Read chapters 1, and 2 in C&T.
Read chapter 3 in C&T.
Read section 11.1, 11.3, method of types and universal source coding.
Read chapter 4.
Read chapter 5.
Read stream code chapter 6 in “Information Theory, Inference, and Learning Algorithms” by David J.C. MacKay (available online http://www.inference.phy.cam.ac.uk/mackay/itila/)
Read chapter 7 in Cover and Thomas, channel capacity
Other than Hamming coding, today’s new material won’t be on the final.
Announcements, Assignments, and Reminders

- Homework 6 is due Friday at 5:00pm electronically. No lates accepted on this homework.
- Final Exam: Monday 3/12 at 5:00pm, but room TBD (please monitor your email, the web page, and the discussion board for where the exam will be).
- Late policy: 10% every 24 hour period that you are late, and no more than 3 days late accepted.
- Lowest grade out of all HW grades is not counted towards final grade (so you can skip one HW with impunity).
- Please do use our discussion board (https://catalyst.uw.edu/gopost/board/karna/25503/) for all questions, so that all will benefit from them being answered.
On Final

- Same format as midterm (8-9 problems).
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5\times11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
- No electronic devices, laptops, pdas, ipads, new ipads, etc.
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
- No electronic devices, laptops, pdas, ipads, new ipads, etc.
- Practice midterm will be posted in the discussion forum by tomorrow at the latest.
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
- No electronic devices, laptops, pdas, ipads, new ipads, etc.
- Practice midterm will be posted in the discussion forum by tomorrow at the latest.
- Exam time: Monday 5:00pm, still do not have room (having trouble finding one). Please monitor electronic communications over the weekend.
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
- No electronic devices, laptops, pdas, ipads, new ipads, etc.
- Practice midterm will be posted in the discussion forum by tomorrow at the latest.
- Exam time: Monday 5:00pm, still do not have room (having trouble finding one). Please monitor electronic communications over the weekend.
- Extra office hours: Next Monday: 11:00am - 1:00pm.
On Final

- Same format as midterm (8-9 problems).
- You are allowed two sides of an 8.5×11 sheet of paper (half of which can be what you had for your midterm).
- You can use computer printouts, reductions, etc. But all of your notes must fit on these two sheets.
- No electronic devices, laptops, pdas, ipads, new ipads, etc.
- Practice midterm will be posted in the discussion forum by tomorrow at the latest.
- Exam time: Monday 5:00pm, still do not have room (having trouble finding one). Please monitor electronic communications over the weekend.
- Extra office hours: Next Monday: 11:00am - 1:00pm.
- Karthik also has extra office hours he sent email about.
Class Road Map - IT-I

- L1 (1/3): Overview, Entropy
- L2 (1/5): Props. Entropy, Mutual Information, KL-Divergence
- L3 (1/10): KL-Divergence, Jensen, properties, Data Proc. Inequality
- L5 (1/17): Fano, AEP
- L6 (1/19): snow
- L6 (1/24): AEP, source coding
- L7 (1/26): Method of Types
- L9 (2/2): HMMs, coding
- L10 (2/7): Coding, Kraft,
- L11 (2/9): Huffman, midterm
- L12 (2/14): Midterm
- L13 (2/16): Shannon Games, Arithmetic
- L14 (2/21): Channel Capacity
- L16 (2/28): Shannon’s 2nd theorem.
- L18 (3/8): Hamming, continuous entropy
- L19 (3/12): Final exam, 5:00pm

Finals Week: March 12th–16th.
Joint Source/Channel Theorem: process

The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
Joint Source/Channel Theorem: process

The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
2. Transmit it over a channel.
Joint Source/Channel Theorem: process

The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
2. Transmit it over a channel.
3. If all sources could share the same channel, would be very useful.
Joint Source/Channel Theorem: process

The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
2. Transmit it over a channel.
3. If all sources could share the same channel, would be very useful.
4. I.e., perhaps the same channel coding scheme could be used regardless of the source, if the source is first compressed down to the entropy. The channel encoder/decoder need not know anything about the original source (or how to encode it).
Joint Source/Channel Theorem: process

The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
2. Transmit it over a channel.
3. If all sources could share the same channel, would be very useful.
4. I.e., perhaps the same channel coding scheme could be used regardless of the source, if the source is first compressed down to the entropy. The channel encoder/decoder need not know anything about the original source (or how to encode it).
5. Joint source/channel decoding as in the following figure:
The process would go something as follows:

1. Compress a source down to its entropy, using Huffman, LZ, arithmetic coding, etc.
2. Transmit it over a channel.
3. If all sources could share the same channel, would be very useful.
4. I.e., perhaps the same channel coding scheme could be used regardless of the source, if the source is first compressed down to the entropy. The channel encoder/decoder need not know anything about the original source (or how to encode it).
5. Joint source/channel decoding as in the following figure:

6. Maybe obvious now, but at the time (1940s) it was a revolutionary idea!
Joint Source/Channel Theorem

- Source: $V \in \mathcal{V}$ that satisfies AEP (e.g., stationary ergodic).
Joint Source/Channel Theorem

- Source: $V \in \mathcal{V}$ that satisfies AEP (e.g., stationary ergodic).
- Send $V_1:n = V_1, V_2, \ldots, V_n$ over channel, entropy rate $H(\mathcal{V})$ of stochastic process (if i.i.d., $H(\mathcal{V}) = H(V_i), \forall i$).
Joint Source/Channel Theorem

- **Source:** $V \in \mathcal{V}$ that satisfies AEP (e.g., stationary ergodic).
- **Send** $V_{1:n} = V_1, V_2, \ldots, V_n$ over channel, entropy rate $H(V)$ of stochastic process (if i.i.d., $H(V) = H(V_i), \forall i$).
- $V_{1:n} \rightarrow \text{Encoder} \rightarrow X^n \rightarrow \text{Channel} \rightarrow Y^n \rightarrow \text{Decoder} \rightarrow \hat{V}_{1:n}$

Error probability and setup:

$$P_e(n) = \sum_{y_{1:n}, v_{1:n}} \Pr(v_{1:n}) \Pr(y_{1:n} | X^n(v_{1:n})) 1\{g(y_{1:n}) \neq v_{1:n}\}$$
Joint Source/Channel Theorem

- **Source**: $V \in \mathcal{V}$ that satisfies AEP (e.g., stationary ergodic).
- **Send** $V_{1:n} = V_1, V_2, \ldots, V_n$ over channel, entropy rate $H(\mathcal{V})$ of stochastic process (if i.i.d., $H(\mathcal{V}) = H(V_i), \forall i$).
- $V_{1:n} \rightarrow \text{Encoder} \rightarrow X^n \rightarrow \text{Channel} \rightarrow Y^n \rightarrow \text{Decoder} \rightarrow \hat{V}_{1:n}$
- **Error probability and setup**:

  $$P_e^{(n)} = P(V_{1:n} \neq \hat{V}_{1:n}) \quad (1)$$

  $$= \sum_{y_{1:n}, v_{1:n}} \Pr(v_{1:n}) \Pr(y_{1:n} | X^n(v_{1:n})) \mathbf{1}\{g(y_{1:n}) \neq v_{1:n}\} \quad (2)$$
Shannon’s theorem says that there exists a sequence of codes such that if \( R < C \) the error goes to zero.
Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero.

