, $c_i = 1 \rightarrow X_i = -1$)
- The decoder receives $\underline{y} = (y_1, ..., y_2)$, and has to decode $\underline{c}$.

12.2 Optimal decoding: Block-wise vs. Bit-wise

1) To minimize the probability of **block (codeword) decoding error** is

$$\hat{\underline{c}} = \arg\max_{\underline{c} \in C} P(\underline{c}|\underline{y}) =^{(a)} \arg\max_{\underline{c} \in C} P(\underline{y}|\underline{c})$$

where (a) holds if all codewords are equally likely. Since

$$P(\underline{c}|\underline{y}) = \frac{P(\underline{c}) \cdot P(\underline{y}|\underline{c})}{P(\underline{y})}$$

and note that P($\underline{c}$) becomes constant when codewords are equally likely.

2) The optimal decoder for minimize the probability of **bit decoding error**, i.e., the probability of decoding bit c_j wrongly for $j \in \{1, ..., n\}$

$$\text{Compute } \hat{c}_j = \arg \max_{c_j \in \{0,1\}} P(c_j | \underline{y}) \quad \text{for } j = 1, ..., n.$$

Both are valid decoding rules, optimal for two different error criteria but both are computationally hard to implement for long codes. Why?

Consider the BSC(p):

1) The block-wise optimal decoding rule becomes

$$\hat{\underline{c}} = \arg \max_{\underline{c} \in C} P(\underline{y} | \underline{c}) = \arg \max_{\underline{c} \in C} p^{d(\underline{y},\underline{c})} (1-p)^{n - d(\underline{y},\underline{c})} = \arg \min_{\underline{c} \in C} d(\underline{y}, \underline{c})$$

Finding the codeword closest to $\underline{y}$ (among 2^{nR} codewords) is hard

2) The bit-wise optimal decoding rule for a BSC is:

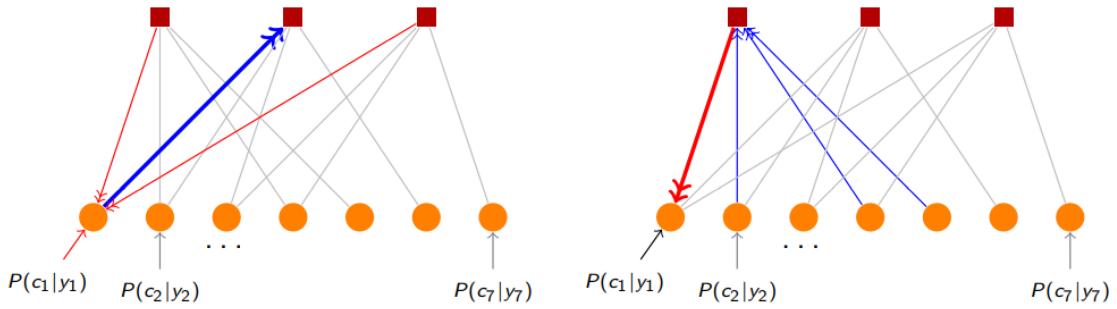
$$\hat{c}_j = \arg \max_{c_j \in \{0,1\}} P(c_j | \underline{y}) =^{(*)} \arg \max_{b \in \{0,1\}} \sum_{\underline{c} \in C : c_j = b} \left(\frac{p}{1-p} \right)^{d(\underline{y},\underline{c})}$$

This decoding is also hard, for each bit j, it requires to compute the sum on the RHS over all codewords whose jth bit $= 0$ when fix the jth bit to 0 and do the same for 1 and compare.

12.3 Message Passing Decoding

12.3.1 Setup

Since optimal decoding is infeasible, MP decoding is considered.



- The MP algorithm will approximately compute the desired bit-wise *a posteriori probabilities* (APPs) $P(c_j = 0 | \underline{y})$.
- Index the variable nodes by j, $j = 1, ..., n$, and the check nodes by i, $i = 1, ..., (n-k)$.

