Class Road Map - IT-I

L1 (9/25): Overview, Communications, Information, Entropy
L2 (9/30): Entropy, Mutual Information, KL-Divergence
L3 (10/2): More KL, Jensen, more Venn, Log Sum, Data Proc. Inequality
L5 (10/9): M. of Conv, AEP
L6 (10/14): AEP, Source Coding, Types
LX (10/16): Makeup
L7 (10/21): Types, Univ. Src Coding, Stoc. Procs, Entropy Rates
L8 (10/23): Entropy rates, HMMs, Coding
L9 (10/28): Kraft ineq., Shannon Codes, Kraft ineq. II, Huffman
L10 (10/30): Huffman, Shannon/Fano/Elias
L11 (11/4): Shannon/Fano/Elias, Games
LXX (11/6): In class midterm exam
L12 (11/11): Vet’s day, makeup lecture: Arith. Coding, Background On Channel Capacity
L13 (11/13): Channel Capacity, Ex. DMC
L15 (11/20): Joint AEP, Shannon’s 2nd Theorem,
L16 (11/25): Zero Error Codes, 2nd Thm Conv, Zero Error, $R = C$, Feedback, Joint Thm, Coding,
L17 (11/27): Hamming Codes, Differential Entropy
L18 (12/2):
L19 (12/4):
LXX (12/10): Final exam

Finals Week: December 9th–13th.
Cumulative Outstanding Reading TODOs

- Chapters 1 and 2 in our book (Cover & Thomas, “Information Theory”) (including Fano’s inequality).
- Chapters 3 and 4 in our book (Cover & Thomas, “Information Theory”).
- Sections 11.1 through 11.3 in our book (Cover & Thomas, “Information Theory”).
- Chapter 4 in our book (Cover & Thomas, “Information Theory”).
- Chapter 5 in our book (Cover & Thomas, “Information Theory”).
- Chapter 13 in our book (Cover & Thomas, “Information Theory”) (there is no chapter on arithmetic coding but the lecture slides will be complete, or see MacKay’s online text).
- Chapter 7 in our book (Cover & Thomas, “Information Theory”) on channel capacity.
- Chapter 8 and 9 in our book (Cover & Thomas, “Information Theory”) on differential entropy and the Gaussian channel.
Homework

- Homework 1 on our assignment dropbox (https://canvas.uw.edu/courses/1319497/assignments), was due Tuesday, Oct 8th, 11:55pm.
- Homework 2 on our assignment dropbox (https://canvas.uw.edu/courses/1319497/assignments), due Friday 10/18/2019, 11:45pm.
- Homework 3 on our assignment dropbox (https://canvas.uw.edu/courses/1319497/assignments), due Tuesday 10/29/2019, 11:45pm.
- Homework 5 on our assignment dropbox (https://canvas.uw.edu/courses/1319497/assignments), due Monday 12/2/2019, 11:45pm.
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? Exponentially large sized table sizes.

In all cases, we add enough redundancy to a message so that the original message can be decoded unambiguously.
Binary Symmetric Channel (BSC)

- A bit that is sent will be flipped with probability $p$.
- $p(Y = 1|X = 0) = p = 1 - p(Y = 0|X = 0)$. $p(Y = 0|X = 1) = p = p(Y = 1|X = 1)$.

The BSC is an important channel since it is a simple model but at the same time captures some of the difficulties of more complicated channels.

Q: can we still achieve reliable (“guaranteed” error free) communication with this channel? A: Yes, if $p < 1/2$ and if we do not ask for too high a transmission rate (which would be $R > C$), then we can.

Actually, any $p \neq 1/2$ is sufficient.

Intuition: think about AEP and/or block coding.

But how to compute $C$ the capacity?
Rather than send message \( x_1 x_2 \ldots x_n \) we repeat each symbol \( K \) times redundantly.

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

Message becomes

\[
x_1 x_1 \ldots x_1 \underbrace{x_2 x_2 \ldots x_2}_{k\times} \ldots
\]

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

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

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

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

Thus, this is not a good code.
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., $x_n \leftarrow \mod\left(\sum_{i=1}^{n-1} x_i, 2\right)$.
  - Thus a necessary condition for valid code word is:
    $$\mod\left(\sum_{i=1}^{n} x_i, 2\right) = 0.$$ 
- Any 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.
Best illustrated by an example.

Let $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. 
Best illustrated by an example.

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

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

Best illustrated by an example.