It doesn't provide such a code, nor does it offer much insight on how to find one.
Coding and Codes

- Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero.
- It doesn’t provide such a code, nor does it offer much insight on how to find one.
- Typical set coding is not practical.
Coding and Codes

- Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero.
- It doesn’t provide such a code, nor does it offer much insight on how to find one.
- Typical set coding is not practical. Why?
Coding and Codes

- Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero.
- It doesn’t provide such a code, nor does it offer much insight on how to find one.
Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero. 

It doesn’t provide such a code, nor does it offer much insight on how to find one.


In all cases, we add enough redundancy to a message so that the original message can be decoded unambiguously.
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
- Use more reliable and expensive circuitry
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
- Use more reliable and expensive circuitry
- Improve environment (e.g., control thermal conditions, remove dust particles or even air molecules)
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
- Use more reliable and expensive circuitry
- Improve environment (e.g., control thermal conditions, remove dust particles or even air molecules)
- In compression, use more physical area/volume for each bit.
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
- Use more reliable and expensive circuitry
- Improve environment (e.g., control thermal conditions, remove dust particles or even air molecules)
- In compression, use more physical area/volume for each bit.
- In communication, use higher power transmitter, use more energy thereby making noise less of a problem.
Physical Solution to Improve Coding

- It is possible to communicate more reliably by changing physical properties to decrease the noise (e.g., decrease $p$ in a BSC).
- Use more reliable and expensive circuitry
- Improve environment (e.g., control thermal conditions, remove dust particles or even air molecules)
- In compression, use more physical area/volume for each bit.
- In communication, use higher power transmitter, use more energy thereby making noise less of a problem.
- These are not IT solutions which is what we want.
Rather than send message $x_1 x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.
Rather than send message $x_1 x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.
Rather than send message $x_1 x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $x_1 x_1 \ldots x_1 x_2 x_2 \ldots x_2 \ldots$

$k \times \quad k \times$
Rather than send message $x_1x_2\ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $\underbrace{x_1x_1\ldots x_1}_k \underbrace{x_2x_2\ldots x_2}_k \ldots$

For many channels (e.g., BSC($p < 1/2$)), error goes to zero as $k \to \infty$. 
Rather than send message $x_1x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $x_1x_1 \ldots x_1 \underbrace{x_2x_2 \ldots x_2}_k \ldots$

For many channels (e.g., BSC($p < 1/2$)), error goes to zero as $k \to \infty$.

Easy decoding: when $k$ is odd, take a majority vote (which is optimal for a BSC)
Rather than send message $x_1 x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $x_1 x_1 \ldots x_1 \underbrace{x_2 x_2 \ldots x_2}_k \ldots$

For many channels (e.g., $\text{BSC}(p < 1/2)$), error goes to zero as $k \to \infty$.

Easy decoding: when $k$ is odd, take a majority vote (which is optimal for a BSC)

On the other hand, $R \propto 1/k \to 0$ as $k \to \infty$
Rather than send message $x_1x_2 \ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $x_1x_1 \ldots x_1 x_2x_2 \ldots x_2 \ldots$

For many channels (e.g., BSC($p < 1/2$)), error goes to zero as $k \to \infty$.

Easy decoding: when $k$ is odd, take a majority vote (which is optimal for a BSC)

On the other hand, $R \propto 1/k \to 0$ as $k \to \infty$

This is really a pre-1948 way of thinking code.
Rather than send message $x_1x_2\ldots x_k$ we repeat each symbol $K$ times redundantly.

Recall our example of repeating each word in a noisy analog radio connection.

Message becomes $x_1x_1\ldots x_1 x_2x_2\ldots x_2\ldots$

For many channels (e.g., BSC($p < 1/2$)), error goes to zero as $k \to \infty$.

Easy decoding: when $k$ is odd, take a majority vote (which is optimal for a BSC)

On the other hand, $R \propto 1/k \to 0$ as $k \to \infty$

This is really a pre-1948 way of thinking code.

Thus, this is not a good code.
Repetition Code Example

- (From D. Mackay) Consider sending message $s = 0\ 0\ 1\ 0\ 1\ 1\ 0$
Repetition Code Example

(From D. Mackay) Consider sending message $s = 0010110$

- **One scenario**

\[
\begin{array}{cccccccc}
  s & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
  t & 000 & 000 & 111 & 000 & 111 & 111 & 000 \\
  n & 000 & 001 & 000 & 000 & 101 & 000 & 000 \\
  r & 000 & 001 & 111 & 000 & 010 & 111 & 000 \\
\end{array}
\]

Thus, can only correct one bit error not two.
Repetition Code Example

(From D. Mackay) Consider sending message $s = 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0$

One scenario

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$r$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Another scenario

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$r$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$\hat{s}$

corrected errors

undetected errors

Thus, can only correct one bit error not two.
(From D. Mackay) Consider sending message $s = 0 0 1 0 1 1 0$

One scenario

\[
\begin{array}{cccccccc}
  s & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
  t & 0 0 0 & 0 0 0 & 1 1 1 & 0 0 0 & 1 1 1 & 1 1 1 & 0 0 0 \\
  n & 0 0 0 & 0 0 1 & 0 0 0 & 0 0 0 & 1 0 1 & 0 0 0 & 0 0 0 \\
  r & 0 0 0 & 0 0 1 & 1 1 1 & 0 0 0 & 0 1 0 & 1 1 1 & 0 0 0 \\
\end{array}
\]

Another scenario

\[
\begin{array}{cccccccc}
  s & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
  t & 0 0 0 & 0 0 0 & 1 1 1 & 0 0 0 & 1 1 1 & 1 1 1 & 0 0 0 \\
  n & 0 0 0 & 0 0 1 & 0 0 0 & 0 0 0 & 1 0 1 & 0 0 0 & 0 0 0 \\
  r & 0 0 0 & 0 0 1 & 1 1 1 & 0 0 0 & 0 1 0 & 1 1 1 & 0 0 0 \\
\hat{s} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Corrected errors

Undetected errors

Thus, can only correct one bit error not two.
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. 
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n-1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
- I.e., $x_n \leftarrow \text{mod} \left( \sum_{i=1}^{n-1} x_i, 2 \right)$.

