Class Road Map - IT-I

L1 (9/26): Overview, Communications, Information, Entropy
L2 (10/1): Props. Entropy, Mutual Information,
L3 (10/3): KL-Divergence, Convex, Jensen, and properties.
L5 (10/10): AEP, Compression
L6 (10/15): Compression, Method of Types,
L7 (10/17): Types, U. Coding., Stoc. Processes, Entropy rates,
L8 (10/22): Entropy rates, HMMs, Coding, Kraft,
L9 (10/24): Kraft, Shannon Codes, Huffman, Shannon/Fano/Elias
L10 (10/28): Huffman, Shannon/Fano/Elias
L11 (10/29): Shannon Games,
LXX (10/31): Midterm, in class.
L12 (11/7): Arith. Coding, Channel Capacity
L13 (11/12): Channel Capacity
L14 (11/14): Channel Capacity, Shannon’s 2nd thm
L15 (11/19): Shannon’s 2nd thm, zero error codes, feedback
L16 (11/21): Joint thm, coding, hamming, diff. entropy
L17
L18
L19

Finals Week: December 12th–16th.
Cumulative Outstanding Reading

- Read chapters 1 and 2 in our book (Cover & Thomas, “Information Theory”) (including Fano’s inequality).
- Chapter 3 in our book (Cover & Thomas, “Information Theory”).
- Section 11.1 (method of types).
- Chapter 4 and 5 in our book (Cover & Thomas, “Information Theory”).
- 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 our book (Cover & Thomas, “Information Theory”).
- Read Chapter 8 in our book (Cover & Thomas, “Information Theory”).
Homework

- Homework 6 on our web page (http://j.ee.washington.edu/~bilmes/classes/ee514a_fall_2013/), due tonight, at 11:45pm.
Announcements

- Office hours, every week, **now Thursdays 4:30-5:30pm**. Can also reach me at that time via a canvas conference.
- Final assignment, need to upload pdf scan we send you.
**Theorem 16.2.2**

All rates below $C \triangleq \max_p(x) I(X; Y)$ are achievable. Specifically, for all $R < C$, there exists a sequence of $(2^{nR}, n)$ codes with maximum probability of error $\lambda^{(n)} \to 0$ as $n \to \infty$. Conversely, any $(2^{nR}, n)$ sequence of codes with $\lambda^{(n)} \to 0$ as $n \to \infty$ must have that $R < C$.

- Implications: as long as we do not code above capacity we can, for all intents and purposes, code with zero error.
- This is true for all noisy channels representable under this model.
- We’re talking about discrete channels now, but we generalize this to continuous channels in the coming weeks.
Zero Error Codes

- If $P_e^{(n)} = 0$, then $H(W|Y^n) = 0$ (no uncertainty)
- For simplicity, assume $H(W) = nR = \log M$ (i.e., uniform dist. over $\{1, 2, \ldots, M\}$). Sufficient since this is max rate under $M$ messages.
- First lets consider the case if $P_e^{(n)} = 0$, in such case it is easy to show that $R \leq C$. Then we get

$$nR = H(W) = H(W|Y^n) + I(W;Y^n) = I(W;Y^n)$$

(16.21)

$$\leq I(X^n;Y^n) \quad \text{//Since } W \rightarrow X^n \rightarrow Y^n \text{ and data proc. ineq.}$$

(16.22)

$$= H(Y^n) - H(Y^n|X^n) = H(Y^n) - \sum_{i=1}^{n} H(Y_i|Y_{1:i-1}, X^n)$$

(16.23)

(16.25)
Sequence of codes w. vanishing error must have $R < C$.

Proof that $\lambda^{(n)} \to 0$ as $n \to \infty \Rightarrow R < C$.

\[
nR = H(W) = H(W|Y^n) + I(W; Y^n) \\
\leq H(W|Y^n) + I(X^n(W); Y^n) \quad /\text{Since } W \to X^n \to Y^n \quad (16.31) \\
\leq 1 + P_e^{(n)} nR + I(X^n(W); Y^n) \quad /\text{by Fano} \\
\leq 1 + P_e^{(n)} nR + nC \quad /\text{by lemma ??} \\
\Rightarrow R \leq P_e^{(n)} R + 1/n + C \\
\Rightarrow R \leq P_e^{(n)} R + 1/n + C \quad (16.34)
\]

Now as $n \to \infty$, $P_e^{(n)} \to 0$, and $1/n \to 0$ as well. Thus

\[
\Rightarrow R < C \\
(16.35)
\]
Sequence of codes w. vanishing error must have $R < C$.

Lower bound on error:

$$P_e^{(n)} \geq 1 - \frac{C}{R} \quad (16.31)$$

generates this plot (lower bound on error):
Zero-error capacity

What if we insist on $R = C$ and $P_e = 0$. In such case, what are the requirements of any such code.

\[
nR = H(W) = H(X^n(W)) \quad \text{//if codewords distinct} \quad (16.33)
\]

\[
= H(X^n|Y^n) + I(X^n; Y^n) = I(X^n; Y^n) \quad (16.34)
\]

\[
= 0 \text{ since } P_e = 0
\]

\[
= H(Y^n) - H(Y^n|X^n) \quad (16.35)
\]

\[
= H(Y^n) - \sum_{i=1}^{n} H(Y_i|X_i) \quad (16.36)
\]

\[
= \sum_{i} H(Y_i) - \sum_{i} H(Y_i|X_i) \quad \text{//if all } Y_i \text{'s are indep} \quad (16.37)
\]

\[
= \sum_{i} I(X_i; Y_i) \quad (16.38)
\]

\[
= nC \quad \text{//if we choose } p^*(x) \in \arg\max_{p(x)} I(X; Y) \quad (16.39)
\]
Feedback for DMC

Definition 16.2.3 \((2^{nR}, n)\) feedback code

Such a code is the encoder \(X_i(W, Y_{1:i-1})\), a decoder
\[ g : Y^n \rightarrow \{1, 2, \ldots, 2^{nR}\}, \text{ and } P_e^{(n)} = \Pr(g(Y^n) \neq W) \text{ for } H(W) = nR \]
(uniform).

Definition 16.2.4 (Capacity)

The capacity with feedback \(C_{FB}\) of a DMC is the max of all rates achievable by feedback codes.

Theorem 16.2.5

\[ C_{FB} = C = \max_{p(x)} I(X; Y) \text{ for a DMC} \quad (16.33) \]
Joint Source/Channel Theorem

- Data compression: We now know that it is possible to achieve error-free compression if our average rate of compression, $R$, measured in units of bits per source symbol, is such that $R > H$ where $H$ is the entropy of the generating source distribution.
Joint Source/Channel Theorem

- **Data compression**: We now know that it is possible to achieve error-free compression if our average rate of compression, $R$, measured in units of bits per source symbol, is such that $R > H$ where $H$ is the entropy of the generating source distribution.

- **Data Transmission**: We now know that it is possible to achieve error-free communication and transmission of information if $R < C$, where $R$ is the average rate of information sent (units of bits per channel use), and $C$ is the capacity of the channel.
Joint Source/Channel Theorem

- **Data compression**: We now know that it is possible to achieve error-free compression if our average rate of compression, $R$, measured in units of bits per source symbol, is such that $R > H$ where $H$ is the entropy of the generating source distribution.

- **Data Transmission**: We now know that it is possible to achieve error-free communication and transmission of information if $R < C$, where $R$ is the average rate of information sent (units of bits per channel use), and $C$ is the capacity of the channel.

- **Q**: Does this mean that if $H < C$, we can reliably send a source of entropy $H$ over a channel of capacity $C$?
Joint Source/Channel Theorem

- Data compression: We now know that it is possible to achieve error-free compression if our average rate of compression, $R$, measured in units of bits per source symbol, is such that $R > H$ where $H$ is the entropy of the generating source distribution.

- Data Transmission: We now know that it is possible to achieve error-free communication and transmission of information if $R < C$, where $R$ is the average rate of information sent (units of bits per channel use), and $C$ is the capacity of the channel.

- Q: Does this mean that if $H < C$, we can reliably send a source of entropy $H$ over a channel of capacity $C$?

- This seems intuitively reasonable.
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).
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 $H(\mathcal{V})$ of stochastic process (if i.i.d., $H(\mathcal{V}) = H(V_i), \forall i$).
- $V_{1:n} \rightarrow \text{Encoder} \rightarrow X^n \rightarrow \text{Channel} \rightarrow Y^n \rightarrow \text{Decoder} \rightarrow \hat{V}_{1:n}$
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**:

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

$$= \sum_{y_{1:n}, v_{1:n}} \Pr(v_{1:n}) \Pr(y_{1:n}|X^n(v_{1:n})) \mathbf{1}\{g(y_{1:n}) \neq v_{1:n}\}$$  \hspace{1cm} (16.2)
Joint Source/Channel Theorem

Theorem 16.3.1 (Source/Channel Coding Theorem)

if $V_1^n$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e(n) \rightarrow 0$ if $H(V) < C$.
Joint Source/Channel Theorem

**Theorem 16.3.1 (Source/Channel Coding Theorem)**

If $V_1:n$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \to 0$ if $H(V) < C$. Conversely, if $H(V) > C$, then $P_e^{(n)} > 0$ for all $n$ and cannot send with arbitrarily low probability of error.
Theorem 16.3.1 (Source/Channel Coding Theorem)

If $V_{1:n}$ satisfies AEP, then there exists a sequence of $(2^{nR}, n)$ codes with $P_e(n) \to 0$ if $H(V) < C$. Conversely, if $H(V) > C$, then $P_e(n) > 0$ for all $n$ and cannot send with arbitrarily low probability of error.

Proof.