12.3.2 Algorithm

In each iteration, the message passing decodes computes:

1) Variable-to-check messages:
   - Each v-node j sends a message m_{ji} to each c-node i that it is connected to.

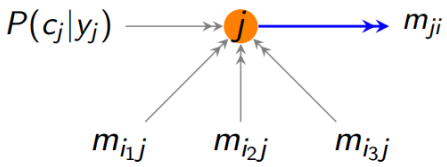
- $m_{ji}(0)$ is an **updated estimate of the posterior probability (or belief)** that the code bit $c_j = 0$
- m_{ji} is computed using the channel evidence $P(c_j|y_j)$ and all the incoming messages into j except from c-node i.

2) Check-to-variable messages:

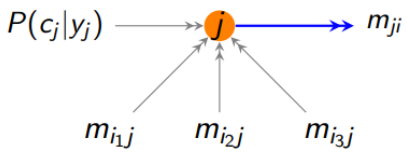
- Each c-node i sends a message m_{ij} to each v-node j that it is connected to.
- $m_{ij}(0)$ is an **updated estimate of the probability** that the parity check equation i is satisfied when $c_j = 0$.
- m_{ij} is computed using all the incoming messages into i except from v-node j.

Using the assumption that the incoming messages at each node are independent, we now derive the message updates.

12.3.3 Variable-to-check messages



$$m_{ji}(0) = P(c_j = 0 \mid y_j, \text{ checks } i_1, i_2, i_3 \text{ are all satisfied})$$
$$m_{ji}(1) = P(c_j = 1 \mid y_j, \text{ checks } i_1, i_2, i_3 \text{ are all satisfied})$$



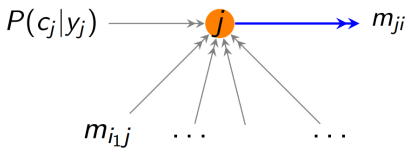
Using Bayes theorem (and assuming independence of incoming messages):

$$m_{ji}(0) = \frac{a_0}{a_0 + a_1}, \qquad m_{ji}(1) = \frac{a_1}{a_0 + a_1}$$

where:

$$a_0 = Pr\,(c_j = 0 \;\&\; \text{checks } i_1, i_2, i_3 \text{ are all satisfied} \mid y_j)$$
$$= P(c_j = 0 \mid y_j)\, m_{i_1 j}(0)\, m_{i_2 j}(0)\, m_{i_3 j}(0), \quad \text{and}$$

$$a_1 = Pr\,(c_j = 1 \;\&\; \text{checks } i_1, i_2, i_3 \text{ are all satisfied} \mid y_j)$$
$$= P(c_j = 1 \mid y_j)\, m_{i_1 j}(1)\, m_{i_2 j}(1)\, m_{i_3 j}(1).$$



In general, m_{ji}, the outgoing message from each v-node j to c-node i, is computed as follows.

- In iteration 1, $m_{ji}(0) = P(c_j = 0|y_j)$.
- For $t > 1$:

$$m_{ji}(0) = \frac{P(c_j = 0 \mid y_j)\prod_{i' \setminus i} m_{i' j}(0)}{P(c_j = 0 \mid y_j)\prod_{i' \setminus i} m_{i' j}(0) + P(c_j = 1 \mid y_j)\prod_{i' \setminus i} m_{i' j}(1)}$$

$$m_{ji}(1) = 1 - m_{ji}(0).$$

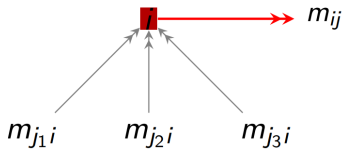
Here the notation $i' \setminus i$ denotes all the c-nodes i' connected to j *except* i.

> The message $m_{ji}(0)$ thus is an updated probability ('belief') that $c_j = 0$ based on the incoming messages $m_{i' j}$. The assumption is that the incoming messages are independent.