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

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
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.
Hamming Codes

Parity bits determined by the following equations:

\[ x_4 \equiv x_1 + x_2 + x_3 \pmod{2} \] (17.1)
\[ x_5 \equiv x_0 + x_2 + x_3 \pmod{2} \] (17.2)
\[ x_6 \equiv x_0 + x_1 + x_3 \pmod{2} \] (17.3)
(7, 4, 3) Hamming Codes

- Parity bits determined by the following equations:

\[
x_4 \equiv x_1 + x_2 + x_3 \mod 2 \tag{17.1}
\]
\[
x_5 \equiv x_0 + x_2 + x_3 \mod 2 \tag{17.2}
\]
\[
x_6 \equiv x_0 + x_1 + x_3 \mod 2 \tag{17.3}
\]

- 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:

\[
\begin{align*}
x_4 &\equiv x_1 + x_2 + x_3 \mod 2 \\
x_5 &\equiv x_0 + x_2 + x_3 \mod 2 \\
x_6 &\equiv x_0 + x_1 + x_3 \mod 2
\end{align*}
\] (17.1)

- 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).

\[
\begin{align*}
x_1 + x_2 + x_3 + x_4 &= 0 \\
x_0 + x_2 + x_3 + x_5 &= 0 \\
x_0 + x_1 + x_3 + x_6 &= 0
\end{align*}
\] (17.4)
Hamming Codes

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

$$H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix} \quad (17.5)$$
Hamming Codes

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

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

(17.5)

- 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
  \[
  H = \begin{pmatrix}
  0 & 1 & 1 & 1 & 1 & 0 & 0 \\
  1 & 0 & 1 & 1 & 0 & 1 & 0 \\
  1 & 1 & 0 & 1 & 0 & 0 & 1
  \end{pmatrix}
  \]  \hspace{1cm} (17.5)

- 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 $x^T = (x_1, x_2, \ldots, x_7)$ and
  
  \[
  H = \begin{pmatrix}
  0 & 1 & 1 & 1 & 1 & 0 & 0 \\
  1 & 0 & 1 & 1 & 0 & 1 & 0 \\
  1 & 1 & 0 & 1 & 0 & 0 & 1
  \end{pmatrix}
  \]

- 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} (17.5)

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

\[
\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100110 & \quad (17.6) \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101001 & \quad (17.7) \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110000 & \quad (17.8) \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111111 & \quad (17.9)
\end{align*}
\]
The 16 vectors in the nullspace (i.e., \( \{ x : Hx = 0 \} \)) are as follows:

\[
\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100110 \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101001 \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110000 \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111111 \\
\end{align*}
\] (17.6) (17.7) (17.8) (17.9)

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).
Hamming Codes

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

  \[
  \begin{align*}
  0000000 & \quad 0100101 & \quad 1000011 & \quad 1100110 \\
  0001111 & \quad 0101010 & \quad 1001100 & \quad 1101001 \\
  0010110 & \quad 0110011 & \quad 1010101 & \quad 1110000 \\
  0011001 & \quad 0111100 & \quad 1011010 & \quad 1111111
  \end{align*}
  \]

  (17.6) (17.7) (17.8) (17.9)

- 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:

\[
\begin{align*}
0000000 & \quad 0100101 & \quad 1000011 & \quad 1100110 \\
0001111 & \quad 0101010 & \quad 1001100 & \quad 1101001 \\
0010110 & \quad 0110011 & \quad 1010101 & \quad 1110000 \\
0011001 & \quad 0111100 & \quad 1011010 & \quad 1111111
\end{align*}
\]

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 vectors constitute the codewords, any codeword must be one of the above.
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).

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

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

- Thus, any codeword is in $C = \{x : Hx = 0\}$.
- Minimum distance of a code is also 3, which is minimum number of differences between any two codewords.
- 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? Recall, given \( v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C \), and hence \( H(v_1 - v_2) = 0 \). Suppose \( v_1, v_2 \in C \) differ in only two places. Then \( H(v_1 - v_2) \) will be a difference or sum (mod 2) of two columns of \( H \), say \( H_{v_1} \) and \( H_{v_2} \). Hence, \( 0 = H(v_1 - v_2) = H_{v_1} - H_{v_2} \neq 0 \), a contradiction. Hence, \( v_1 - v_2 \) 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? Recall, given $v_1, v_2 \in C \Rightarrow (v_1 - v_2) \in C$, and hence $H(v_1 - v_2) = 0$. Suppose $v_1, v_2 \in C$ differ in only two places. Then $H(v_1 - v_2)$ will be a difference or sum (mod 2) of two columns of $H$, say $H_{v_1}$ and $H_{v_2}$. Hence, $0 = H(v_1 - v_2) = H_{v_1} - H_{v_2} \neq 0$, a contradiction. Hence, $v_1 - v_2$ 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 verse.
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

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

(17.10)

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

$$y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6) \quad (17.10)$$

where $z = (z_0, z_2, \ldots, z_6)$ is the noise vector.
- Receiver knows $y$ but wants to know $x$. 
Now a BSC\((p)\) (crossover probability \(p\)) will chance some of the bits (noise), meaning a 0 might change to a 1 and vice versa.