...
Joint Source/Channel Theorem

**Theorem 16.3.1 (Source/Channel Coding Theorem)**

If $V_{1:n}$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \to 0$ if $H(V) < C$. Conversely, if $H(V) > C$, then $P_e^{(n)} > 0$ for all $n$ and cannot send with arbitrarily low probability of error.

**Proof.**

- If $V$ satisfies AEP, then $\exists$ a set $A_{\epsilon}^{(n)}$ with $|A_{\epsilon}^{(n)}| \leq 2^{n(H(V) + \epsilon)}$ ($A_{\epsilon}^{(n)}$ has all the probability).
Theorem 16.3.1 (Source/Channel Coding Theorem)

If $V_{1:n}$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \rightarrow 0$ if $H(V) < C$. Conversely, if $H(V) > C$, then $P_e^{(n)} > 0$ for all $n$ and cannot send with arbitrarily low probability of error.

Proof.

- If $V$ satisfies AEP, then $\exists$ a set $A_{\epsilon}^{(n)}$ with $|A_{\epsilon}^{(n)}| \leq 2^{n(H(V) + \epsilon)}$ ($A_{\epsilon}^{(n)}$ has all the probability).

- We only encode the typical set, and signal an error otherwise. This contributes $\epsilon$ to $P_e$. 

...
Joint Source/Channel Theorem

**Theorem 16.3.1 (Source/Channel Coding Theorem)**

If $V_1:n$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \to 0$ if $H(V) < C$. Conversely, if $H(V) > C$, then $P_e^{(n)} > 0$ for all $n$ and cannot send with arbitrarily low probability of error.

**Proof.**

- If $V$ satisfies AEP, then $\exists$ a set $A_\epsilon^{(n)}$ with $|A_\epsilon^{(n)}| \leq 2^{n(H(V)+\epsilon)}$ ($A_\epsilon^{(n)}$ has all the probability).
- We only encode the typical set, and signal an error otherwise. This contributes $\epsilon$ to $P_e$.
- We index elements of $A_\epsilon^{(n)}$ as $\{1, 2, \ldots, 2^{n(H+\epsilon)}\}$, so need $n(H + \epsilon)$ bits.
Joint Source/Channel Theorem

**Theorem 16.3.1 (Source/Channel Coding Theorem)**

If $V_{1:n}$ satisfies AEP, then $\exists$ a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \to 0$

If $H(V) < C$. Conversely, if $H(V) > C$, then $P_e^{(n)} > 0$ for all $n$ and cannot send with arbitrarily low probability of error.

**Proof.**

- If $V$ satisfies AEP, then $\exists$ a set $A_{\epsilon}^{(n)}$ with $|A_{\epsilon}^{(n)}| \leq 2^{n(H(V)+\epsilon)}$ ($A_{\epsilon}^{(n)}$ has all the probability).

- We only encode the typical set, and signal an error otherwise. This contributes $\epsilon$ to $P_e$.

- We index elements of $A_{\epsilon}^{(n)}$ as $\{1, 2, \ldots, 2^{n(H+\epsilon)}\}$, so need $n(H+\epsilon)$ bits.

- This gives a rate of $R = H(V) + \epsilon$. If $R < C$ then the error $< \epsilon$ which we can make as small as we wish.

...
Joint Source/Channel Theorem

\[ P(n) = \Pr(V_1:n \neq \hat{V}_1:n) \leq \Pr(V_1:n / \in A(n) \epsilon) + \Pr(g(Y_n) \neq V_n | V_n \in A(n) \epsilon) < \epsilon \]

since \( R < C \)

\[ \leq \epsilon + \epsilon = 2 \epsilon \]

(16.4)

(16.5)

And the first part of the theorem is proved.

To show the converse, show that \( P(n) \rightarrow 0 \Rightarrow H(V) \leq C \) for source channel codes.
Joint Source/Channel Theorem

proof continued.

Then

\[ P_e^{(n)} = \Pr(V_{1:n} \neq \hat{V}_{1:n}) \]
\[ \leq \Pr(V_{1:n} \notin A^{(n)}_\epsilon) + \Pr(g(Y^n) \neq V^n | V^n \in A^{(n)}_\epsilon) \]
\[ < \epsilon \text{ since } R < C \]
\[ \leq \epsilon + \epsilon = 2\epsilon \]
Then

\[ P_e^{(n)} = \Pr(V_1:n \neq \hat{V}_1:n) \leq \Pr(V_1:n \notin A_\epsilon^{(n)}) + \Pr(g(Y^n) \neq V^n|V^n \in A_\epsilon^{(n)}) \leq \epsilon + \epsilon = 2\epsilon \]

And the first part of the theorem is proved.
Joint Source/Channel Theorem

... proof continued.

Then

\[ P_e^{(n)} = \Pr(V_1:n \neq \hat{V}_1:n) \]

\[ \leq \Pr(V_1:n \notin A_\epsilon^{(n)}) + \Pr(g(Y^n) \neq V^n | V^n \in A_\epsilon^{(n)}) \]  

\[ < \epsilon \text{ since } R < C \]  

\[ \leq \epsilon + \epsilon = 2\epsilon \]  

And the first part of the theorem is proved.

To show the converse, show that \( P_e^{(n)} \to 0 \Rightarrow H(\mathcal{V}) \leq C \) for source channel codes.
Joint Source/Channel Theorem

Define:

\[ X_n(V_n) : V_n \rightarrow X_n \quad \text{encoder} \] (16.6)

\[ g_n(Y_n) : Y_n \rightarrow V_n \quad \text{decoder} \] (16.7)

Now recall, original Fano says

\[ H(X|Y) \leq 1 + P_e \log |X|. \]

Here we have

\[ H(V_n|\hat{V}_n) \leq 1 + P_n e \log |V_n| = 1 + nP_e \log |X| (16.8) \]

... proof continued.
... proof continued.

Define:

\[ X^n(V^n) : V^n \rightarrow X^n \] \hspace{0.5cm} //encoder \hspace{1cm} (16.6)

\[ g_n(Y^n) : Y^n \rightarrow V^n \] \hspace{0.5cm} //decoder \hspace{1cm} (16.7)
Define:\n\[ X^n(V^n) : V^n \rightarrow \mathcal{X}^n \] \hspace{1cm} //encoder \hspace{1cm} (16.6)\n\[ g_n(Y^n) : Y^n \rightarrow V^n \] \hspace{1cm} //decoder \hspace{1cm} (16.7)\n\nNow recall, original Fano says \( H(X|Y) \leq 1 + P_e \log |\mathcal{X}| \).\n
... proof continued.
Joint Source/Channel Theorem

... proof continued.

- Define:

\[ X^n(V^n) : V^n \rightarrow X^n \quad \text{///encoder} \] \hspace{1cm} (16.6)

\[ g_n(Y^n) : Y^n \rightarrow V^n \quad \text{///decoder} \] \hspace{1cm} (16.7)

- Now recall, original Fano says \( H(X|Y) \leq 1 + P_e \log |X| \).

- Here we have

\[ H(V^n|\hat{V}^n) \leq 1 + P_e(n) \log |V^n| = 1 + nP_e \log |X| \] \hspace{1cm} (16.8)
Joint Source/Channel Theorem

... proof continued.

\[ H(V) \leq H(V_1, V_2, \ldots, V_n) = H(V_1: n) \]

\[ = \frac{1}{n} H(V_1: n | \hat{V}_1: n) + \frac{1}{n} I(V_1: n; \hat{V}_1: n) \]

\[ \leq \frac{1}{n} \left( 1 + P(n) e^{n \log |V|} \right) + \frac{1}{n} I(X_1: n; Y_1: n) \]

//by Fano

\[ \leq \frac{1}{n} + P(n) e^{n \log |V|} + C \]

//memoryless

Letting \( n \to \infty \), \( \frac{1}{n} \) and \( P(n) e \to 0 \) which leaves us with

\[ H(V) \leq C. \]
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[
H(V) \leq H(V_1, V_2, ..., V_n) = H(V_1:n)_{n}(16.9)
\]

\[
\leq \frac{1}{n} H(V_1:n | \hat{V}_1:n) + \frac{1}{n} I(V_1:n; \hat{V}_1:n) \quad \text{(16.10)}
\]

\[
\leq \frac{1}{n} \left(1 + P(n) e^{n \log |V|}\right) + \frac{1}{n} I(X_1:n; Y_1:n) \quad \text{by Fano}\n\]

\[
\leq \frac{1}{n} \left(1 + P(n) e^{n \log |V|}\right) + C \quad \text{memoryless}\n\]

Letting \(n \to \infty\), \(1/n\) and \(P e \to 0\) which leaves us with

\[
H(V) \leq C. \quad (16.13)
\]
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[ H(V) \]

\[ \leq \frac{1}{n} \left( 1 + P(e) n \log |V| \right) + \frac{1}{n} I(V_1:n; \hat{V}_1:n) \]  

\[ \leq \frac{1}{n} \left( 1 + P(e) \right) e^{n \log |V|} + C \]  

\[ (16.13) \]
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[
H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n}
\]

(16.13)
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[ H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_{1:n})}{n} \]  

(16.9)

\[ \leq \frac{1}{n} \left( 1 + P(n) e^{n \log |V|} \right) + \frac{1}{n} I(V_1; \hat{V}_1) \]  

(16.10)

\[ \leq \frac{1}{n} \left( 1 + P(n) e^{n \log |V|} \right) + \frac{1}{n} I(X_1; Y_1) \]  

V → X → Y → \hat{V} and DPP

(16.11)

\[ \leq \frac{1}{n} + P(n) e^{n \log |V|} + C \]  

memoryless

(16.12)

Letting \( n \to \infty \), \( \frac{1}{n} \) and \( P(n) e^{n \log |V|} \to 0 \) which leaves us with

\[ H(V) \leq C. \]  

(16.13)
Joint Source/Channel Theorem

proof continued.