11 /

12.3.4 Check-to-variable messages

- For $m_{ij}(0)$, we need to compute the probability of check equation i being satisfied when $c_j = 0$.
- We do this using the incoming messages $m_{j' i}$ into c-node i.
- Recall $m_{j' i}$ is the estimated probability of $c_{j'}$ being 0 vs. 1.

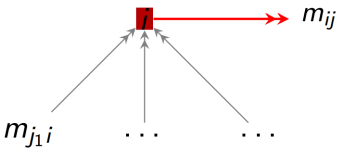


In the above example, with $c_j = 0$, the check equation i is satisfied when $(c_{j_1} = 0, c_{j_2} = 0, c_{j_3} = 0)$ or $(c_{j_1} = 1, c_{j_2} = 1, c_{j_3} = 0)$ or $(c_{j_1} = 0, c_{j_2} = 1, c_{j_3} = 1)$ or $(c_{j_1} = 1, c_{j_2} = 0, c_{j_3} = 1)$. Therefore,

$$m_{ij}(0) = m_{j_1 i}(0) m_{j_2 i}(0) m_{j_3 i}(0) \;+\; m_{j_1 i}(1) m_{j_2 i}(1) m_{j_3 i}(0)$$
$$+\; m_{j_1 i}(1) m_{j_2 i}(0) m_{j_3 i}(1) + m_{j_1 i}(0) m_{j_2 i}(1) m_{j_3 i}(1).$$

In general, we see that the message $m_{ij}(0)$ is obtained by multiplying the incoming $m_{j'i}$ such that an even number of these are evaluated at 1 and the rest at 0. The following result lets us express $m_{ij}(0)$ in a compact form: Consider a sequence of M independent binar digits $b_1, ..., b_M$ such that $P(b_k = 1) = p_k$ for all k. Then the probability that the sequence $(b_1, ..., b_M)$ contains an even number of ones is

$$\frac{1}{2} + \frac{1}{2}\prod_{k=1}^{M}(1 - 2p_k).$$



Using this result, we have

$$m_{ij}(0) = \frac{1}{2} + \frac{1}{2}\prod_{j' \backslash j}(1 - 2m_{j'i}(1)),$$

$$m_{ij}(1) = 1 - m_{ij}(0).$$

Here $j' \backslash j$ denotes all the v-nodes j' connected to i *except* j.

12.3.5 Overall decoding algorithm

The message passing decoding algorithm we just described is often called the **sum-product algorithm** or **belief propagation** as the beliefs (posterior probabilities) of each code bit being 0/1 are updated in ea ch step. The message passing updates are summarised below, where $j \in \{1, ..., n\}$ is a v-node and $i \in \{1, ..., n - k\}$ is a c-node.

At $t = 1$, set $m_{ji}(0) = P(c_j = 0|y_j)$ for all edges $j \to i$. Also set $m_i j(0) = \frac{1}{2}$ for all edges $i \to j$. For step $t > 1$:

1. variable-to-check message:

$$m_{ji}(0) \propto P(c_j = 0|y_j)\prod_{i'/i} m_{i'j}(0),$$

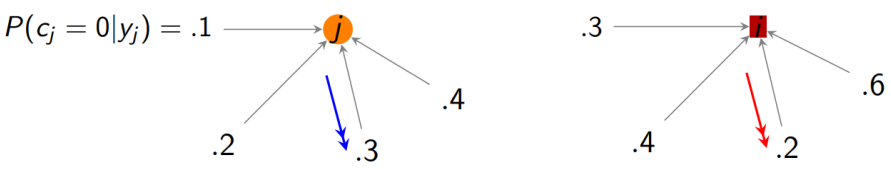
$$m_{ji}(1) \propto P(c_j = 1|y_j)\prod_{i'/i} m_{i'j}(1),$$

$$m_{ji}(0) + m_{ji}(1) = 1.$$

2. Check-to-variable message:

$$m_{ij}(0) = \frac{1}{2} + \frac{1}{2}\prod_{j'/j}(1 - 2m_{j'i}(1)), \quad m_{ij}(1) = 1 - m_{ij}(0).$$

12.3.6 Example



Calculate the outgoing messages along the indicated edges from v-node j and c-node i, assuming that the numbers on incoming edges are the messages (beliefs) evaluated at 0.