So if \(x = (x_0, x_2, \ldots, x_6)\) is transmitted, what is received is

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  \hspace{1cm} (17.10)

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

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

\[
\text{s} = H y = H (x + z) = H x + Hz
\]  \hspace{1cm} (17.11)
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  \hspace{1cm} (17.10)

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

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

\[
s
\]  \hspace{1cm} (17.11)
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

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

(17.10)

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

\[
s = Hy
\]

(17.11)
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

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

(17.10)

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

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

(17.11)
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
  \[ y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6) \]  
  (17.10)

  where $z = (z_0, z_2, \ldots, z_6)$ is the noise vector.
- Receiver knows $y$ but wants to know $x$. We then compute
  \[ s = Hy = H(x + z) = Hx + Hz \]  
  (17.11)
Hamming Codes : BSC

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

So if $x = (x_0, x_2, \ldots, x_6)$ is transmitted, what is received is

$$y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)$$  \hspace{1cm} (17.10)

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

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

$$s = Hy = H(x + z) = Hx + Hz$$  \hspace{1cm} (17.11)

$s$ 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.
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  
(17.10)

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

\[
s = Hy = H(x + z) = \underbrace{Hx}_{=0} + Hz = Hz
\]  
(17.11)
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

\[
y = x + z = (x_0 + z_0, x_1 + z_1, \ldots, x_6 + z_6)
\]  \(\text{(17.10)}\)

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

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

\[
s = Hy = H(x + z) = Hx + Hz = Hz
\]  \(\text{(17.11)}\)

\(s\) 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.
Hamming Codes: BSC

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

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

(17.12)
Hamming Codes : BSC

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

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

- We know $y$, and since $y = x + z$, then if we know $z$ we know $x$. 

$$
(17.12)
$$
Hamming Codes : BSC

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

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (17.12)$$

- We know $y$, and since $y = x + z$, then 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 $H$

$$s = z_0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + z_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \cdots + z_6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (17.12)$$

We know $y$, and since $y = x + z$, then if we know $z$ we know $x$.

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

Ex: Suppose that $y^\top = 0111001$ is received (which is not a codeword), then $s = H y = (101)^\top$ and the 16 solutions for $z$ are:

$$\begin{array}{cccc}
0100000 & 0010011 & 0101111 & 1001001 \\
1100011 & 0001010 & 1000110 & 1111010 \\
0000101 & 0111001 & 1110101 & 0011100 \\
0110110 & 1010000 & 1101100 & 1011111 \\
\end{array}$$
Hamming Codes: BSC

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

- 16 is better than 128 (possible $z$ vectors) but still many.
- What is the probability of each solution? Since we are assuming a BSC($p$) with $p < 1/2$, the most probable solution for $z$ has the least weight. Any solution with weight $k$ has probability $p^k(1 - p)^{m-k}$.
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 $p^k(1 - p)^{m-k}$.
- There's only one possible solution with weight 1, and this is most probable solution since $p < 1/2$, a single error bit.
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 \( p^k(1 - p)^{m-k} \).

There's only one possible solution with weight 1, and this is most probable solution since \( p < 1/2 \), a single error bit.

In previous example, most probable solution is \( z^\top = (01000000) \) and in \( y = x + z \) with \( y^\top = 0111001 \) 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 $p^k(1 - p)^{m-k}$.
- There's only one possible solution with weight 1, and this is most probable solution since $p < 1/2$, a single error bit.
- 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 $s = Hz$. This weight is no more than 1.
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 \( p^k (1 - p)^{m-k} \).

There's only one possible solution with weight 1, and this is most probable solution since \( p < 1/2 \), a single error bit.

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 \( s = Hz \). 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 $p^k(1 - p)^{m-k}$.
- There's only one possible solution with weight 1, and this is most probable solution since $p < 1/2$, a single error bit.
- 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 $s = H^Tz$. This weight is no more than 1.
- If $s = (000)$ then the unique solution is $z = (0000000)$.
- For any other $s$, in minimum weight case (single bit error), $s$ 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 (syndrome decoding) on receiving $y$:

1. Compute the syndrome $s = Hy$. 
Hamming Decoding Procedure

Here is the final decoding procedure (syndrome decoding) 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.
Hamming Decoding Procedure

Here is the final decoding procedure (syndrome decoding) 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.
Hamming Decoding Procedure

Here is the final decoding procedure (syndrome decoding) 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 $x \leftarrow y + z$. 