- We get the following derivation

\[
H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_1:n)}{n} \leq \frac{1}{n} H(V_1:n|\hat{V}_1:n) + \frac{1}{n} I(V_1:n; \hat{V}_1:n)
\]

(16.10)
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[
H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_{1:n})}{n} = \frac{1}{n}H(V_{1:n}|\hat{V}_{1:n}) + \frac{1}{n}I(V_{1:n};\hat{V}_{1:n}) \leq \frac{1}{n}(1 + P_e^{(n)}n \log |\mathcal{V}|) + \frac{1}{n}I(V_{1:n};\hat{V}_{1:n}) \quad \text{//by Fano}
\]
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[
H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_{1:n})}{n} \tag{16.9}
\]

\[
= \frac{1}{n} H(V_{1:n}|\hat{V}_{1:n}) + \frac{1}{n} I(V_{1:n}; \hat{V}_{1:n}) \tag{16.10}
\]

\[
\leq \frac{1}{n} \left( 1 + P_e^{(n)} n \log |\mathcal{V}| \right) + \frac{1}{n} I(V_{1:n}; \hat{V}_{1:n}) \quad \text{//by Fano} \tag{16.11}
\]

\[
\leq \frac{1}{n} \left( 1 + P_e^{(n)} n \log |\mathcal{V}| \right) + \frac{1}{n} I(X_{1:n}; Y_{1:n}) \quad \text{//by DPP and memoryless} \tag{16.12}
\]

\[
\leq \frac{1}{n} \left( 1 + P_e^{(n)} n \log |\mathcal{V}| \right) + \frac{1}{n} I(X_{1:n}; Y_{1:n}) \quad \text{//by DPP} \tag{16.13}
\]
Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[
H(\mathcal{V}) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_{1:n})}{n} \tag{16.9}
\]

\[
= \frac{1}{n} H(V_{1:n} \mid \hat{V}_{1:n}) + \frac{1}{n} I(V_{1:n} \mid \hat{V}_{1:n}) \tag{16.10}
\]

\[
\leq \frac{1}{n} \left(1 + P_e^{(n)} n \log |\mathcal{V}|\right) + \frac{1}{n} I(V_{1:n} \mid \hat{V}_{1:n}) \tag{16.11}
\]

\[
\leq \frac{1}{n} \left(1 + P_e^{(n)} n \log |\mathcal{V}|\right) + \frac{1}{n} I(X_{1:n} \mid Y_{1:n}) \tag{16.12}
\]

\[
\leq \frac{1}{n} + P_e^{(n)} \log |\mathcal{V}| + C \tag{16.13}
\]

// by Fano

L16 F18/58 (pg.49/277)
Joint Thm
Coding
Hamming Codes
Differential Entropy

Joint Source/Channel Theorem

... proof continued.

- We get the following derivation

\[ H(V) \leq \frac{H(V_1, V_2, \ldots, V_n)}{n} = \frac{H(V_{1:n})}{n} \]  
\[ = \frac{1}{n} H(V_{1:n}|\hat{V}_{1:n}) + \frac{1}{n} I(V_{1:n}; \hat{V}_{1:n}) \]  
\[ \leq \frac{1}{n} \left(1 + P_e^{(n)} n \log |V|\right) + \frac{1}{n} I(V_{1:n}; \hat{V}_{1:n}) \quad \text{//by Fano} \]  
\[ \leq \frac{1}{n} \left(1 + P_e^{(n)} n \log |V|\right) + \frac{1}{n} I(X_{1:n}; Y_{1:n}) \quad \text{//V \to X \to Y \to \hat{V} and memoryless} \]  
\[ \leq \frac{1}{n} + P_e^{(n)} \log |V| + C \quad \text{//memoryless} \]

- Letting \( n \to \infty, 1/n \) and \( P_e \to 0 \) which leaves us with \( H(V) \leq C. \) □
Shannon’s theorem says that there exists a sequence of codes such that if $R < C$ the error goes to zero.
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.

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?
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 $x_1 x_1 \ldots x_1 x_2 x_2 \ldots x_2 \ldots$

$\underbrace{x_1 x_1 \ldots x_1}_{k \times}$ $\underbrace{x_2 x_2 \ldots x_2}_{k \times}$
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 $x_1 x_1 \ldots x_1 x_2 x_2 \ldots x_2 \ldots$

$k \times$ $k \times$

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 $x_1x_1\ldots x_1 x_2x_2\ldots x_2\ldots$

$k \times k \times$

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 $x_1x_2\ldots x_k$ 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 $k \to \infty$
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 $x_1 x_1 \ldots x_1 x_2 x_2 \ldots x_2 \ldots$

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, $R \propto \frac{1}{k} \to 0$ 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 $x_1x_1\ldots x_1 x_2x_2\ldots x_2\ldots$

$k\times$ $k\times$

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

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

\[
\begin{array}{cccccccc}
  & s & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
 s & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
t & 000 & 000 & 111 & 000 & 111 & 111 & 000 \\
 n & 000 & 001 & 000 & 000 & 101 & 000 & 000 \\
r & 000 & 001 & 111 & 000 & 010 & 111 & 000 \\
\end{array}
\]
(From D. Mackay) Consider sending message $s = 0 0 1 0 1 1 0$

**One scenario**

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

**Another scenario**

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

corrected errors → *
detected but uncorrected errors → *
Repetition Code Example

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

One scenario

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

Another scenario

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

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

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

\[ X = \mathcal{Y} = \{0, 1\} \]
Simple Parity Check Code

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

Thus a necessary condition for valid code word is:

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

Any instance of an odd number of errors (bit swaps) won't pass this condition, and such an error is hence detected. Although an even number of errors will pass the condition (error goes undetected). Parity checks cannot correct all 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 $x_{1:n-1}$. 

Thus a necessary condition for valid code word is:

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

Any instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected. Although an even number of errors will pass the condition (error goes undetected).

Hamming codes can not correct all 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 $x_{1:n-1}$.
- 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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected. Although an even number of errors will pass the condition (error goes undetected). Parity checks cannot correct all 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 $x_{1:n-1}$.
- 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 $x_{1:n-1}$.
  
  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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected.
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 $x_{1:n-1}$.
- 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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected.
- although an even number of errors will pass the condition (error goes undetected).
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 $x_{1:n-1}$.
  
  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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected.
- although an even number of errors will pass the condition (error goes undetected).
- can not correct all 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 $x_{1:n-1}$.
  
  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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected.
- although an even number of errors will pass the condition (error goes undetected).
- can not correct all 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 $x_{1:n-1}$.
- 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 instance of an odd number of errors (bit swaps) won’t pass this condition, and such an error is hence detected.
- Although an even number of errors will pass the condition (error goes undetected).
- Can not correct all 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 $X = Y = \{0, 1\}$. 
Best illustrated by an example.

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

Fix the desired rate at $R = 4/7$ bit per channel use.
(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.
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 $x_0, x_1, x_2, x_3 \in \{0, 1\}$. 
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 $x_4, x_5, x_6$. 

(7, 4, 3) Hamming Codes
(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 $x_4, x_5, x_6$.
- 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:

\[
\begin{align*}
x_4 &\equiv x_1 + x_2 + x_3 \mod 2 \\
x_5 &\equiv x_0 + x_2 + x_3 \mod 2 \\
x_6 &\equiv x_0 + x_1 + x_3 \mod 2
\end{align*}
\] (16.14, 16.15, 16.16)
(7, 4, 3) Hamming Codes

- Parity bits determined by the following equations:

\[
\begin{align*}
x_4 & \equiv x_1 + x_2 + x_3 \pmod{2} \\
x_5 & \equiv x_0 + x_2 + x_3 \pmod{2} \\
x_6 & \equiv x_0 + x_1 + x_3 \pmod{2}
\end{align*}
\] (16.14) (16.15) (16.16)

- 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:
  \[ x_4 \equiv x_1 + x_2 + x_3 \mod 2 \]  
  \[ x_5 \equiv x_0 + x_2 + x_3 \mod 2 \]  
  \[ x_6 \equiv x_0 + x_1 + x_3 \mod 2 \]  
  \[ (16.14) \]
  \[ (16.15) \]
  \[ (16.16) \]

- 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).
  \[
  \begin{align*}
  x_1 + x_2 + x_3 + x_4 &= 0 \\
  x_0 + x_2 + x_3 + x_5 &= 0 \\
  x_0 + x_1 + x_3 + x_6 &= 0
  \end{align*}
  \]  
  \[ (16.17) \]
Hamming Codes

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

$$H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}$$ (16.18)
Hamming Codes

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

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

(16.18)

- Codewords lie in null-space of \( H \)
Hamming Codes

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

  $$H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix} \quad (16.18)$$

- Codewords lie in null-space of $H$

- Notice that $H$ is a column permutation of all seven non-zero length-3 column vectors.
Hamming Codes

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

$$H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix} \quad (16.18)$$

- Codewords lie in null-space of $H$
- Notice that $H$ is a column permutation of all seven 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 $x^T = (x_1, x_2, \ldots, x_7)$ and

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$  \hspace{1cm} (16.18)

- Codewords lie in null-space of $H$
- Notice that $H$ is a column permutation of all seven 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

The 16 vectors in the nullspace (i.e., \( \{ x : Hx = 0 \} \)) are as follows:

\[
\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100011 & \quad (16.19) \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101100 & \quad (16.20) \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110101 & \quad (16.21) \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111010 & \quad (16.22)
\end{align*}
\]
Hamming Codes

- The 16 vectors in the nullspace (i.e., \( \{ x : Hx = 0 \} \)) are as follows:

  \[
  \begin{align*}
  &0000000 \quad 0100101 \quad 1000011 \quad 1100011 \\
  &0001111 \quad 0101010 \quad 1001100 \quad 1101100 \\
  &0010110 \quad 0110011 \quad 1010101 \quad 1110101 \\
  &0011001 \quad 0111100 \quad 1011010 \quad 1111010
  \end{align*}
  \]

- Note the first (highest order) four bits of these vectors range from 0 to 15 in binary (i.e., all bit strings of length 4).
The 16 vectors in the nullspace (i.e., $\{x : Hx = 0\}$) are as follows:

\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100011 \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101100 \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110101 \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111010
\end{align*}

Note the first (highest order) four bits of these vectors range from 0 to 15 in binary (i.e., all bit strings of length 4).