1. The outgoing message from v-node j along the indicated edge () is

$$m_{ji_2}(0) = \frac{0.1 * 0.2 * 0.4}{0.1 * 0.2 * 0.4 + 0.9 * 0.8 * 0.6} = 0.0182$$

2. The outgoing message from c-node i along the indicated edge is

$$m_{ij}(0) = \frac{1}{2} + \frac{1}{2}(1 - 2 * 0.7)(1 - 2 * (0.6))(1 - 2 * (0.4)) = 0.0508$$

12.3.7 Computing $P(c_j|y_j)$

For each variable-to-check message involves the *a posteriori* probabilities (APPs) $P(c_j|y_j)$. These can be calculated using Bayes rule and the channel transition probabilities:

$$P(c_j|y_j) = \frac{P(c_j)P(y_j|c_j)}{P(y_j)}$$

Compute APPs for BEC, BSC and the B-AWGN channel.

1. BEC(ϵ): $y_j \in \{0, 1, ?\}$

$$P(c_j = 0|y_j) = \begin{cases} 1, & \text{if } y_j = 0 \\ 0, & \text{if } y_j = 1 \\ 0.5, & \text{if } y_j = ?. \end{cases}$$

2. BSC(p): $y_j \in \{0, 1\}$

$$P(c_j = 0|y_j) = \begin{cases} 1 - p, & \text{if } y_j = 0, \\ p, & \text{if } y_j = 1, \end{cases}$$

3. B-AWGN channel: $y_j \in \mathbb{R}$

$$\begin{aligned} P(c_j = 0|y_j) &= P(X_j = +1|y_j) \\ &= \frac{P_{Y|X}(y_j| + 1)P_X(+1)}{P_{Y|X}(y_j| + 1)P_X(+1) + P_{Y|X}(y_j| - 1)P_X(-1)} \\ &= \frac{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_j-1)^2}{2\sigma^2}} \cdot 1/2}{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_j-1)^2}{2\sigma^2}} \cdot 1/2 + \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_j+1)^2}{2\sigma^2}} \cdot 1/2} \\ &= \frac{1}{1 + e^{-\frac{2y_j}{\sigma^2}}} \end{aligned}$$

In all cases, $P(c_j = 1|y_j) = 1 - P(c_j = 0|y_j)$.

As the APPs are computed as above the message passing updates are exactly the same for all three channels.

12.4 Log-domain message passing

- The messages in the above decoding algorithm involve multiplying lots of probabilities, which can cause the implementation to be numerically unstable.
- So belief propagation decoding is usually implemented with *log-likelihood* ratios (LLRs), which turns most of the multiplications into additions.

Message log-likelihood ratios can be denoted by:

$$L_{ji} := \ln \frac{m_{ji}(0)}{m_{ji}(1)}, \quad L_{ij} := \ln \frac{m_{ij}(0)}{m_{ij}(1)}.$$

To update the messages, we also need the channel evidence in LLR form:

$$L(y_j) := \ln \frac{P(c_j = 0|y_j)}{P(c_j = 1|y_j)} = \ln \frac{P(c_j = 0)P(y_j|c_j = 0)}{P(c_j = 1)P(y_j|c_j = 1)} = \ln \frac{P(y_j|c_j = 0)}{P(y_j|c_j = 1)}$$

The $L(y_j)$ can easily be computed from the APPs before:

1. BEC(ϵ): $y_j \in \{0, 1, ?\}$

$$L(y_j) = \begin{cases} \infty, & \text{if } y_j = 0 \\ \infty, & \text{if } y_j = 1 \\ 0, & \text{if } y_j =?. \end{cases}$$