This procedure can correct any single bit error, but fails when there is more than one error.
Here is the final decoding procedure (syndrome decoding) 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.
Hamming Decoding Procedure

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

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

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

- We can visualize the decoding procedure using Venn Diagrams

(a)

(b)

Here, the 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 \((x_0, x_2, ..., x_6) = (1, 0, 0, 0, 1, 0, 1)\) with parity check bits:

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

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

\[
x_6 \equiv x_0 + x_2 + x_3 \pmod{2}
\]
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:

\[
\begin{align*}
x_4 &\equiv x_0 + x_1 + x_2 \pmod{2} \quad (17.13) \\
x_5 &\equiv x_1 + x_2 + x_3 \pmod{2} \quad (17.14) \\
x_6 &\equiv x_0 + x_2 + x_3 \pmod{2} \quad (17.15)
\end{align*}
\]
Hamming Decoding: Venn Diagrams

- 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.
Example: Here, $z_1$ can be flipped to achieve parity.
Example: Here, \( z_4 \) can be flipped to achieve parity.
Example: And here, $z_2$ can be flipped to achieve parity.
Example: And here, there are two errors, $y_6$ and $y_2$ (each of which are marked with a *).

![Venn Diagram](image)

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

Example: And here, there are two errors, $y_6$ and $y_2$ (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.

- 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.
- Recent modifications and enhancements to the above.
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.
- Recent modifications and enhancements to the above.

All developed in order 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.
- Recent modifications and enhancements to the above.

All developed in order to find good codes with low rate that achieve Shannon’s promise.

Discussion of LDPC and Turbo codes would be good in a class on coding theory
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.
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,
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.
Continuous/Differential Entropy

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

\[ F(x) = \text{Pr}(X \leq x) \quad (17.16) \]

and \( f(x) = \frac{d}{dx} F(x) \) is the density function.
Let $X$ now be a continuous r.v. with cumulative distribution

$$F(x) = \Pr(X \leq x) \quad (17.16)$$

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

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

$$F(x) = \Pr(X \leq x) \quad (17.16)$$

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

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

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

$$F(x) = \Pr(X \leq x) \quad (17.16)$$

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

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

**Definition 17.4.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) dx \quad (17.17)$$
Continuous/Differential Entropy

- Let $X$ now be a continuous r.v. with cumulative distribution
  \[ F(x) = \Pr(X \leq x) \quad (17.16) \]
  and $f(x) = \frac{d}{dx} F(x)$ is the density function.
- Let $S = \{ x : f(x) > 0 \}$ be the support set. Then

**Definition 17.4.1 (differential entropy $h(X)$)**

\[ h(X) = - \int_S f(x) \log f(x) \, dx \quad (17.17) \]

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

---

Profs. Jeff Bilmes
EE514a/Fall 2019/Info. Theory I – Lecture 17 - Nov 27th, 2019
L17 F27/38 (pg.90/167)
Continuous/Differential Entropy

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

$$F(x) = \Pr(X \leq x)$$  \hspace{1cm} (17.16)

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

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

**Definition 17.4.1 (differential entropy $h(X)$)**

$$h(X) = - \int_S f(x) \log f(x) \, dx$$  \hspace{1cm} (17.17)

- Since we integrate over only the support set, no worries about $\log 0$.
- Let's jump right into some examples . . .
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

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 “count” or “volume” of a typical set.

Example: $2^H(X)$ is the number of things that happen, on average. We easily can have that $2^H(X) \ll |X|$ which is what allows compression.

Consider a uniform r.v. $Y$ such that $2^H(Y) = |Y|$. Thus, negative exponent means the count, or “volume” is small.
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = - \int_0^a \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a} \quad (17.18)$$
Continuous Entropy Of Uniform Distribution

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

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a} \quad (17.18)$$

- Note: continuous entropy can be both positive or negative.
Continuous Entropy Of Uniform Distribution

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

Then

$$h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} dx = - \log \frac{1}{a}$$  \hspace{1cm} (17.18)

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

\[
 h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} \, dx = -\log \frac{1}{a} \quad (17.18)
\]

- 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 “count” or “volume” of a typical set.
Here, $X \sim U[0, a]$ with $a \in \mathbb{R}_{++}$.

Then

$$h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = -\log \frac{1}{a} \quad (17.18)$$

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 “count” or “volume” of a typical set.

Example: $2^H(X)$ is the number of things that happen, on average. We easily can have that $2^H(X) \ll |\mathcal{X}|$ which is what allows compression.
Continuous Entropy Of Uniform Distribution

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

\[
h(X) = - \int_{0}^{a} \frac{1}{a} \log \frac{1}{a} \, dx = - \log \frac{1}{a} \quad (17.18)
\]

- 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 “count” or “volume” of a typical set.
- Example: $2^{H(X)}$ is the number of things that happen, on average. We easily can have that $2^{H(X)} \ll |X|$ which is what allows compression.
- Consider a uniform r.v. $Y$ such that $2^{H(Y)} = |Y|$.
Continuous Entropy Of Uniform Distribution

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

Then

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

(17.18)

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 “count” or “volume” of a typical set.