Thus a necessary condition for valid code word is:

$$\text{mod} \left( \sum_{i=1}^{n-1} x_i, 2 \right) = 0.$$ 

Any any instance of an odd number of errors (bit swaps) won't pass this condition (although an even number of errors will pass the condition).

Quite perfect: can not correct errors, and moreover only detects some of the kinds of errors (odd number of swaps).

On the other hand, parity checks form the basis for many sophisticated coding schemes (e.g., low-density parity check (LDPC) codes, Hamming codes etc.).

We study Hamming codes next.
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
- I.e., $x_n \leftarrow \text{mod} \left( \sum_{i=1}^{n-1} x_i, 2 \right)$.
- Thus a necessary condition for valid code word is:
  \[ \text{mod} \left( \sum_{i=1}^{n} x_i, 2 \right) = 0. \]
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n$th bit is an indicator of an odd number of 1 bits in.
  - I.e., $x_n \leftarrow \text{mod} \left( \sum_{i=1}^{n-1} x_i, 2 \right)$.
- Thus a necessary condition for valid code word is:
  \[
  \text{mod} \left( \sum_{i=1}^{n} x_i, 2 \right) = 0.
  \]
- Any any instance of an odd number of errors (bit swaps) won’t pass this condition (although an even number of errors will pass the condition).
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
- I.e., $x_n \leftarrow \text{mod} \left( \sum_{i=1}^{n-1} x_i, 2 \right)$.
- Thus a necessary condition for valid code word is:
  \[ \text{mod} \left( \sum_{i=1}^{n} x_i, 2 \right) = 0. \]
- Any any instance of an odd number of errors (bit swaps) won’t pass this condition (although an even number of errors will pass the condition).
- Quite perfect: can not correct errors, and moreover only detects some of the kinds of errors (odd number of swaps).
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
- I.e., $x_n \leftarrow \mod\left(\sum_{i=1}^{n-1} x_i, 2\right)$.
- Thus a necessary condition for valid code word is:
  $\mod\left(\sum_{i=1}^{n} x_i, 2\right) = 0$.
- Any any instance of an odd number of errors (bit swaps) won’t pass this condition (although an even number of errors will pass the condition).
- Quite perfect: can not correct errors, and moreover only detects some of the kinds of errors (odd number of swaps).
- On the other hand, parity checks form the basis for many sophisticated coding schemes (e.g., low-density parity check (LDPC) codes, Hamming codes etc.).
Simple Parity Check Code

- Binary input/output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Block sizes of $n - 1$ bits: $x_{1:n-1}$.
- $n^{th}$ bit is an indicator of an odd number of 1 bits in.
- I.e., $x_n \leftarrow \text{mod}\left(\sum_{i=1}^{n-1} x_i, 2\right)$.
- Thus a necessary condition for valid code word is:
  \[
  \text{mod}\left(\sum_{i=1}^{n} x_i, 2\right) = 0.
  \]
- Any any instance of an odd number of errors (bit swaps) won’t pass this condition (although an even number of errors will pass the condition).
- Quite perfect: can not correct errors, and moreover only detects some of the kinds of errors (odd number of swaps).
- On the other hand, parity checks form the basis for many sophisticated coding schemes (e.g., low-density parity check (LDPC) codes, Hamming codes etc.).
- We study Hamming codes next.
Best illustrated by an example.
(7, 4, 3) Hamming Codes

- Best illustrated by an example.
- Let $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. 
(7, 4, 3) Hamming Codes

- Best illustrated by an example.
- Let $\mathcal{X} = \mathcal{Y} = \{0, 1\}$.
- Fix the desired rate at $R = \frac{4}{7}$ bit per channel use.
Best illustrated by an example.

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

Fix the desired rate at $R = 4/7$ bit per channel use.

Thus, in order to send 4 data bits, we need to use the channel 7 times.
Best illustrated by an example.

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

Fix the desired rate at $R = \frac{4}{7}$ bit per channel use.

Thus, in order to send 4 data bits, we need to use the channel 7 times.

Let the four data bits be denoted $x_0, x_1, x_2, x_3 \in \{0, 1\}$. 

$(7, 4, 3)$ Hamming Codes
Best illustrated by an example.

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

Fix the desired rate at $R = 4/7$ bit per channel use.

Thus, in order to send 4 data bits, we need to use the channel 7 times.

Let the four data bits be denoted $x_0, x_1, x_2, x_3 \in \{0, 1\}$.

When we send these 4 bits, we are also going to send 3 additional parity or redundancy bits, named $x_4, x_5, x_6$. 
Best illustrated by an example.

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

Fix the desired rate at $R = 4/7$ bit per channel use.

Thus, in order to send 4 data bits, we need to use the channel 7 times.

Let the four data bits be denoted $x_0, x_1, x_2, x_3 \in \{0, 1\}$.

When we send these 4 bits, we are also going to send 3 additional parity or redundancy bits, named $x_4, x_5, x_6$.

Note: all arithmetic in the following will be mod 2. I.e. $1 + 1 = 0$, $1 + 0 = 1$, $1 = 0 - 1 = -1$, etc.
(7, 4, 3) Hamming Codes

- Parity bits determined by the following equations:

  \[ x_4 \equiv x_1 + x_2 + x_3 \mod 2 \quad (3) \]
  \[ x_5 \equiv x_0 + x_2 + x_3 \mod 2 \quad (4) \]
  \[ x_6 \equiv x_0 + x_1 + x_3 \mod 2 \quad (5) \]
(7, 4, 3) Hamming Codes

- Parity bits determined by the following equations:

\[ x_4 \equiv x_1 + x_2 + x_3 \pmod{2} \]  
\[ x_5 \equiv x_0 + x_2 + x_3 \pmod{2} \]  
\[ x_6 \equiv x_0 + x_1 + x_3 \pmod{2} \]  

- I.e., if \((x_0, x_1, x_2, x_3) = (0110)\) then \((x_4, x_5, x_6) = (011)\) and complete 7-bit codeword sent over channel would be \((0110011)\).
(7, 4, 3) Hamming Codes

- Parity bits determined by the following equations:
  \[
  x_4 \equiv x_1 + x_2 + x_3 \mod 2 \\
  x_5 \equiv x_0 + x_2 + x_3 \mod 2 \\
  x_6 \equiv x_0 + x_1 + x_3 \mod 2
  \]

- I.e., if \((x_0, x_1, x_2, x_3) = (0110)\) then \((x_4, x_5, x_6) = (011)\) and complete 7-bit codeword sent over channel would be \((0110011)\).