These first four bits are the data bits, and the 2nd three bits are the redundancy bits.
The 16 vectors in the nullspace (i.e., \( \{ x : Hx = 0 \} \)) are as follows:

\[
\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100011 \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101100 \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110101 \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111010
\end{align*}
\]  (16.19)  (16.20)  (16.21)  (16.22)

Note the first (highest order) four bits of these vectors range from 0 to 15 in binary (i.e., all bit strings of length 4).

These first four bits are the data bits, and the 2nd three bits are the redundancy bits.

The vectors constitute the codewords, any codeword must be one of the above.
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{ x : H x = 0 \}$. 
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{ x : Hx = 0 \}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.

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

Minimum number of ones in any non-zero codeword is 3. This is called the weight of a code.

Why weight 3? Suppose there was a weight-two code word with non-zeros at position $i$ and $j$. Thus, sum of columns $i$ and $j$ would be zero. But since columns of $H$ are all different, sum of any two columns is non-zero. Hence, can't have any weight-2 codeword.

Q: why can't we have a weight 1 code word? Can have weight 3 codeword, since sum of two columns will equal another column, and sum of two equal binary vectors is zero (mod 2).
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.
- Also, $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.
- Also, $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- Minimum number of ones in any non-zero codeword is 3.
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.
- Also, $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- Minimum number of ones in any non-zero codeword is 3. This is called the weight of a code.
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{ x : Hx = 0 \}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.
- Also, $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- Minimum number of ones in any non-zero codeword is 3. This is called the weight of a code.
- Why weight 3? Suppose there was a weight-two code word with non-zeros at position $i$ and $j$. Thus, sum of columns $i$ and $j$ would be zero. But since columns of $H$ are all different, sum of any two columns is non-zero. Hence, can’t have any weight-2 codeword.
Hamming Codes: weight

- Thus, any valid codeword is in $C = \{x : Hx = 0\}$.
- Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.
- Also, $v_1 - v_2 \in C$ due to linearity (codewords closed under addition and subtraction).
- Minimum number of ones in any non-zero codeword is 3. This is called the weight of a code.
- Why weight 3? Suppose there was a weight-two code word with non-zeros at position $i$ and $j$. Thus, sum of columns $i$ and $j$ would be zero. But since columns of $H$ are all different, sum of any two columns is non-zero. Hence, can’t have any weight-2 codeword.
- Q: why can’t we have a weight 1 code word?
Thus, any valid codeword is in $C = \{ x : Hx = 0 \}$.

Thus, if $v_1, v_2 \in C$ then $H(v_1 + v_2) = Hv_1 + Hv_2 = 0$ and thus $v_1 + v_2 \in C$.

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

Minimum number of ones in any non-zero codeword is 3. This is called the weight of a code.

Why weight 3? Suppose there was a weight-two code word with non-zeros at position $i$ and $j$. Thus, sum of columns $i$ and $j$ would be zero. But since columns of $H$ are all different, sum of any two columns is non-zero. Hence, can’t have any weight-2 codeword.

Q: why can’t we have a weight 1 code word?

Can have weight 3 codeword, since sum of two columns will equal another column, and sum of two equal binary vectors is zero (mod 2).
Hamming Codes: Distance

- Thus, any codeword is in $C = \{ x : Hx = 0 \}$. 
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.
- Another way of saying this: if $v_1, v_2 \in C$ then $d_H(v_1, v_2) \geq 3$ where $d_H(x, y) = \sum_i 1 \{ x(i) \neq y(i) \}$ 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.
- Another way of saying this: if \( v_1, v_2 \in C \) then \( d_H(v_1, v_2) \geq 3 \) where \( d_H(x, y) = \sum_i 1\{x(i) \neq y(i)\} \) is the Hamming distance.
- Why? Suppose \( v_1, v_2 \in C \) differ in only two places. Then \( H(v_1 - v_2) \) will be a difference or sum of two columns of \( H \) (mod 2).
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.

Another way of saying this: if $v_1, v_2 \in C$ then $d_H(v_1, v_2) \geq 3$ where $d_H(x, y) = \sum_i 1\{x(i) \neq y(i)\}$ is the Hamming distance.

Why? Suppose $v_1, v_2 \in C$ differ in only two places. Then $H(v_1 - v_2)$ will be a difference or sum of two columns of $H \pmod{2}$. But given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$. Can’t have difference or sum, $(1+1 = 1-1 \pmod{2})$ 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.
- Another way of saying this: if $v_1, v_2 \in C$ then $d_H(v_1, v_2) \geq 3$ where $d_H(x, y) = \sum_i 1\{x(i) \neq y(i)\}$ is the Hamming distance.
- Why? Suppose $v_1, v_2 \in C$ differ in only two places. Then $H(v_1 - v_2)$ will be a difference or sum of two columns of $H \pmod{2}$. But given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$. Can’t have difference or sum, $(1+1 = 1-1 \pmod{2})$ of any two columns equaling zero, so $v_1 - v_2$ can’t differ in only two places.
- 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 $i \in \text{argmin}_i d_H(\hat{v}, v_i)$ 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.
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 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.
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6) \tag{16.23}
\]

where \( z = (z_0, z_2, \ldots, z_6) \) is the noise vector.
- Receiver knows \( y \) but wants to know \( x \).
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  

(16.23)

where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

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

(16.24)
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  (16.23)

where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

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

\[
s = Hy = H(x + z) = Hx + Hz
\]  (16.24)
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 versa.
- So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is
  \[
  y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
  \]  \hspace{1cm} (16.23)
  where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.
- Receiver knows \(y\) but wants to know \(x\). We then compute
  \[
  s = H y
  \]  \hspace{1cm} (16.24)
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]

(16.23)

where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.
- Receiver knows \(y\) but wants to know \(x\). We then compute

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

(16.24)
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

\[
s = Hy = H(x + z) = \hat{H}x + Hz
\]
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 versa.
- So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is
  \[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
  \] (16.23)

where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.
- Receiver knows \(y\) but wants to know \(x\). We then compute
  \[
s = Hy = H(x + z) = Hx + Hz
  \] (16.24)

\(s\) is called the syndrome of \(y\). If \(s = 0\), then all parity checks are satisfied by \(y\) and is a necessary condition for a correct codeword.
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  

(16.23)

where \( z = (z_0, z_2, \ldots, z_6) \) is the noise vector.

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

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

(16.24)

is called the syndrome of \( y \). If \( s = 0 \), then all parity checks are satisfied by \( y \) and is a necessary condition for a correct codeword.
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]

(16.23)

where \(z = (z_0, z_2, \ldots, z_6)\) is the noise vector.

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

\[
s = H y = H(x + z) = H x + H z = H z
\]

(16.24)

\(s\) is called the syndrome of \(y\). If \(s = 0\), then all parity checks are satisfied by \(y\) and is a necessary condition for a correct codeword.
Moreover, we see that \( s \) is a linear combination of columns of \( H \)

\[
\begin{align*}
\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + z_1 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + z_2 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \cdots + z_6 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}
\end{align*}
\]
Moreover, we see that $s$ is a linear combination of columns of $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$ (16.25)

Since $y = x + z$, we know $y$, so if we know $z$ we know $x$. 
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} (16.25)

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 $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (16.25)$$

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 (which is not a codeword), then $s = Hy = (101)^T$ and the 16 solutions are:

0100000 0010011 0101111 1001001
1100011 0001010 1000110 1111010
0000101 0111001 1110101 0011100
0110110 1010000 1101100 1011111
Hamming Codes: BSC

- 16 is better than 128 (possible $z$ vectors) but still many.
16 is better than 128 (possible \( z \) vectors) but still many.

What is the probability of each solution? Since we are assuming a \( \text{BSC}(p) \) with \( p < 1/2 \), the most probable solution for \( z \) 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 for $z$ 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 for $z$ 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 for $z$ 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)!. 
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 for $z$ 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)!
- 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 for $z$ 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 $s = H\bar{z}$ (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.

This procedure can correct any single bit error, but fails when there is more than one error.
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 $x \leftarrow y + z$. 
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.
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.

- This procedure can correct any single bit error, but fails when there is more than one error.
We can visualize the decoding procedure using Venn Diagrams.

(a)

- $x_0$
- $x_1$
- $x_2$
- $x_4$
- $x_5$
- $x_6$

(b)

- 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 \((x_0, x_2, \ldots, x_6) = (1, 0, 0, 0, 1, 0, 1)\) with parity check bits:

\[
x_4 \equiv x_0 + x_1 + x_2 \pmod{2} \quad \text{(16.26)}
\]
\[
x_5 \equiv x_1 + x_2 + x_3 \pmod{2} \quad \text{(16.27)}
\]
\[
x_6 \equiv x_0 + x_2 + x_3 \pmod{2} \quad \text{(16.28)}
\]
The syndrome can be seen as a condition where the parity conditions are not satisfied.
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.