Example: $2^H(X)$ is the number of things that happen, on average. We easily can have that $2^H(X) \ll |X|$ which is what allows compression.

Consider a uniform r.v. $Y$ such that $2^H(Y) = |Y|$.

Thus, negative exponent means the count, or “volume” is small.
Normal (Gaussian) distributions are very important.
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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.

Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19) $$

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

$$ \text{EX}^2 = \frac{1}{2} \log(2\pi e\sigma^2) \text{ bits} \quad (17.22) $$

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.

Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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.
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:
  \[ X \sim N(0, \sigma^2) \quad \Leftrightarrow \quad f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19) \]

- Let's compute the entropy of $f$ in nats.
  \[ h(X) = -\int f \ln f \]
  \[ (17.22) \]
Normal (Gaussian) distributions are very important. 
Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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

$$h(X) = -\int f \ln f = -\int f(x) \left[-\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2}\right] dx \quad (17.20)$$

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

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:
  \[ X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19) \]
- Let's compute the entropy of $f$ in nats.
  \[ h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (17.20) \]
  \[ \frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) \]
  \[ \quad (17.22) \]
Normal (Gaussian) distributions are very important.

Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \Leftrightarrow f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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

$$h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (17.20)$$

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (17.21)$$

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

So entropy of a Gaussian is monotonically related to the variance.
Normal (Gaussian) distributions are very important. Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2}$$ \hspace{1cm} (17.19)

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} (17.20)

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2)$$ \hspace{1cm} (17.21)

$$= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2)$$ \hspace{1cm} (17.22)
Normal (Gaussian) distributions are very important.

Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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

$$h(X) = -\int f \ln f = -\int f(x) \left[-\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2}\right] dx \quad (17.20)$$

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (17.21)$$

$$= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} \ln(2\pi e\sigma^2) \text{ nats} \quad (17.22)$$
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2}$$  \hspace{1cm} (17.19)

- 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} (17.20)

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2)$$  \hspace{1cm} (17.21)

$$= \frac{1}{2} \ln e + \frac{1}{2} \ln(2\pi\sigma^2) + \frac{1}{2} \ln(2\pi e\sigma^2) \text{nats} \times \left( \frac{1}{\ln 2} \text{bits/nats} \right)$$  \hspace{1cm} (17.22)
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \iff f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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

$$h(X) = -\int f \ln f = -\int f(x) \left[ -\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2} \right] dx \quad (17.20)$$

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (17.21)$$

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

$$= \frac{1}{2} \log(2\pi e\sigma^2) \text{bits} \quad (17.22)$$
Normal (Gaussian) distributions are very important. Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:

$$X \sim N(0, \sigma^2) \Leftrightarrow f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2} \quad (17.19)$$

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

$$h(X) = -\int f \ln f = -\int f(x) \left[-\frac{x^2}{2\sigma^2} - \ln \sqrt{2\pi\sigma^2}\right] dx \quad (17.20)$$

$$\frac{EX^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma^2) = \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2) \quad (17.21)$$

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

$$= \frac{1}{2} \log(2\pi e\sigma^2) \text{bits} \quad (17.22)$$

Note: only a function of the variance $\sigma^2$, not the mean. Why?
Continuous Entropy Of Normal Distribution

- Normal (Gaussian) distributions are very important.
- Univariate (1D) Gaussian with 0-mean and variance $\sigma^2$:
  
  $$X \sim N(0, \sigma^2) \Leftrightarrow f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2}x^2/\sigma^2}$$  

  (17.19)

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

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

  (17.20)

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

  (17.21)

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

  $$= \frac{1}{2} \log(2\pi e\sigma^2) \text{bits}$$  

  (17.22)

- 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 \( |A_\epsilon^{(n)}| = 2^{nH} = (2^H)^n \).
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 $|A^{(n)}_\epsilon| = 2^{nH} = (2^H)^n$.