- We can also describe this using linear equalities as follows (all mod 2).

  \[
  x_1 + x_2 + x_3 + x_4 = 0 \\
  x_0 + x_2 + x_3 + x_5 = 0 \\
  x_0 + x_1 + x_3 + x_6 = 0
  \]
Or alternatively, as $Hx = 0$ where $x^T = (x_1, x_2, \ldots, x_7)$ and

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$ (7)
Hamming Codes

- Or alternatively, as \( Hx = 0 \) where \( x^\top = (x_1, x_2, \ldots, x_7) \) and

\[
H = \begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\]

(7)

- Notice that \( H \) is a column permutation of all non-zero length-3 column vectors.
Hamming Codes

- Or alternatively, as \( Hx = 0 \) where \( x^\top = (x_1, x_2, \ldots, x_7) \) and

\[
H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\]  \( (7) \)

- Notice that \( H \) is a column permutation of all non-zero length-3 column vectors.

- Thus the code words are defined by the null-space of \( H \). I.e., \( \{ x : Hx = 0 \} \).
Hamming Codes

- Or alternatively, as $Hx = 0$ where $x^\top = (x_1, x_2, \ldots, x_7)$ and

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (7)$$

- Notice that $H$ is a column permutation of all non-zero length-3 column vectors.
- Thus the code words are defined by the null-space of $H$. I.e.,
  $$\{x : Hx = 0\}.$$  
- Since the rank of $H$ is 3, the null-space is 4, and we expect there to be $16 = 2^4$ binary vectors in this null space.
Hamming Codes

These 16 vectors are:

0000000 0100101 1000011 1000011
0001111 0101010 1001100 1001100
0010110 0110011 1010101 1010101
0011001 0111100 1011010 1011010

(8) (9) (10) (11)
Hamming Codes: weight

Thus, any codeword is in $C = \{ x : Hx = 0 \}$. 

- Thus, if $v_1, v_2 \in C$ then $v_1 + v_2 \in C$ and $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction). 

- Weight of a code is 3, which is minimum number of ones in any non-zero codeword. 

  Why? Since columns of $H$ are all different, sum of any two columns is non-zero, so can't have any weight-2 $v$ (summing two columns is never zero). 

  Minimum weight is 3 since sum of two columns will equal another column, and sum of two equal column vectors is zero.
Hamming Codes: weight

Thus, any codeword is in \( C = \{ x : Hx = 0 \} \).

Thus, if \( v_1, v_2 \in C \) then \( v_1 + v_2 \in C \) and \( v_1 - v_2 \in C \) due to linearity (codewords closed under addition and subtraction).
Hamming Codes: weight

- Thus, any codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $v_1 + v_2 \in C$ and $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- Weight of a code is 3, which is minimum number of ones in any non-zero codeword.
Thus, any codeword is in $C = \{ x : Hx = 0 \}$.

Thus, if $v_1, v_2 \in C$ then $v_1 + v_2 \in C$ and $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).

Weight of a code is 3, which is minimum number of ones in any non-zero codeword.

Why? Since columns of $H$ are all different, sum of any two columns is non-zero, so can’t have any weight-2 $v$ (summing two columns is never zero).
Hamming Codes: weight

- Thus, any codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $v_1 + v_2 \in C$ and $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- **Weight of a code** is 3, which is minimum number of ones in any non-zero codeword.
- Why? Since columns of $H$ are all different, sum of any two columns is non-zero, so can’t have any weight-2 $v$ (summing two columns is never zero).
- Minimum weight is 3 since sum of two columns will equal another column, and sum of two equal column vectors is zero.
Hamming Codes : Distance

- Thus, any codeword is in $C = \{x : Hx = 0\}$. 
Hamming Codes: Distance

- Thus, any codeword is in $C = \{ x : Hx = 0 \}$.
- minimum distance of a code is also 3, which is minimum number of differences between any two codewords.
Hamming Codes: Distance

- Thus, any codeword is in $C = \{ x : Hx = 0 \}$.
- **Minimum distance** of a code is also 3, which is minimum number of differences between any two codewords.
- Why? Given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$. Can’t have difference (or sum, and $1+1 = 1-1$) of any two columns equaling zero, so $v_1 - v_2$ can’t differ in only two places.
Hamming Codes: Distance

- Thus, any codeword is in $C = \{ x : Hx = 0 \}$.
- Minimum distance of a code is also 3, which is minimum number of differences between any two codewords.
- Why? Given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$. Can’t have difference (or sum, and $1+1 = 1-1$) of any two columns equaling zero, so $v_1 - v_2$ can’t differ in only two places.
- Another way of saying this: if $v_1, v_2 \in C$ then $d_H(v_1, v_2) \geq 3$ where $d_H(\cdot, \cdot)$ is the Hamming distance.
Hamming Codes : Distance

- Thus, any codeword is in $C = \{ x : Hx = 0 \}$.
- **minimum distance** of a code is also 3, which is minimum number of differences between any two codewords.
- Why? Given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$. Can’t have difference (or sum, and $1+1 = 1-1$) of any two columns equaling zero, so $v_1 - v_2$ can’t differ in only two places.
- Another way of saying this: if $v_1, v_2 \in C$ then $d_H(v_1, v_2) \geq 3$ where $d_H(\cdot, \cdot)$ is the Hamming distance.
- In general, codes with large minimum distance is good because then it is possible to correct errors. I.e., if $\hat{v}$ is received codeword, then we can find $i \in \arg\min_i d_H(\hat{v}, v_i)$ as the decoding procedure.
Now a BSC($p$) (crossover probability $p$) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.
Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is
\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6).
\]
where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.
Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is \(y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)\). where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

Receiver knows \(y\) but wants to know \(x\).
Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is \(y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)\). where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

Receiver knows \(y\) but wants to know \(x\). We then compute

\[
\text{(12)}
\]
Hamming Codes : BSC

- Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice verse.
- So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is 
  \(y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)\). where 
  \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.
- Receiver knows \(y\) but wants to know \(x\). We then compute 

\[
\begin{equation}
\mathbf{s} = \mathbf{H} \mathbf{y}
\end{equation}
\]
Now a BSC(\(p\)) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice verse.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is 
\[ y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6). \]
where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

Receiver knows \(y\) but wants to know \(x\). We then compute

\[ s = Hy \]  \hspace{1cm} (12)
Now a BSC($p$) (crossover probability $p$) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if $x = (x_0, x_2, \ldots, x_6)$ is transmitted, what is received is $y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)$. where $z = (z_0, z_2, \ldots, z_6)$ is the noise vector.

Receiver knows $y$ but wants to know $x$. We then compute

$$s = H y = H(x + z)$$  \hspace{1cm} (12)
Hamming Codes : BSC

- Now a BSC\( (p) \) (crossover probability \( p \)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice verse.

- So if \( x = (x_0, x_2, \ldots, x_6) \) is transmitted, what is received is \( y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6) \). where \( z = (z_0, z_2, \ldots, z_6) \) is the noise vector.