(a)  

(b)  

(c)  

(d)  

(e)  

(e')
Example: Here, $z_1$ can be flipped to achieve parity.
Example: Here, $x_4$ can be flipped to achieve parity.
Example: And here, $z_2$ can be flipped to achieve parity.
Example: And here, there are two errors, $y_6$ and $y_2$ (each of which are 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).
Example: And here, there are two errors, \( y_6 \) and \( y_2 \) (each of which are 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.
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.
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.
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 in IT-I have been discrete.
$
H(X) = - \sum_x p(x) \log p(x)
$

All entropic quantities we’ve encountered in IT-I 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 in IT-I 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 in IT-I 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 explore this next.
Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x) \quad (16.29)$$

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

  \[ F(x) = \Pr(X \leq x) \tag{16.29} \]

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

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

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

$$F(x) = \Pr(X \leq x) \quad (16.29)$$

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

$$F(x) = \Pr(X \leq x) \quad (16.29)$$

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

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

**Definition 16.6.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) \, dx \quad (16.30)$$
Continuous/Differential Entropy

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

$$F(x) = \Pr(X \leq x)$$  \hspace{1cm} (16.29)

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

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

**Definition 16.6.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) dx$$  \hspace{1cm} (16.30)

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

Continuous/Differential Entropy

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

$$F(x) = \Pr(X \leq x) \quad (16.29)$$

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

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

**Definition 16.6.1 (differential entropy $h(X)$)**

$$h(X) = - \int_{S} f(x) \log f(x) \, dx \quad (16.30)$$

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

- Perhaps it is best to do some examples.
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.
Continuous Entropy Of Uniform Distribution

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

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a} \quad (16.31)$$
Continuous Entropy Of Uniform Distribution

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

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \left( \frac{1}{a} \right) dx = - \log \frac{1}{a} \quad (16.31)$$

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

Then

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

(16.31)

Note: continuous entropy can be both positive or negative.

How can entropy (which we know to mean “uncertainty”, or “information”) be negative?
Continuous Entropy Of Uniform Distribution

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

\[
h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = -\log \frac{1}{a}
\]  

(16.31)

- 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.
Continuous Entropy Of Uniform Distribution

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

$$h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} \, dx = -\log \frac{1}{a} \quad (16.31)$$

- 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 $2^{H(X)} \ll |X|$. 

Prof. Jeff Bilmes
Continuous Entropy Of Uniform Distribution

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

$$ h(X) = - \int_0^a \frac{1}{a} \log \frac{1}{a} dx = - \log \frac{1}{a} \quad (16.31) $$

- 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 $2^{H(X)} \ll |\mathcal{X}|$.
- Consider a uniform r.v. $Y$ such that $2^{H(X)} = |\mathcal{Y}|$. 
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a} \quad (16.31)$$

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 $2^{H(X)} \ll |\mathcal{X}|$.

Consider a uniform r.v. $Y$ such that $2^{H(X)} = |\mathcal{Y}|$.

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:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]
Normal (Gaussian) distributions are very important. We have:

\[ X \sim N(0, \sigma^2) \quad \Leftrightarrow \quad f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

Let's compute the entropy of \( f \) in nats.

\[
\begin{align*}
\mathbb{E}[X^2] &= \frac{1}{\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) \\
\mathbb{H}(X) &= \frac{1}{2} \ln(\sqrt{e}) + \frac{1}{2} \ln(2\pi\sigma^2) \\
&= \frac{1}{2} \ln(2\pi e\sigma^2) \text{ nats} \\
&= \frac{1}{2} \ln(2\pi e) + \frac{1}{2} \ln(\sigma^2) \text{ bits}
\end{align*}
\] (16.35)
Normal (Gaussian) distributions are very important. We have:

$$X \sim N(0, \sigma^2) \Leftrightarrow f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2}$$  \hspace{1cm} (16.32)$$

Let's compute the entropy of $f$ in nats.

$$h(X)$$

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:

\[ X \sim N(0, \sigma^2) \Leftrightarrow f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

Let's compute the entropy of \( f \) in nats.

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

(16.35)
Normal (Gaussian) distributions are very important. We have:

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

Let's compute the entropy of \( f \) in nats.

\[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (16.33) \]

\[ \text{(16.35)} \]
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  
  (16.32)

- Let's compute the entropy of \( f \) in nats.
  \[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \]  
  (16.33)

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

(16.35)
Continuous Entropy Of Normal Distribution

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

\[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \]  \hspace{1cm} (16.32)

- Let's compute the entropy of \( f \) in nats.

\[ h(X) = -\int f \ln f = -\int f(x) \left[-\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2}\right] \, dx \]  \hspace{1cm} (16.33)

\[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \]  \hspace{1cm} (16.34)

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

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

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

\[
X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2}
\] (16.32)

- Let's compute the entropy of \( f \) in nats.

\[
h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx
\] (16.33)

\[
\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2)
\] (16.34)

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2)
\] (16.35)
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- We have:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

- Let's compute the entropy of \( f \) in nats.
  \[
  h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx 
  \]
  \[
  \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) 
  \]
  \[
  = \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{ nats} 
  \quad (16.35) 
  \]
Continuous Entropy Of Normal Distribution

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

\[ X \sim N(0, \sigma^2) \quad \iff \quad f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

- Let's compute the entropy of \( f \) in nats.

\[
h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (16.33)
\]

\[
\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln (2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln (2\pi\sigma^2) \quad (16.34)
\]

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln (2\pi\sigma^2) = \frac{1}{2} \ln (2\pi e\sigma^2) \text{nats} \times \left( \frac{1}{\ln 2} \text{bits/nats} \right) \quad (16.35)
\]
Continuous Entropy Of Normal Distribution

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

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

(let's compute the entropy of \( f \) in nats.

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

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

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \times \left( \frac{1}{\ln 2} \text{bits/nats} \right)
\]

\[
= \frac{1}{2} \log(2\pi e\sigma^2) \text{bits}
\]
Normal (Gaussian) distributions are very important.

We have:

\[ X \sim N(0, \sigma^2) \quad \Leftrightarrow \quad f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (16.32) \]

Let's compute the entropy of \( f \) in nats.

\[
h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] \, dx \quad (16.33)
\]

\[
\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (16.34)
\]

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{ nats} \times \left( \frac{1}{\ln 2} \right) \text{ bits/nats}
\]

\[
= \frac{1}{2} \log(2\pi e\sigma^2) \text{ bits} \quad (16.35)
\]

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

We have:

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

(16.32)

Let's compute the entropy of \( f \) in nats.

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

(16.33)

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

(16.34)

\[
= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \times \left( \frac{1}{\ln 2} \text{bits/nats} \right) 
\]

(16.35)

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.
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).
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 could mean small side length (small \( 2^H \) but still positive).
Things are similar for the continuous case. Indeed
Things are similar for the continuous case. Indeed

**Theorem 16.6.2**

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

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (16.36)$$
AEP

Things are similar for the continuous case. Indeed

**Theorem 16.6.2**

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

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[- \log f(X)] = h(X) \quad (16.36)$$

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

**Theorem 16.6.2**

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} (16.36)

this follows via the weak law of large numbers (WLLN) just like in the discrete case.

**Definition 16.6.3**

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

**Theorem 16.6.2**

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

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

(16.36)

this follows via the weak law of large numbers (WLLN) just like in the discrete case.

**Definition 16.6.3**

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

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

- Things are similar for the continuous case. Indeed

**Theorem 16.6.2**

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} (16.36)

- this follows via the weak law of large numbers (WLLN) just like in the discrete case.

**Definition 16.6.3**

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

- Note: $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i)$.
- Thus, we have upper/lower bounds on the probability

$$2^{-n(h+\epsilon)} \leq f(x_{1:n}) \leq 2^{-n(h-\epsilon)}$$  \hspace{1cm} (16.37)
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

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

(16.38)
AEP

- 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} (16.38)

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

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

(16.38)

- then we have

Theorem 16.6.4
• The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

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

• then we have

**Theorem 16.6.4**

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} (16.38)

then we have

**Theorem 16.6.4**

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

2. $(1 - \epsilon)2^{n(h(X) - \epsilon)} \leq \text{Vol}(A^{(n)}_\epsilon) \leq 2^{n(h(X) + \epsilon)}$
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} (16.38)

then we have

**Theorem 16.6.4**

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

2. $(1 - \epsilon)2^{n(h(X) - \epsilon)} \leq \text{Vol}(A^{(n)}_{\epsilon}) \leq 2^{n(h(X) + \epsilon)}$

Note this is a bound on volume $\text{Vol}(A^{(n)}_{\epsilon})$ of typical set.
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n \quad (16.38)$$

then we have

**Theorem 16.6.4**

1. $\Pr(A^{(n)}_\epsilon) > 1 - \epsilon$
2. $(1 - \epsilon)2^n(h(X) - \epsilon) \leq \text{Vol}(A^{(n)}_\epsilon) \leq 2^n(h(X) + \epsilon)$

Note this is a bound on volume $\text{Vol}(A^{(n)}_\epsilon)$ of typical set.

In discrete AEP, we bound cardinality of typical set $|A^{(n)}_\epsilon|$, and never need entropy to be negative — $H(X) \geq 0$ suffices to limit $|A^{(n)}_\epsilon|$ down to its lowest sensible value, namely 1.
proof of theorem 16.6.4.

1: First,

\[ p(A^{(n)}_\epsilon) \]

(16.40)
proof of theorem 16.6.4.

1: First,

\[ p(A^{(n)}_\varepsilon) = \int_{x_1:n \in A^{(n)}_\varepsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  

(16.39)

\[ \Pr(\left|\left|\sum_{i=1}^{n} x_i - h(X)\right|\right| \leq \varepsilon) \geq 1 - \varepsilon \]  

(16.40)
proof of theorem 16.6.4.