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

- We even have our own AEP in the continuous case, but before that a bit more intuition.
- In the discrete case, we have $P_r(x_1, x_2, \ldots, x_n) \approx 2^{-nH(X)}$ 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 17.4.2**

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

$$
- \frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[- \log f(X)] = h(X)
$$  \hspace{1cm} (17.23)
AEP

- Things are similar for the continuous case. Indeed

**Theorem 17.4.2**

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

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (17.23)$$

- 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 17.4.2**

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

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (17.23)$$

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

**Definition 17.4.3**

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

**Theorem 17.4.2**

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

$$-rac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (17.23)$$

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

**Definition 17.4.3**

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

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

---

**Prof. Jeff Bilmes**

EE514a/Fall 2019/Info. Theory I – Lecture 17 - Nov 27th, 2019

L17 F31/38 (pg.122/167)
Things are similar for the continuous case. Indeed

**Theorem 17.4.2**

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

$$-\frac{1}{n} \log f(X_1, X_2, \ldots, X_n) \to E[-\log f(X)] = h(X) \quad (17.23)$$

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

**Definition 17.4.3**

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

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

Thus, we have upper/lower bounds on the probability

$$2^{-n(h+\epsilon)} \leq f(x_{1:n}) \leq 2^{-n(h-\epsilon)} \quad (17.24)$$
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

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

(17.25)
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

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

(17.25)

then we have

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

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.
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} (17.25)

then we have

**Theorem 17.4.4**
The volume of $A \subseteq \mathbb{R}^n$ is well defined as:

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (17.25)

then we have

**Theorem 17.4.4**

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

$$\text{Vol}(A) = \int_A dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (17.25)

then we have

**Theorem 17.4.4**

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} (17.25)

then we have

**Theorem 17.4.4**

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

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

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

$$\text{Vol}(A) = \int_{A} dx_1 dx_2 \ldots dx_n$$  \hspace{1cm} (17.25)

then we have

Theorem 17.4.4

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 17.4.4.

1: First,

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

(17.27)
proof of theorem 17.4.4.

1: First,

\[ p(A^{(n)}_c) = \int_{x_1:n \in A^{(n)}_c} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  \hspace{1cm} (17.26)

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

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \geq \int_{A^{(n)}_c} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  \hspace{1cm} (17.28)

\[ \geq \int_{A^{(n)}_c} 2^{-n(h(X)+\epsilon)} \, dx_1 \ldots dx_n \]  \hspace{1cm} (17.29)

\[ \Rightarrow \text{Vol}(A^{(n)}_c) \leq 2^{-n(h(X)+\epsilon)} \text{Vol}(A^{(n)}_c) \]
proof of theorem 17.4.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  (17.26)

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  (17.27)
proof of theorem 17.4.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  \hspace{1cm} (17.26)

\[ = \Pr \left( \left\{ -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right\} \leq \epsilon \right) \geq 1 - \epsilon \]  \hspace{1cm} (17.27)

for big enough \( n \) which follows from the WLLN.
proof of theorem 17.4.4.

1: First,

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

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

(17.26)  

(17.27)

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

2: Next, we have

(17.29)
proof of theorem 17.4.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \] (17.26)

\[ = \Pr \left( \left| -\frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon (17.27) \]

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

2: Next, we have

\[ \] (17.29)
AEP

proof of theorem 17.4.4.

1: First,

\[ p(A^{(n)}_\epsilon) = \int_{x_1:n \in A^{(n)}_\epsilon} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  
(17.26)

\[ = \Pr \left( \left| \frac{1}{n} f(x_1, x_2, \ldots, x_n) - h(X) \right| \leq \epsilon \right) \geq 1 - \epsilon \]  
(17.27)

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

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) \, dx_1 \ldots dx_n \]  
(17.29)
proof of theorem 17.4.4.

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

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

\[ \geq \int_{A^{(n)}_{\epsilon}} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \quad (17.28) \]

(17.29)
proof of theorem 17.4.4.

1: First,

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

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

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

2: Next, we have

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

\[
\geq \int_{A_{\epsilon}^{(n)}} 2^{-n(h(X) + \epsilon)} \, dx_1:n
\]
proof of theorem 17.4.4.

1: First, 

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

(17.26)

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

(17.27)

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  

(17.28)

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

(17.29)
AEP

proof of theorem 17.4.4.

1: First,

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

2: Next, we have

\[ 1 = \int_{S^n} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \geq \int_{A_{\varepsilon}^{(n)}} f(x_1, \ldots, x_n) dx_1 \ldots dx_n \]  
\[ \geq \int_{A_{\varepsilon}^{(n)}} 2^{-n(h(X)+\varepsilon)} dx_1:n = 2^{-n(h(X)+\varepsilon)} \text{Vol}(A_{\varepsilon}^{(n)}) \]  
\[ \Rightarrow \text{Vol}(A_{\varepsilon}^{(n)}) \leq 2^{n(h(X)+\varepsilon)}. \]
proof of theorem 17.4.4 cont.

Similarly,

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

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

(17.30)  

(17.31)
proof of theorem 17.4.4 cont.