- Receiver knows \( y \) but wants to know \( x \). We then compute

\[
s = Hy = H(x + z) = Hx + Hz
\]  
(12)
Hamming Codes: BSC

- Now a BSC(p) (crossover probability p) will change some of the bits (noise), meaning a 0 might change to a 1 and vice versa.
- So if \( x = (x_0, x_2, \ldots, x_6) \) is transmitted, what is received is
  \[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6).
\]
  where \( z = (z_0, z_2, \ldots, z_6) \) is the noise vector.
- Receiver knows \( y \) but wants to know \( x \). We then compute

\[
s = Hy = H(x + z) = Hx + Hz = 0 \tag{12}
\]
Now a BSC\((p)\) (crossover probability \(p\)) will change some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is \(y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)\). where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

Receiver knows \(y\) but wants to know \(x\). We then compute

\[
s = Hy = H(x + z) = Hx + Hz = Hz
\]

(12)
Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is \(y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)\). where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

Receiver knows \(y\) but wants to know \(x\). We then compute

\[
\begin{align*}
s &= Hy = H(x + z) = \underbrace{Hx}_{=0} + Hz
\end{align*}
\]

\(s\) is called the syndrome of \(y\). \(s = 0\) means that all parity checks are satisfied by \(y\) and is a necessary condition for a correct codeword.
Hamming Codes: BSC

Moreover, we see that $s$ is a linear combination of columns of $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$  \hspace{1cm} (13)
Moreover, we see that $s$ is a linear combination of columns of $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13)$$

Since $y = x + z$, we know $y$, so if we know $z$ we know $x$. 

Hamming Codes: BSC

- Moreover, we see that $s$ is a linear combination of columns of $H$

$$ s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13) $$

- Since $y = x + z$, we know $y$, so if we know $z$ we know $x$.
- We only need to solve for $z$ in $s = Hz$, 16 possible solutions.
Moreover, we see that $s$ is a linear combination of columns of $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

(13)

Since $y = x + z$, we know $y$, so if we know $z$ we know $x$.

We only need to solve for $z$ in $s = Hz$, 16 possible solutions.

Ex: Suppose that $y^T = 0111001$ is received, then $s^T = (101)$ and the 16 solutions are:

<table>
<thead>
<tr>
<th>Solution 1</th>
<th>Solution 2</th>
<th>Solution 3</th>
<th>Solution 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100000</td>
<td>0010011</td>
<td>0101111</td>
<td>1001001</td>
</tr>
<tr>
<td>1100011</td>
<td>0001010</td>
<td>1000110</td>
<td>1111010</td>
</tr>
<tr>
<td>0000101</td>
<td>0111001</td>
<td>1110101</td>
<td>0011100</td>
</tr>
<tr>
<td>0110110</td>
<td>1010000</td>
<td>1101100</td>
<td>1011111</td>
</tr>
</tbody>
</table>
Hamming Codes: BSC

- 16 is better than 128 (possible ≥ vectors) but still many.
Hamming Codes : BSC

- 16 is better than 128 (possible \( z \) vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC(\( p \)) with \( p < 1/2 \), the most probable solution has the least weight. Any solution with weight \( k \) has probability \( p^k \).
Hamming Codes : BSC

- 16 is better than 128 (possible \(z\) vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC\((p)\) with \(p < 1/2\), the most probable solution has the least weight. Any solution with weight \(k\) has probability \(p^k\).
- Notice that there is only one possible solution with weight 1, and this is most probable solution.
Hamming Codes : BSC

- 16 is better than 128 (possible \(z\) vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC(\(p\)) with \(p < 1/2\), the most probable solution has the least weight. Any solution with weight \(k\) has probability \(p^k\).
- Notice that there is only one possible solution with weight 1, and this is most probable solution.
- In previous example, most probable solution is \(z^T = (01000000)\) and in \(y = x + z\) with \(y^T = 0111001\) this leads to codeword \(x = 0011001\) and information bits 0011.
Hamming Codes : BSC

- 16 is better than 128 (possible $z$ vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC($p$) with $p < 1/2$, the most probable solution has the least weight. Any solution with weight $k$ has probability $p^k$.
- Notice that there is only one possible solution with weight 1, and this is most probable solution.
- In previous example, most probable solution is $z^T = (01000000)$ and in $y = x + z$ with $y^T = 0111001$ this leads to codeword $x = 0011001$ and information bits 0011.
- In fact, for any $s$, there is a unique minimum weight solution for $z$ in $s = H^T z$ (in fact, this weight is no more than 1)!.
Hamming Codes: BSC

1. 16 is better than 128 (possible $z$ vectors) but still many.
2. What is the probability of each solution? Since we are assuming a BSC($p$) with $p < 1/2$, the most probable solution has the least weight. Any solution with weight $k$ has probability $p^k$.
3. Notice that there is only one possible solution with weight 1, and this is most probable solution.
4. In previous example, most probable solution is $z^T = (01000000)$ and in $y = x + z$ with $y^T = 0111001$ this leads to codeword $x = 0011001$ and information bits 0011.
5. In fact, for any $s$, there is a unique minimum weight solution for $z$ in $s = Hz$ (in fact, this weight is no more than 1)!
6. If $s = (000)$ then the unique solution is $z = (0000000)$. 
Hamming Codes: BSC

- 16 is better than 128 (possible $z$ vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC($p$) with $p < 1/2$, the most probable solution has the least weight. Any solution with weight $k$ has probability $p^k$.
- Notice that there is only one possible solution with weight 1, and this is most probable solution.
- In previous example, most probable solution is $z^T = (01000000)$ and in $y = x + z$ with $y^T = 0111001$ this leads to codeword $x = 0011001$ and information bits 0011.
- In fact, for any $s$, there is a unique minimum weight solution for $z$ in $s = Hz$ (in fact, this weight is no more than 1)!
- If $s = (000)$ then the unique solution is $z = (0000000)$.
- For any other $s$, then $s$ must be equal to one of the columns of $H$, so we can generate $s$ by flipping the corresponding bit of $z$ on (giving weight 1 solution).
Hamming Decoding Procedure

Here is the final decoding procedure on receiving \( y \):

1. Compute the syndrome \( s = H y \).
Hamming Decoding Procedure

Here is the final decoding procedure on receiving $y$:

1. Compute the syndrome $s = H y$.
2. If $s = (000)$ set $z \leftarrow (0000000)$ and goto 4.
Hamming Decoding Procedure

Here is the final decoding procedure on receiving $y$:

1. Compute the syndrome $s = Hy$.
2. If $s = (000)$ set $z \leftarrow (0000000)$ and goto 4.
3. Otherwise, locate unique column of $H$ equal to $s$ form $z$ all zeros but with a 1 in that position.
4. Set $x \leftarrow y + z$.
5. Output $(x_0, x_1, x_2, x_3)$ as the decoded string.