1: First,

\[ p(A_{\epsilon}^{(n)}) = \int_{x_1: n \in A_{\epsilon}(n)} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  
\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  

(16.39)

(16.40)
proof of theorem 16.6.4.

1: First,

\[ p(A_{\epsilon}^{(n)}) = \int_{x_1:n \in A_{\epsilon}^{(n)}} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  \hspace{1cm} (16.39)

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  \hspace{1cm} (16.40)

for big enough \( n \) which follows from the WLLN.
proof of theorem 16.6.4.

1: First,

\[
p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n
\]

\[
= \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon
\]

for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[
(16.42)
\]
AEP

proof of theorem 16.6.4.

1: First,

\[
p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n
\]

(16.39)

\[
= \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon
\]

(16.40)

for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[
\Rightarrow \text{Vol}(A^{(n)}_\epsilon) \leq 2^{-n(h(X) + \epsilon)}
\]

(16.42)
proof of theorem 16.6.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  

(16.39)  

(16.40)

for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  

(16.42)
proof of theorem 16.6.4.

1: First, 

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  

for big enough \( n \) which follows from the WLLN.

2: Next, we have 

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \geq \int_{A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  

\[ \geq \int_{A^{(n)}_\epsilon} \frac{1}{2^n} \]  

\[ \geq 1 - \epsilon \]  

...
proof of theorem 16.6.4.

1: First,

\[ p(A_{\epsilon}^{(n)}) = \int_{x_1:n \in A_{\epsilon}^{(n)}} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  

(16.39)  

(16.40)

for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \geq \int_{A_{\epsilon}^{(n)}} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]

\[ \geq \int_{A_{\epsilon}^{(n)}} 2^{-n(h(X) + \epsilon)} \, dx_{1:n} \]  

(16.41)  

(16.42)
proof of theorem 16.6.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]

for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \geq \int_{A^{(n)}_\epsilon} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]

\[ \geq \int_{A^{(n)}_\epsilon} 2^{-n(h(X)+\epsilon)} dx_1:n = 2^{-n(h(X)+\epsilon)} \text{Vol}(A^{(n)}_\epsilon) \]
proof of theorem 16.6.4.

1: First,

\[ p(A^{(n)}_{\epsilon}) = \int_{x_1:n \in A^{(n)}_{\epsilon}} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  
\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  
for big enough \( n \) which follows from the WLLN.

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \geq \int_{A^{(n)}_{\epsilon}} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  
\[ \geq \int_{A^{(n)}_{\epsilon}} 2^{-n(h(X)+\epsilon)} \, dx_1:n = 2^{-n(h(X)+\epsilon)} \, \text{Vol}(A^{(n)}_{\epsilon}) \]  
\[ \Rightarrow \text{Vol}(A^{(n)}_{\epsilon}) \leq 2^n(h(X)+\epsilon). \]
proof of theorem 16.6.4 cont.

Similarly,

\[ 1 - \epsilon \leq \Pr(A_{\epsilon}^{(n)}) = \int_{A_{\epsilon}^{(n)}} f(x_1:n) dx_1:n \]

\[ \leq \int_{A_{\epsilon}^{(n)}} 2^{-n(h(X) - \epsilon)} dx_1:n = 2^{-n(h(X) - \epsilon)} \text{Vol}(A_{\epsilon}^{(n)}) \]  

(16.43)  

(16.44)
proof of theorem 16.6.4 cont.

Similarly,

\[ 1 - \epsilon \leq \Pr(A_{\epsilon}^{(n)}) = \int_{A_{\epsilon}^{(n)}} f(x_{1:n}) dx_{1:n} \quad (16.43) \]

\[ \leq \int_{A_{\epsilon}^{(n)}} 2^{-n(h(X) - \epsilon)} dx_{1:n} = 2^{-n(h(X) - \epsilon)} \text{Vol}(A_{\epsilon}^{(n)}) \quad (16.44) \]

Like in the discrete case, \( A_{\epsilon}^{(n)} \) is the smallest volume that contains, essentially, all of the probability, and that volume is \( \approx 2^{nh} \).
proof of theorem 16.6.4 cont.

Similarly,

$$1 - \epsilon \leq \Pr(A_{\epsilon}^{(n)}) = \int_{A_{\epsilon}^{(n)}} f(x_{1:n}) dx_{1:n} \quad (16.43)$$

$$\leq \int_{A_{\epsilon}^{(n)}} 2^{-n(h(X) - \epsilon)} dx_{1:n} = 2^{-n(h(X) - \epsilon)} \text{Vol}(A_{\epsilon}^{(n)}) \quad (16.44)$$

- Like in the discrete case, $A_{\epsilon}^{(n)}$ is the smallest volume that contains, essentially, all of the probability, and that volume is $\approx 2^{nh}$.
- If we look at $(2^{nh})^{1/n}$, we get a “side length” of $2^h$. 

---

Prof. Jeff Bilmes
EE514a/Fall 2013/Information Theory I – Lecture 16 - Nov 21st, 2013

F49/58 (pg.226/277)
Similarly,

\[ 1 - \epsilon \leq \Pr(\mathcal{A}(n)) = \int_{\mathcal{A}(n)} f(x_1:n) \, dx_1:n \]

\[ \leq \int_{\mathcal{A}(n)} 2^{-n(h(X)-\epsilon)} \, dx_1:n = 2^{-n(h(X)-\epsilon)} \text{Vol}(\mathcal{A}(n)) \]

Like in the discrete case, \( \mathcal{A}(n) \) is the smallest volume that contains, essentially, all of the probability, and that volume is \( \approx 2^{nh} \).

If we look at \( (2^{nh})^{1/n} \), we get a “side length” of \( 2^h \).

So, \( -\infty < h < \infty \) is a meaningful range for entropy since it is the exponent of the equivalent side length of the \( n \)-D volume.
proof of theorem 16.6.4 cont.

Similarly,

\[ 1 - \epsilon \leq \Pr(A_\epsilon^{(n)}) = \int_{A_\epsilon^{(n)}} f(x_1:n) dx_1:n \]

\[ \leq \int_{A_\epsilon^{(n)}} 2^{-n(h(X)-\epsilon)} dx_1:n = 2^{-n(h(X)-\epsilon)} \text{Vol}(A_\epsilon^{(n)}) \]  

(16.43) 

(16.44)

- Like in the discrete case, \( A_\epsilon^{(n)} \) is the smallest volume that contains, essentially, all of the probability, and that volume is \( \approx 2^{nh} \).
- If we look at \((2^{nh})^{1/n}\), we get a “side length” of \( 2^h \).
- So, \(-\infty < h < \infty \) is a meaningful range for entropy since it is the exponent of the equivalent side length of the \( n \)-D volume.
- Large negative entropy just means small volume.
Differential vs. Discrete Entropy

Let $X \sim f(x)$, and divide the range of $X$ up into bins of length $\Delta$. 

\[ f(x) \Delta = 2^{-n} \rightarrow |\Delta| \leftarrow x \]

Mean value theorem, i.e., that if continuous within bin $\exists x_i$ such that $f(x_i) = \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx$ (16.45)
Differential vs. Discrete Entropy

- Let $X \sim f(x)$, and divide the range of $X$ up into bins of length $\Delta$.
- E.g., quantize the range of $X$ using $n$ bits, so that $\Delta = 2^{-n}$. 
Differential vs. Discrete Entropy

- Let $X \sim f(x)$, and divide the range of $X$ up into bins of length $\Delta$.
- E.g., quantize the range of $X$ using $n$ bits, so that $\Delta = 2^{-n}$.
- We can then view this as follows:

\[ f(x) \quad \Delta = 2^{-n} \]

![Diagram showing quantization of a continuous distribution into discrete bins.]
Differential vs. Discrete Entropy

- Let \( X \sim f(x) \), and divide the range of \( X \) up into bins of length \( \Delta \).
- E.g., quantize the range of \( X \) using \( n \) bits, so that \( \Delta = 2^{-n} \).
- We can then view this as follows:

\[
\int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx
\]

(16.45)

Mean value theorem, i.e., that if continuous within bin \( \exists x_i \) such that

\[
f(x_i) = \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx
\]
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta \]  
  (16.46)
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  
  $X^\Delta = x_i$ if $i\Delta \leq X < (i + 1)\Delta$  
  
  (16.46)

- This gives a discrete distribution

  $$Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)dx = \Delta f(x_i)$$

  (16.47)
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta \]  
  (16.46)

- This gives a discrete distribution
  \[
  \Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)\,dx = \Delta f(x_i)
  \]  
  (16.47)

  and we can calculate the entropy

  \[
  H(X^\Delta)
  \]  
  (16.50)
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta \]  

- This gives a discrete distribution
  \[
  \Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)dx = \Delta f(x_i)
  \]

  and we can calculate the entropy
  \[
  H(X^\Delta) = -\sum_{i=-\infty}^{\infty} p_i \log p_i = -\sum_{i} f(x_i)\Delta \log(f(x_i)\Delta)
  \]
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that

$$X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta$$  \hfill (16.46)

- This gives a discrete distribution

$$\Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)\,dx = \Delta f(x_i)$$  \hfill (16.47)

and we can calculate the entropy

$$H(X^\Delta) = -\sum_{i=-\infty}^{\infty} p_i \log p_i = -\sum_i f(x_i)\Delta \log(f(x_i)\Delta)$$  \hfill (16.48)

$$= -\sum_i \Delta f(x_i) \log f(x_i) - \sum_i f(x_i)\Delta \log \Delta$$  \hfill (16.49)

$$= -\sum_i \Delta f(x_i) \log f(x_i) - \sum_i f(x_i)\Delta \log \Delta$$  \hfill (16.50)
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta \]  
  (16.46)