2. BSC(p): $y_j \in \{0, 1\}$

$$L(y_j) = \begin{cases} \ln \frac{1-p}{p}, & \text{if } y_j = 0, \\ -\ln \frac{1-p}{p}, & \text{if } y_j = 1, \end{cases}$$

3. B-AWGN channel: $y_j \in \mathbb{R}$

$$L(y_j) = \ln \frac{1}{e^{-\frac{2y_j}{\sigma^2}}} = \frac{2y_j}{\sigma^2}.$$

The LLR-based belief propagation updates are given below, where $j \in \{1, ..., n\}$ is a v-node and $i \in \{1, ..., n-k\}$ is a c-node.

At $t = 1$, set $L_{ji} = L(y_j)$ for all edges $j \to i$, where $L(y_j)$ for each channel is given above.

Also set $L_{ij} = 0$ for all edges $i \to j$. For step $t > 1$:

1. variable-to-check-message:

$$L_{ji} = L(y_j) + \sum_{i'/i} L_{i'j}$$

2. Check-to-variable message:

$$L_{ij} = 2 \tanh^{-1} \left[\prod_{j'/j} \left(\frac{1}{2} L_{j'i} \right) \right]$$

12.5 Algorithm termination

The algorithm is run for a pre-determined number of steps, and the final LLRs for each code bit are computed as

$$L_j = L(y_j) + \sum_{i'} L_{i'j}, \quad \text{for } j = 1, ..., n$$

where the sum in this final stage is over all the checks i' connected to code bit j.

- A high positive value of L_j indicates a high probability (strong belief) that the code bit j equals 0.
- A high negative value of L_j indicates a high probability (strong belief) that the code bit j equals 1.

The final decoded codeword is $\underline{\hat{c}} = (\hat{c}_1, ..., \hat{c}_n)$ where

$$\hat{c}_j = \begin{cases} 0 & \text{if } L_j \geq 0 \\ 1 & \text{if } L_j < 0 \end{cases}$$

12.6 Complexity of decoding

- The complexity of any message passing decoder

$$\propto \underbrace{(\#\text{ edges in the graph})}\, n\bar{d}_v(\#\text{ of iterations })$$

  low density of ones in H $\rightarrow$ low complexity of decoder

- There are reduced-complexity versions of message passing.
  - The min-sum algorithm simplifies the c-to-v message involving tanh and $\tanh^{-1}$ as follows. Using $\text{sign}(x) \in \{+1, -1\}$ to denote the sign of x, we compute $L_{ij} = \text{sign}(L_i j)|L_{ij}|$, where

$$\text{sign}L_i j = \prod_{j'/j} \text{sign}(L_{j'i}) \quad \text{and} \quad |L_{ij}| = \min_{j/j} |L_{j'i}|.$$

- The performance of min-sum is slightly worse than belief propagation, but it is easier to implement in hardware.

12.7 Constructing good LDPC codes

- LDPC codes are typically constructed by first choosing a pair of degree distributions $\lambda(x), \rho(x)$ which yield the desired rate.
- Then pick an H matrix (factor graph) with this degree distribution. This can be done in a random or deterministic manner.
- For a given rate R, the goal is to choose $(\lambda(x), \rho(x))$ to give reliable decoding up to as high a noise level as possible.
- Analogous to the BEC, one can derive density evolution equations.

12.8 Summary

We derived the sum-product/Belief-propagation (BP) decoder for LDPC codes over general binary-input symmetric output channels. (BEC, BSC, and B-AWGN are three such channels.)

Key principles

- Computing the optimal bit-wise posterior prob. $P(c_j|\underline{y})$ is hard. The BP decoder iteratively approximates these using "locally" available information at each node, i.e. channel output + messages coming into the node.
- Messages passed along each edge of the graph represent beliefs about the bit the edge is connected to.
- Messages are always formed using extrinsic information: the message sent along an edge does not depend on the incoming message along the same edge.