This procedure can correct any single bit error, but fails when there is more than one error.
Here is the final decoding procedure on receiving $y$:

1. Compute the syndrome $s = Hy$.
2. If $s = (000)$ set $z \leftarrow (0000000)$ and goto 4.
3. Otherwise, locate unique column of $H$ equal to $s$ form $z$ all zeros but with a 1 in that position.
4. Set $x \leftarrow y + z$. 
Hamming Decoding Procedure

Here is the final decoding procedure on receiving $y$:

1. Compute the syndrome $s = H y$.

2. If $s = (000)$ set $z \leftarrow (0000000)$ and goto 4.

3. Otherwise, locate unique column of $H$ equal to $s$ form $z$ all zeros but with a 1 in that position.

4. Set $x \leftarrow y + z$.

5. output $(x_0, x_1, x_2, x_3)$ as the decoded string.
Hamming Decoding Procedure

Here is the final decoding procedure on receiving $y$:

1. Compute the syndrome $s = Hy$.
2. If $s = (000)$ set $z \leftarrow (0000000)$ and goto 4.
3. Otherwise, locate unique column of $H$ equal to $s$ form $z$ all zeros but with a 1 in that position.
4. Set $x \leftarrow y + z$.
5. output $(x_0, x_1, x_2, x_3)$ as the decoded string.

This procedure can correct any single bit error, but fails when there is more than one error.
Hamming Decoding: Venn Diagrams

- We can visualize the decoding procedure using Venn Diagrams

(a) x

\[ x_4 \]

\[ x_0 \]

\[ x_2 \]

\[ x_1 \]

\[ x_6 \]

\[ x_3 \]

\[ x_5 \]

(b) 1

\[ 1 \]

\[ 0 \]

\[ 0 \]

\[ 1 \]

\[ 0 \]

\[ 0 \]
Hamming Decoding: Venn Diagrams

- We can visualize the decoding procedure using Venn Diagrams

Here, first four bits to be sent \((x_0, x_1, x_2, x_3)\) are set as desired and parity bits \((x_4, x_5, x_6)\) are also set. Figure shows \((x_1, x_2, \ldots, x_6) = (1, 0, 0, 0, 1, 0, 1)\) with parity check bits:

\[
\begin{align*}
    x_4 &\equiv x_0 + x_1 + x_2 \pmod{2} \\
    x_5 &\equiv x_1 + x_2 + x_3 \pmod{2} \\
    x_6 &\equiv x_0 + x_2 + x_3 \pmod{2}
\end{align*}
\]
Hamming Decoding: Venn Diagrams

- The syndrome can be seen as a condition where the parity conditions are not satisfied.
Hamming Decoding: Venn Diagrams

- The syndrome can be seen as a condition where the parity conditions are not satisfied.
- Above we argued that for \( s \neq (0, 0, 0) \) there is always a one bit flip that will satisfy all parity conditions.
Example: Here, $z_1$ can be flipped to achieve parity.
Example: Here, $z_4$ can be flipped to achieve parity.
Example: And here, $z_2$ can be flipped to achieve parity.

(d)
Example: And here, there are two errors, $y_6$ and $y_2$ (marked with a *).

(d)

Flipping $y_1$ will achieve parity, but this will lead to three errors (i.e., we will switch to a wrong codeword, and since codewords have minimum Hamming distance of 3, we'll get 3 bit errors).
**Hamming Decoding: Venn Diagrams**

- Example: And here, there are two errors, $y_6$ and $y_2$ (marked with a *).

- Flipping $y_1$ will achieve parity, but this will lead to three errors (i.e., we will switch to a wrong codeword, and since codewords have minimum Hamming distance of 3, we’ll get 3 bit errors).
Many other coding algorithms.
Coding

- Many other coding algorithms.
- Reed Solomon Codes (used by CD players).
Many other coding algorithms.

Reed Solomon Codes (used by CD players).

Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.
Coding

- Many other coding algorithms.
- Reed Solomon Codes (used by CD players).
- Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.
- Convolutional codes
Many other coding algorithms.

Reed Solomon Codes (used by CD players).

Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.

Convolutional codes

Turbo codes (two convolutional codes with permutation network)
Many other coding algorithms.
Reed Solomon Codes (used by CD players).
Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.
Convolutional codes
Turbo codes (two convolutional codes with permutation network)
Low Density Parity Check (LDPC) codes.
Coding

- Many other coding algorithms.
- Reed Solomon Codes (used by CD players).
- Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.
- Convolutional codes
- Turbo codes (two convolutional codes with permutation network)
- Low Density Parity Check (LDPC) codes.
- All developed on our journey to find good codes with low rate that achieve Shannon’s promise.
Coding

- Many other coding algorithms.
- Reed Solomon Codes (used by CD players).
- Bose, Ray-Chaudhuri, Hocquenghem (BCH) codes.
- Convolutional codes
- Turbo codes (two convolutional codes with permutation network)
- Low Density Parity Check (LDPC) codes.
- All developed on our journey to find good codes with low rate that achieve Shannon’s promise.
- We may discuss LDPC and Turbo codes a bit more next quarter (but there are a few things we need to do first, such as . . . )
Entropy

\[ H(X) = - \sum_x p(x) \log p(x) \]
Entropy

- \( H(X) = - \sum_x p(x) \log p(x) \)
- All entropic quantities we’ve encountered so far have been discrete.
Entropy

\[ H(X) = - \sum_x p(x) \log p(x) \]

All entropic quantities we’ve encountered so far have been discrete.

The world is continuous, channels are continuous, noise is continuous,
Entropy

\[ H(X) = - \sum_x p(x) \log p(x) \]

All entropic quantities we’ve encountered so far have been discrete.

The world is continuous, channels are continuous, noise is continuous,

We need a theory of compression, entropy, and channel capacity that applies to such continuous domains.
Entropy

- \[ H(X) = - \sum_x p(x) \log p(x) \]
- All entropic quantities we’ve encountered so far have been discrete.
- The world is continuous, channels are continuous, noise is continuous,
- We need a theory of compression, entropy, and channel capacity that applies to such continuous domains.
- We next explore this, and continue where we left off next quarter.
Entropy

\[ H(X) = - \sum_x p(x) \log p(x) \]

- All entropic quantities we’ve encountered so far have been discrete.
- The world is continuous, channels are continuous, noise is continuous,
- We need a theory of compression, entropy, and channel capacity that applies to such continuous domains.
- We next explore this, and continue where we left off next quarter.
- Note: the material presented henceforth won’t be on the final exam.
Entropy

\[ H(X) = -\sum_x p(x) \log p(x) \]

All entropic quantities we’ve encountered so far have been discrete.

The world is continuous, channels are continuous, noise is continuous,

We need a theory of compression, entropy, and channel capacity that applies to such continuous domains.