- This gives a discrete distribution
  \[
  \Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)dx = \Delta f(x_i)
  \]  
  (16.47)

and we can calculate the entropy

\[
H(X^\Delta) = - \sum_{i=-\infty}^{\infty} p_i \log p_i = - \sum_{i} f(x_i)\Delta \log(f(x_i)\Delta)
\]  
(16.48)

\[
= - \sum_{i} \Delta f(x_i) \log f(x_i) - \sum_{i} f(x_i)\Delta \log \Delta
\]  
(16.49)

\[
= - \sum_{i} \Delta f(x_i) \log f(x_i) - \log \Delta
\]  
(16.50)
This follows since (as expected)

\[ \sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1 \]

(16.51)
This follows since (as expected)

\[ \sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1 \]  

(16.51)

Also, as \( \Delta \to 0 \), we have \( -\log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)

\[ -\sum_i \Delta f(x_i) \log f(x_i) \to -\int f(x) \log f(x) \, dx \]  

(16.52)
Differential vs. Discrete Entropy

- This follows since (as expected)
  \[
  \sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x)dx = \Delta \frac{1}{\Delta} \int f(x)dx = 1
  \]
  (16.51)

- Also, as \( \Delta \to 0 \), we have \(- \log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)
  \[
  - \sum_i \Delta f(x_i) \log f(x_i) \to - \int f(x) \log f(x)dx
  \]
  (16.52)

- So, \( H(X^\Delta) + \log \Delta \to h(f) \) as \( \Delta \to 0 \).
This follows since (as expected)

\[ \sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1 \]  

(16.51)

Also, as \( \Delta \to 0 \), we have \( -\log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)

\[ -\sum_i \Delta f(x_i) \log f(x_i) \to -\int f(x) \log f(x) \, dx \]  

(16.52)

So, \( H(X^\Delta) + \log \Delta \to h(f) \) as \( \Delta \to 0 \).

Loosely, \( h(f) \approx H(X^\Delta) + \log \Delta \) and for an \( n \)-bit quantization with \( \Delta = 2^{-n} \), we have

\[ H(X^\Delta) \approx h(f) - \log \Delta = h(f) + n \]  

(16.53)
Differential vs. Discrete Entropy

This follows since (as expected)

\[
\sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1
\]

(16.51)

Also, as \( \Delta \to 0 \), we have \(- \log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)

\[
- \sum_i \Delta f(x_i) \log f(x_i) \to - \int f(x) \log f(x) \, dx
\]

(16.52)

So, \( H(X^\Delta) + \log \Delta \to h(f) \) as \( \Delta \to 0 \).

Loosely, \( h(f) \approx H(X^\Delta) + \log \Delta \) and for an \( n \)-bit quantization with \( \Delta = 2^{-n} \), we have

\[
H(X^\Delta) \approx h(f) - \log \Delta = h(f) + n
\]

(16.53)

This means that as \( n \to \infty \), \( H(X^\Delta) \) gets larger.
Joint Thm Coding Hamming Codes Differential Entropy

Differential vs. Discrete Entropy

This follows since (as expected)

\[ \sum_i \Delta f(x_i) = \Delta \sum_i \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1 \]

(16.51)

Also, as \( \Delta \to 0 \), we have \(-\log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)

\[ -\sum_i \Delta f(x_i) \log f(x_i) \to -\int f(x) \log f(x) \, dx \]

(16.52)

So, \( H(X^\Delta) + \log \Delta \to h(f) \) as \( \Delta \to 0 \).

Loosely, \( h(f) \approx H(X^\Delta) + \log \Delta \) and for an \( n \)-bit quantization with \( \Delta = 2^{-n} \), we have

\[ H(X^\Delta) \approx h(f) - \log \Delta = h(f) + n \]

(16.53)

This means that as \( n \to \infty \), \( H(X^\Delta) \) gets larger. Why?
This makes sense. We start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy.
Differential vs. Discrete Entropy

- This makes sense. We start with a continuous random variable \( X \) and quantize it at an \( n \)-bit accuracy.
- For a discrete representation to represent \( 2^n \) values, we expect the entropy to go up with \( n \), and as \( n \) gets large so would the entropy, but then adjusted by \( h(X) \).
Differential vs. Discrete Entropy

- This makes sense. We start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy.
- For a discrete representation to represent $2^n$ values, we expect the entropy to go up with $n$, and as $n$ gets large so would the entropy, but then adjusted by $h(X)$.
- $H(X^\Delta)$ is the number of bits to describe this $n$-bit equally spaced quantization of the continuous random variable $X$. 
This makes sense. We start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy.

For a discrete representation to represent $2^n$ values, we expect the entropy to go up with $n$, and as $n$ gets large so would the entropy, but then adjusted by $h(X)$.

$H(X^\Delta)$ is the number of bits to describe this $n$-bit equally spaced quantization of the continuous random variable $X$.

$H(X^\Delta) \approx h(f) + n$ says that it might take either more than $n$ bits to describe $X$ at $n$-bit accuracy, or less than $n$ bits to describe $X$ at $n$-bit accuracy, depending on the concentration of $X$. 
This makes sense. We start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy.

For a discrete representation to represent $2^n$ values, we expect the entropy to go up with $n$, and as $n$ gets large so would the entropy, but then adjusted by $h(X)$.

$H(X^\Delta)$ is the number of bits to describe this $n$-bit equally spaced quantization of the continuous random variable $X$.

$H(X^\Delta) \approx h(f) + n$ says that it might take either more than $n$ bits to describe $X$ at $n$-bit accuracy, or less than $n$ bits to describe $X$ at $n$-bit accuracy, depending on the concentration of $X$.

If $X$ is very concentrated $h(f) < 0$ then fewer bits. If $X$ is very spread out, then more than $n$ bits.
Like discrete case, we have entropy for vectors of r.v.s
Joint Differential Entropy

- Like discrete case, we have entropy for vectors of r.v.s
- The joint differential entropy is defined as:

\[
h(X_1, X_2, \ldots, X_n) = - \int f(x_{1:n}) \log f(x_{1:n}) dx_{1:n}\quad (16.54)
\]
**Joint Differential Entropy**

- Like discrete case, we have entropy for vectors of r.v.s
- The joint differential entropy is defined as:

\[ h(X_1, X_2, \ldots, X_n) = - \int f(x_1:n) \log f(x_1:n) dx_1:n \quad (16.54) \]

- Conditional differential entropy

\[ h(X|Y) = - \int f(x, y) \log f(x|y) dx dy = h(X, Y) - h(Y) \quad (16.55) \]
When $X$ is distributed according to a multivariate Gaussian distribution, i.e.,

$$X \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2} (x-\mu)^\top \Sigma^{-1} (x-\mu)}$$  \hspace{1cm} (16.56)
When $X$ is distributed according to a multivariate Gaussian distribution, i.e.,

$$X \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{|2\pi\Sigma|^{1/2}} e^{-\frac{1}{2} (x-\mu)^\top \Sigma^{-1} (x-\mu)} \quad (16.56)$$

then the entropy of $X$ has a nice form, in particular

$$h(X) = \frac{1}{2} \log \left[ (2\pi e)^n |\Sigma| \right] \text{ bits} \quad (16.57)$$
Entropy of a Multivariate Gaussian

• When $X$ is distributed according to a multivariate Gaussian distribution, i.e.,

$$X \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{|2\pi\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)} \quad (16.56)$$

then the entropy of $X$ has a nice form, in particular

$$h(X) = \frac{1}{2} \log \left[(2\pi e)^n |\Sigma|\right] \text{ bits} \quad (16.57)$$

• Notice that the entropy is monotonically related to the determinant of the covariance matrix $\Sigma$ and is not at all dependent on the mean $\mu$. 

Prof. Jeff Bilmes
EE514a/Fall 2013/Information Theory I – Lecture 16 - Nov 21st, 2013
F55/58 (pg.255/277)
When $X$ is distributed according to a multivariate Gaussian distribution, i.e.,

$$X \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{|2\pi \Sigma|^{1/2}} e^{-\frac{1}{2} (x-\mu)^\top \Sigma^{-1} (x-\mu)}$$  \hspace{1cm} (16.56)$$

then the entropy of $X$ has a nice form, in particular

$$h(X) = \frac{1}{2} \log \left[ (2\pi e)^n |\Sigma| \right] \text{ bits}$$  \hspace{1cm} (16.57)$$

Notice that the entropy is monotonically related to the determinant of the covariance matrix $\Sigma$ and is not at all dependent on the mean $\mu$.