Similarly,

\[ 1 - \epsilon \leq \Pr(A_\epsilon^{(n)}) = \int_{A_\epsilon^{(n)}} f(x_{1:n}) \, dx_{1:n} \]  \hspace{1cm} (17.30)

\[ \leq \int_{A_\epsilon^{(n)}} 2^{-n(h(X)-\epsilon)} \, dx_{1:n} = 2^{-n(h(X)-\epsilon)} \cancel{\text{Vol}(A_\epsilon^{(n)})} \]  \hspace{1cm} (17.31)

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

Proof of Theorem 17.4.4 cont.

Similarly,

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

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

(17.30) \hspace{4cm} (17.31)

- 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 \( 2^h \).
proof of theorem 17.4.4 cont.

Similarly,

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

$$\leq \int_{A_{\epsilon}(n)} 2^{-n(h(X)-\epsilon)} dx_1:n = 2^{-n(h(X)-\epsilon)} \text{Vol}(A_{\epsilon}(n))$$  \hspace{1cm} (17.30) \hspace{1cm} (17.31)

- 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.
proof of theorem 17.4.4 cont.

Similarly,

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

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

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 magnitude negative entropy just means small volume.
Let $X \sim f(x)$, and divide the range of $X$ up into bins of length $\Delta$. 

E.g., quantize range of $X$ using $n$ bits, $2^n$ values, so $\Delta = 2^{-n}$. We can then view this as follows:

$$f(x) \Delta = 2^{-n} \rightarrow |\Delta| \rightarrow x$$

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

$$f(x_i) = \frac{1}{\Delta} \int_{i \Delta}^{(i+1) \Delta} f(x) \, dx \quad (17.32)$$

$$f(x_i) \Delta (i+1) \Delta x_i$$

\[ \text{Prof. Jeff Bilmes} \]
Differential vs. Discrete Entropy

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

E.g., quantize range of $X$ using $n$ bits, $2^n$ values, so $\Delta = 2^{-n}$.

We can then view this as follows:
Differential vs. Discrete Entropy

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

E.g., quantize range of $X$ using $n$ bits, $2^n$ values, so $\Delta = 2^{-n}$.

We can then view this as follows:

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

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

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

(17.32)
Differential vs. Discrete Entropy

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

$$X^\Delta = x_i \quad \text{if} \quad i\Delta \leq X < (i + 1)\Delta \quad (17.33)$$
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \quad \text{if} \quad i\Delta \leq X < (i + 1)\Delta \]  
  (17.33)

- This gives a discrete distribution
  \[
  \Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x) dx = \Delta \times f(x_i)
  \]  
  (17.34)
Create a quantized random variable $X^\Delta$ having those values so that

$$X^\Delta = x_i \text{ if } i\Delta \leq X < (i + 1)\Delta$$  \hspace{1cm} (17.33)

This gives a discrete distribution

$$\Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)dx = \Delta \times f(x_i)$$  \hspace{1cm} (17.34)

and we can calculate the entropy

$$H(X^\Delta)$$  \hspace{1cm} (17.37)
Differential vs. Discrete Entropy

- Create a quantized random variable \( X^\Delta \) having those values so that
  \[
  X^\Delta = x_i \quad \text{if} \quad i\Delta \leq X < (i+1)\Delta
  \]  
  (17.33)

- This gives a discrete distribution

\[
\Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \times f(x_i)
\]  
  (17.34)

and we can calculate the entropy

\[
H(X^\Delta) = - \sum_{i=-\infty}^{\infty} p_i \log p_i = - \sum_{i} (\Delta f(x_i)) \log (f(x_i)\Delta)
\]  
  (17.35)
Differential vs. Discrete Entropy

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

$$X^\Delta = x_i \quad \text{if} \quad i\Delta \leq X < (i+1)\Delta \quad (17.33)$$

- This gives a discrete distribution

$$\Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \times f(x_i) \quad (17.34)$$

and we can calculate the entropy

$$H(X^\Delta) = -\sum_{i=-\infty}^{\infty} p_i \log p_i = -\sum_i (\Delta f(x_i)) \log(f(x_i)\Delta) \quad (17.35)$$

$$= -\sum_i \Delta f(x_i) \log f(x_i) - \sum_i \Delta f(x_i) \log \Delta \quad (17.36)$$

$$= H(X) - \log \Delta \quad (17.37)$$
Differential vs. Discrete Entropy

- Create a quantized random variable $X^\Delta$ having those values so that
  \[ X^\Delta = x_i \quad \text{if} \quad i\Delta \leq X < (i + 1)\Delta \tag{17.33} \]