We next explore this, and continue where we left off next quarter.

Note: the material presented henceforth won’t be on the final exam so you can go to sleep now 😊.
Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x)$$  \hspace{1cm} (17)

and $f(x) = \frac{d}{dx}F(x)$ is the density function.
Continuous/Differential Entropy

- Let $X$ now be a continuous r.v. with cumulative distribution
  \[
  F(x) = \Pr(X \leq x)
  \] (17)

  and $f(x) = \frac{d}{dx} F(x)$ is the density function.

- Let $S = \{x : f(x) > 0\}$ be the support set.
Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x) \quad (17)$$

and $f(x) = \frac{d}{dx} F(x)$ is the density function.

Let $S = \{x : f(x) > 0\}$ be the support set. Then
Continuous/Differential Entropy

• Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x) \quad (17)$$

and $f(x) = \frac{d}{dx} F(x)$ is the density function.

• Let $S = \{x : f(x) > 0\}$ be the support set. Then

**Definition 4.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) \, dx \quad (18)$$
Continuous/Differential Entropy

Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x) \quad (17)$$

and $f(x) = \frac{d}{dx} F(x)$ is the density function.

Let $S = \{x : f(x) > 0\}$ be the support set. Then

**Definition 4.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) dx \quad (18)$$

Since we integrate over only the support set, no worries about $\log 0$. 
Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x)$$  \hspace{1cm} (17)

and $f(x) = \frac{d}{dx} F(x)$ is the density function.

Let $S = \{x : f(x) > 0\}$ be the support set. Then

**Definition 4.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) \, dx$$  \hspace{1cm} (18)

Since we integrate over only the support set, no worries about $\log 0$.

Perhaps it is best to do some examples.
Continuous Entropy Of Uniform Distribution

- Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_+$. 

\[
\begin{align*}
\text{Note: continuous entropy can be both positive or negative.} \\
\text{How can entropy (which we know to mean “uncertainty”, or “information”) be negative?} \\
\text{In fact, entropy (as we’ve seen perhaps once or twice) can be interpreted as the exponent of the “volume” of a typical set.} \\
\text{Example: } 2^H(X) \text{ is the number of things that happen, on average, and can have } 2^H(X) \ll |X|. \\
\text{Consider a uniform r.v. } Y \text{ such that } 2^H(X) = |Y|. \\
\text{Thus having a negative exponent just means the volume is small.}
\end{align*}
\]
Continuous Entropy Of Uniform Distribution

- Here, \( X \sim U[0, a] \) with \( a \in \mathbb{R}_{++} \).
- Then

\[
h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a}
\]  

(19)
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = -\log \frac{1}{a} \quad (19)$$

Note: continuous entropy can be both positive or negative.
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = -\int_{0}^{a} \frac{1}{a} \log \frac{1}{a} dx = -\log \frac{1}{a} \quad (19)$$

Note: continuous entropy can be both positive or negative.

How can entropy (which we know to mean “uncertainty”, or “information”) be negative?
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} dx = - \log \frac{1}{a} \quad (19)$$

Note: continuous entropy can be both positive or negative.

How can entropy (which we know to mean “uncertainty”, or “information”) be negative?

In fact, entropy (as we’ve seen perhaps once or twice) can be interpreted as the exponent of the “volume” of a typical set.
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = -\int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = -\log \frac{1}{a} \quad (19)$$

Note: continuous entropy can be both positive or negative.

How can entropy (which we know to mean “uncertainty”, or “information”) be negative?

In fact, entropy (as we’ve seen perhaps once or twice) can be interpreted as the exponent of the “volume” of a typical set.

Example: $2^{H(X)}$ is the number of things that happen, on average, and can have $2^{H(X)} \ll |\mathcal{X}|$. 
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}^{++}$.

Then

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} dx = - \log \frac{1}{a}$$

(19)

Note: continuous entropy can be both positive or negative.

How can entropy (which we know to mean “uncertainty”, or “information”) be negative?

In fact, entropy (as we’ve seen perhaps once or twice) can be interpreted as the exponent of the “volume” of a typical set.

Example: $2^{H(X)}$ is the number of things that happen, on average, and can have $2^{H(X)} \ll |X|$.

Consider a uniform r.v. $Y$ such that $2^{H(X)} = |Y|$.
Continuous Entropy Of Uniform Distribution

- Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.
- Then
  \[
  h(X) = -\int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = -\log \frac{1}{a} \quad (19)
  \]
- Note: continuous entropy can be both positive or negative.
- How can entropy (which we know to mean “uncertainty”, or “information”) be negative?
- In fact, entropy (as we’ve seen perhaps once or twice) can be interpreted as the exponent of the “volume” of a typical set.
- Example: $2^{H(X)}$ is the number of things that happen, on average, and can have $2^{H(X)} \ll |X|$.
- Consider a uniform r.v. $Y$ such that $2^{H(X)} = |Y|$.
- Thus having a negative exponent just means the volume is small.
Normal (Gaussian) distributions are very important.
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (20) \]
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  

(20)

Let's compute this in nats.

\[ h(X) = -\int f(x) \ln f(x) \, dx \]

(21)

\[ \mathbb{E} X^2/2 \sigma^2 + \frac{1}{2} \ln(2\pi\sigma^2) \]

(22)

\[ = \frac{1}{2} \ln(e) + \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \]

(23)

\[ = \frac{1}{2} \ln(2\pi e\sigma^2) \text{ bits} \]

(24)
Normal (Gaussian) distributions are very important. We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (20) \]

Let's compute this in nats.

\[ h(X) \]

\[ (24) \]
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2} \quad (20) \]
- Let's compute this in nats.
  \[ h(X) = -\int f \ln f \]

Note: only a function of the variance \( \sigma^2 \), not the mean. Why?
So entropy of a Gaussian is monotonically related to the variance.
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  
  \[ (20) \]
- Let's compute this in nats.
  \[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \]  
  \[ (21) \]

Note: only a function of the variance \( \sigma^2 \), not the mean. Why?

So entropy of a Gaussian is monotonically related to the variance.
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  
  \[ (20) \]
- Let's compute this in nats.
  \[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \]  
  \[ (21) \]
  \[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) \]  
  \[ (24) \]
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:

\[ X \sim N(0, \sigma^2) \quad \Leftrightarrow \quad f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (20) \]

- Let's compute this in nats.

\[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (21) \]

\[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (22) \]

\[ \text{Note: only a function of the variance } \sigma^2, \text{ not the mean. Why?} \]

So entropy of a Gaussian is monotonically related to the variance.
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  

(20)

- Let's compute this in nats.

\[ h(X) = - \int f \ln f = - \int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \]  

(21)

\[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \]  

(22)

\[ = \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) \]  

(23)

Note: only a function of the variance \( \sigma^2 \), not the mean. Why?