The determinant is a form of spread, or dispersion of the distribution.
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) \]

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = -\int f(x) \left[ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]  

(16.58)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = - \int f(x) \left[ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]

(16.58)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

(16.59)

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = - \int f(x) \left[ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]

(16.58)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] + \frac{1}{2} \ln [(2\pi)^n |\Sigma|] \]

(16.59)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu) (x - \mu)^T \Sigma^{-1} \right] + \frac{1}{2} \ln [(2\pi)^n |\Sigma|] \]

(16.60)

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[
h(X) = -\int f(x) \left[ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right]
\]

\[
= \frac{1}{2} E_f \left[ \text{tr} \left( (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right)
\]

\[
= \frac{1}{2} E_f \left[ \text{tr} (x - \mu)(x - \mu)^T \Sigma^{-1} \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right)
\]

\[
= \frac{1}{2} \text{tr} E_f \left[ (x - \mu)(x - \mu)^T \right] \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right)
\]

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = - \int f(x) \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]  

(16.58)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)^T \Sigma^{-1} (x - \mu)\right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.59)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)(x - \mu)^T \Sigma^{-1} \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.60)

\[ = \frac{1}{2} \text{tr} E_f [(x - \mu)(x - \mu)^T] \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.61)

\[ = \frac{1}{2} \text{tr} \Sigma \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.62)

\[ = \frac{1}{2} \ln \left( (2\pi e)^n |\Sigma| \right) \]  

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = - \int f(x) \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]

\[ = \frac{1}{2} EF \left[ \text{tr} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

\[ = \frac{1}{2} E \left[ \text{tr} (x - \mu)(x - \mu)^T \Sigma^{-1} \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

\[ = \frac{1}{2} \text{tr} \left( EF \right) [(x - \mu)(x - \mu)^T] \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

\[ = \frac{1}{2} \text{tr} \Sigma \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

\[ = \frac{1}{2} \text{tr} I + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]

This uses the "trace trick", that \( \text{tr}(ABC) = \text{tr}(CAB) \).
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = -\int f(x) \left[ -\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu) - \ln \left( (2\pi)^{n/2}|\Sigma|^{1/2} \right) \right] \]

\[ \begin{align*} &\quad = \frac{1}{2} E_f \left[ \text{tr} \left( (x - \mu)(x - \mu)^T \Sigma^{-1} \right) \right] + \frac{1}{2} \ln \left( (2\pi)^n|\Sigma| \right) \\ &\quad = \frac{1}{2} \text{tr} \left( E_f \left[ (x - \mu)(x - \mu)^T \right] \Sigma^{-1} \right) + \frac{1}{2} \ln \left( (2\pi)^n|\Sigma| \right) \\ &\quad = \frac{1}{2} \text{tr} \Sigma \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n|\Sigma| \right) \\ &\quad = \frac{1}{2} \text{tr} I + \frac{1}{2} \ln \left( (2\pi)^n|\Sigma| \right) \\ &\quad = \frac{n}{2} + \frac{1}{2} \ln \left( (2\pi)^n|\Sigma| \right) \end{align*} \]
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = -\int f(x) \left[ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]  

(16.58)

\[ = \frac{1}{2} E_f \left[ \text{tr} \, (x - \mu)^T \Sigma^{-1} (x - \mu) \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.59)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)(x - \mu)^T \Sigma^{-1} \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.60)

\[ = \frac{1}{2} \text{tr} E_f [(x - \mu)(x - \mu)^T] \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.61)

\[ = \frac{1}{2} \text{tr} \Sigma \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.62)

\[ = \frac{1}{2} \text{tr} I + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  

(16.63)

\[ = \frac{n}{2} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) = \frac{1}{2} \ln \left( (2\pi e)^n |\Sigma| \right) \]  

(16.64)
Entropy of a Multivariate Gaussian: Derivation

\[ h(X) = -\int f(x) \left[ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) - \ln \left( (2\pi)^{n/2} |\Sigma|^{1/2} \right) \right] \]

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  \hspace{1cm} (16.58)

\[ = \frac{1}{2} E_f \left[ \text{tr} (x - \mu)(x - \mu)^\top \Sigma^{-1} \right] + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  \hspace{1cm} (16.59)

\[ = \frac{1}{2} \text{tr} E_f \left[ (x - \mu)(x - \mu)^\top \right] \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  \hspace{1cm} (16.60)

\[ = \frac{1}{2} \text{tr} \Sigma \Sigma^{-1} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  \hspace{1cm} (16.61)

\[ = \frac{1}{2} \text{tr} I + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) \]  \hspace{1cm} (16.62)

\[ = \frac{n}{2} + \frac{1}{2} \ln \left( (2\pi)^n |\Sigma| \right) = \frac{1}{2} \ln \left( (2\pi e)^n |\Sigma| \right) \]  \hspace{1cm} (16.63)

This uses the “trace trick”, that \( \text{tr}(ABC) = \text{tr}(CAB) \).
The relative entropy (or Kullback-Leibler divergence) for continuous distributions also has a familiar form

\[ D(f||g) = \int f(x) \log \frac{f(x)}{g(x)} \, dx \geq 0 \]  

(16.65)
The relative entropy (or Kullback-Leibler divergence) for continuous distributions also has a familiar form

\[ D(f||g) = \int f(x) \log \frac{f(x)}{g(x)} \, dx \geq 0 \]  

(16.65)

We can, like in the discrete case, use Jensen’s inequality to prove the non-negativity of \( D(f||g) \).
The relative entropy (or Kullback-Leibler divergence) for continuous distributions also has a familiar form

\[ D(f||g) = \int f(x) \log \frac{f(x)}{g(x)} \, dx \geq 0 \]  

(16.65)

We can, like in the discrete case, use Jensen’s inequality to prove the non-negativity of \( D(f||g) \).

Mutual Information:

\[ D(f(X,Y)||f(X)f(Y)) = I(X;Y) = h(X) - h(X|Y) \]  

(16.66)

\[ = h(Y) - h(Y|X) \geq 0 \]  

(16.67)
The relative entropy (or Kullback-Leibler divergence) for continuous distributions also has a familiar form

\[ D(f \| g) = \int f(x) \log \frac{f(x)}{g(x)} dx \geq 0 \]  

(16.65)

We can, like in the discrete case, use Jensen’s inequality to prove the non-negativity of \( D(f \| g) \).

**Mutual Information:**

\[ D(f(X,Y) \| f(X)f(Y)) = I(X;Y) = h(X) - h(X|Y) \]  

(16.66)

\[ = h(Y) - h(Y|X) \geq 0 \]  

(16.67)

Thus, since \( I(X;Y) \geq 0 \) we have again that conditioning reduces entropy, i.e., \( h(Y) \geq h(Y|X) \).
Chain rules and more

- We still have chain rules

\[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \]  

(16.68)

For discrete entropy, we have monotonicity. I.e.,

\[ H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}) \]

More generally,

\[ f(A) = H(X_A) \]

is monotonic non-decreasing in set \( A \) (i.e., \( f(A) \leq f(B), \forall A \subseteq B \)).

Is \( f(A) = h(X_A) \) monotonic?

No, consider Gaussian entropy with diagonal \( \Sigma \) with small diagonal values. So \( h(X) = \frac{1}{2} \log [(2\pi e)^n |\Sigma|] \) can get smaller with more random variables.

Similarly, when some variables independent, adding independent variables with negative entropy can decrease overall entropy.
Chain rules and more

- We still have chain rules
  \[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \] (16.68)

- And bounds of the form
  \[ \sum_i h(X_i | X_{1:n \setminus \{i\}}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \] (16.69)
Chain rules and more

- We still have chain rules
  
  \[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \]  
  (16.68)

- And bounds of the form
  
  \[ \sum_i h(X_i | X_{1:n \setminus \{i\}}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \]  
  (16.69)

- For discrete entropy, we have monotonicity. I.e.,
  
  \[ H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}). \]
Chain rules and more

- We still have chain rules

\[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \]  (16.68)

- And bounds of the form

\[ \sum_i h(X_i | X_{1:n \setminus \{i\}}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \]  (16.69)

- For discrete entropy, we have monotonicity. I.e.,

\[ H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}) \]. More generally

\[ f(A) = H(X_A) \]  (16.70)

is monotonic non-decreasing in set \( A \) (i.e., \( f(A) \leq f(B), \forall A \subseteq B \)).
Chain rules and more

- We still have chain rules
  \[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \]  
  (16.68)

- And bounds of the form
  \[ \sum_i h(X_i | X_{1:n \setminus \{i\}}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \]  
  (16.69)

- For discrete entropy, we have monotonicity. I.e.,
  \[ H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}) \]. More generally
  \[ f(A) = H(X_A) \]  
  (16.70)
  is monotonic non-decreasing in set \( A \) (i.e., \( f(A) \leq f(B), \forall A \subseteq B \)).
- Is \( f(A) = h(X_A) \) monotonic?
Chain rules and more

- We still have chain rules
  \[ h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \]  
  (16.68)

- And bounds of the form
  \[ \sum_i h(X_i | X_{1:n} \setminus \{i\}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \]  
  (16.69)

- For discrete entropy, we have monotonicity. I.e.,
  \[ H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}) \]. More generally
  \[ f(A) = H(X_A) \]  
  (16.70)
  is monotonic non-decreasing in set \( A \) (i.e., \( f(A) \leq f(B), \forall A \subseteq B \)).

- Is \( f(A) = h(X_A) \) monotonic? No, consider Gaussian entropy with diagonal \( \Sigma \) with small diagonal values. So \( h(X) = \frac{1}{2} \log [(2\pi e)^n |\Sigma|] \) can get smaller with more random variables.
Chain rules and more

- We still have chain rules

\[
    h(X_1, X_2, \ldots, X_n) = \sum_i h(X_i | X_{1:i-1}) \quad (16.68)
\]

- And bounds of the form

\[
    \sum_i h(X_i | X_{1:n} \setminus \{i\}) \leq h(X_1, X_2, \ldots, X_n) \leq \sum_i h(X_i) \quad (16.69)
\]

- For discrete entropy, we have monotonicity. I.e.,

\[
    H(X_1, X_2, \ldots, X_k) \leq H(X_1, X_2, \ldots, X_k, X_{k+1}). \quad \text{More generally}
\]

\[
    f(A) = H(X_A) \quad (16.70)
\]

is monotonic non-decreasing in set \( A \) (i.e., \( f(A) \leq f(B), \forall A \subseteq B \)).

- Is \( f(A) = h(X_A) \) monotonic? No, consider Gaussian entropy with diagonal \( \Sigma \) with small diagonal values. So \( h(X) = \frac{1}{2} \log \left( (2\pi e)^n |\Sigma| \right) \) can get smaller with more random variables.

- Similarly, when some variables independent, adding independent variables with negative entropy can decrease overall entropy.