- This gives a discrete distribution
  \[
  \Pr(X^\Delta = x_i) = p_i = \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx = \Delta \times f(x_i) \tag{17.34}
  \]
  and we can calculate the entropy
  \[
  H(X^\Delta) = - \sum_{i=-\infty}^{\infty} p_i \log p_i = - \sum_i (\Delta f(x_i)) \log(f(x_i)\Delta) \tag{17.35}
  \]
  \[
  = - \sum_i \Delta f(x_i) \log f(x_i) - \sum_i \Delta f(x_i) \log \Delta \tag{17.36}
  \]
  \[
  = - \sum_i \Delta f(x_i) \log f(x_i) - \log \Delta \tag{17.37}
  \]
Differential vs. Discrete Entropy

- This follows since (as expected)

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

(17.38)

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

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

(17.39)

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

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

(17.40)

This means that as \( n \to \infty \), \( H(X_\Delta) \) can get larger.
Differential vs. Discrete Entropy

- This follows since (as expected)
  \[
  \sum_i \Delta f(x_i) = \sum_i \Delta \left( \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx \right) = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1
  \]
  (17.38)

- Also, as \( \Delta \rightarrow 0 \), we have \( -\log \Delta \rightarrow \infty \) and (assuming all is integrable in a Riemannian sense)
  \[
  -\sum_i \Delta f(x_i) \log f(x_i) \rightarrow -\int f(x) \log f(x) \, dx
  \]
  (17.39)
Differential vs. Discrete Entropy

- This follows since (as expected)
  \[ \sum_i \Delta f(x_i) = \sum_i \Delta \left( \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx \right) = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1 \]
  \hspace{1cm} (17.38)

- Also, as \( \Delta \to 0 \), we have \( -\log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)
  \[ -\sum_i \Delta f(x_i) \log f(x_i) \to -\int f(x) \log f(x) \, dx \]  
  \hspace{1cm} (17.39)

- So, \( H(X^\Delta) + \log \Delta \to h(f) \) as \( \Delta \to 0 \).
Differential vs. Discrete Entropy

- This follows since (as expected)
  \[
  \sum_i \Delta f(x_i) = \sum_i \Delta \left( \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) \, dx \right) = \Delta \frac{1}{\Delta} \int f(x) \, dx = 1
  \]
  (17.38)

- Also, as \( \Delta \to 0 \), we have \(- \log \Delta \to \infty \) and (assuming all is integrable in a Riemannian sense)
  \[
  - \sum_i \Delta f(x_i) \log f(x_i) \to - \int f(x) \log f(x) \, dx
  \]
  (17.39)

- 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
  \[
  H(X^{\Delta}) \approx h(f) - \log \Delta = h(f) + n
  \]
  (17.40)
Differential vs. Discrete Entropy

- This follows since (as expected)

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

(17.38)

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

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

(17.39)

- 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

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

(17.40)

- This means that as \( n \to \infty \), \( H(X^\Delta) \) can get larger.
Differential vs. Discrete Entropy

- This follows since (as expected)

\[
\sum_i \Delta f(x_i) = \sum_i \Delta \left( \frac{1}{\Delta} \int_{i\Delta}^{(i+1)\Delta} f(x) dx \right) = \Delta \frac{1}{\Delta} \int f(x) dx = 1
\]  
(17.38)

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

\[
- \sum_i \Delta f(x_i) \log f(x_i) \to - \int f(x) \log f(x) dx
\]  
(17.39)

- 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

\[
H(X^\Delta) \approx h(f) - \log \Delta = h(f) + n
\]  
(17.40)

- This means that as \( n \to \infty \), \( H(X^\Delta) \) can get larger. Why?
Differential vs. Discrete Entropy

- I.e., we start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy, so $2^n$ values.
Differential vs. Discrete Entropy

- I.e., we start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy, so $2^n$ values.

- 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 $h(X)$ via
  \[ H(X^\Delta) \approx h(X) - \log \Delta = h(X) - \log 2^{-n} = h(X) + n \]
Differential vs. Discrete Entropy

- I.e., we start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy, so $2^n$ values.

- 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 $h(X)$ via
  \[ H(X^{\Delta}) \approx h(X) - \log \Delta = h(X) - \log 2^{-n} = h(X) + n \]

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

- I.e., we start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy, so $2^n$ values.

- 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 $h(X)$ via

$$H(X^\Delta) \approx h(X) - \log \Delta = h(X) - \log 2^{-n} = h(X) + n$$

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

- $H(X^\Delta) \approx h(X) + n$ 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 $X$. 

Differential vs. Discrete Entropy

- I.e., we start with a continuous random variable $X$ and quantize it at an $n$-bit accuracy, so $2^n$ values.
- 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 $h(X)$ via
  
  \[ H(X^\Delta) \approx h(X) - \log \Delta = h(X) - \log 2^{-n} = h(X) + n \]

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