So entropy of a Gaussian is monotonically related to the variance.
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2} \frac{x^2}{\sigma^2}} \]  

(20)

Let's compute this in nats.

\[
h(X) = - \int f \ln f = - \int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx
\]

(21)

\[
\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2)
\]

(22)

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats}
\]

(23)
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  

(20)

- Let's compute this in nats.

\[
\begin{align*}
    h(X) &= - \int f \ln f = - \int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] \, dx \\
    \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) &= \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \\
    &= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \\
    &= \frac{1}{2} \ln(2\pi e\sigma^2) \text{ bits}
\end{align*}
\]

(21, 22, 23, 24)
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \] (20)

Let's compute this in nats.

\[ h(X) = -\int f \ln f = -\int f(x) \left[-\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2}\right] dx \] (21)

\[
\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \\
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \\
= \frac{1}{2} \ln(2\pi e\sigma^2) \text{bits} 
\] (22-24)

Note: only a function of the variance \( \sigma^2 \), not the mean. Why?
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  

Let's compute this in nats.

\[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \]  

\[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \]  

\[ = \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \]  

\[ = \frac{1}{2} \ln(2\pi e\sigma^2) \text{bits} \]  

Note: only a function of the variance \( \sigma^2 \), not the mean. Why?

So entropy of a Gaussian is monotonically related to the variance.
We even have our own AEP in the continuous case, but before that a bit more intuition.
AEP lives

- We even have our own AEP in the continuous case, but before that a bit more intuition.
- In the discrete case, we have \( \Pr(x_1, x_2, \ldots, x_n) \approx 2^{-nH(X)} \) for big \( n \) and \( |A_{\epsilon}^{(n)}| = 2^{nH} = (2^H)^n \).
AEP lives

- We even have our own AEP in the continuous case, but before that a bit more intuition.

- In the discrete case, we have $\Pr(x_1, x_2, \ldots, x_n) \approx 2^{-nH(X)}$ for big $n$ and $|A_{\epsilon}^{(n)}| = 2^{nH} = (2^H)^n$.

- Thus, $2^H$ can be seen like a “side length” of an $n$-dimensional hypercube, and $2^{nH}$ is like the volume of this hypercube (or volume of the typical set).
AEP lives

- We even have our own AEP in the continuous case, but before that a bit more intuition.
- In the discrete case, we have $\Pr(x_1, x_2, \ldots, x_n) \approx 2^{-nH(X)}$ for big $n$ and $|A_\epsilon^{(n)}| = 2^{nH} = (2^H)^n$.
- Thus, $2^H$ can be seen like a “side length” of an $n$-dimensional hypercube, and $2^{nH}$ is like the volume of this hypercube (or volume of the typical set).
- So $H$ being negative just means small side length.
Things are similar for the continuous case. Indeed
Things are similar for the continuous case. Indeed

**Theorem 4.2**

Let $X_1, X_2, \ldots, X_n$ be a sequence of r.v.’s, i.i.d. $\sim f(x)$. Then

$$\frac{-1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (25)$$
Things are similar for the continuous case. Indeed

**Theorem 4.2**

Let $X_1, X_2, \ldots, X_n$ be a sequence of r.v.'s, i.i.d. $\sim f(x)$. Then

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[- \log f(X)] = h(X) \quad (25)$$

this follows via the weak law of large numbers (WLLN) just like before.
Things are similar for the continuous case. Indeed

**Theorem 4.2**

Let $X_1, X_2, \ldots, X_n$ be a sequence of r.v.'s, i.i.d. $\sim f(x)$. Then

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X)$$  \hspace{1cm} (25)

this follows via the weak law of large numbers (WLLN) just like before.

**Definition 4.3**

$$A^n(\epsilon) = \{ x_{1:n} \in S^n : | -\frac{1}{n} f(x_1, \ldots, x_n) - h(X) | \leq \epsilon \}$$
Things are similar for the continuous case. Indeed

**Theorem 4.2**

Let $X_1, X_2, \ldots, X_n$ be a sequence of r.v.'s, i.i.d. $\sim f(x)$. Then

$$\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X)$$

(25)

this follows via the weak law of large numbers (WLLN) just like before.

**Definition 4.3**

$$A^{(n)}_\epsilon = \{ x_{1:n} \in S^n : \left| -\frac{1}{n} f(x_1, \ldots, x_n) - h(X) \right| \leq \epsilon \}$$

Note: $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i)$. 
• Things are similar for the continuous case. Indeed

**Theorem 4.2**

Let $X_1, X_2, \ldots, X_n$ be a sequence of r.v.’s, i.i.d. $\sim f(x)$. Then

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X)$$  \hspace{1cm} (25)

• this follows via the weak law of large numbers (WLLN) just like before.

**Definition 4.3**

$$A_{\epsilon}^{(n)} = \{x_{1:n} \in S^n : | -\frac{1}{n}f(x_1, \ldots, x_n) - h(X)| \leq \epsilon \}$$

• Note: $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i)$.

• The above means that

$$2^{-n(h+\epsilon)} \leq f(x_{1:n}) \leq 2^{-n(h-\epsilon)}$$  \hspace{1cm} (26)
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$

(27)
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$

(27)

then we have
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$

(27)

then we have

**Theorem 4.4**
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (27)

then we have

**Theorem 4.4**

1. $\Pr(A^{(n)}_\epsilon) > 1 - \epsilon$
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (27)

then we have

**Theorem 4.4**

1. $Pr(A_{\epsilon}^{(n)}) > 1 - \epsilon$
2. $(1 - \epsilon)2^n(h(X) - \epsilon) \leq \text{Vol}(A_{\epsilon}^{(n)}) \leq 2^n(h(X) + \epsilon)$
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (28)
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$

then we have

- Theorem 4.5

$$\Pr \left( A^{(n)} \epsilon \right) > 1 - \epsilon^2 n \left( h(X) - \epsilon \right) \leq \text{Vol} \left( A^{(n)} \epsilon \right) \leq 2^n \left( h(X) + \epsilon \right)$$
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$Vol(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (28)

then we have

Theorem 4.5
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$

then we have

**Theorem 4.5**

1. $Pr(A_{\epsilon}^{(n)}) > 1 - \epsilon$
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (28)

then we have

**Theorem 4.5**

1. $\Pr(A_{\epsilon}^{(n)}) > 1 - \epsilon$

2. $(1 - \epsilon)2^{n(h(X) - \epsilon)} \leq \text{Vol}(A_{\epsilon}^{(n)}) \leq 2^{n(h(X) + \epsilon)}$
Open Discussion

What is information?
Open Discussion

- What is information?
- Look again at lecture 1 . . .
What is information?

Look again at lecture 1 . . .

And see you next quarter! 😊
Scratch Paper